МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____Скарга-Бандурова І.С.
«____»_____ 2018 р.

ДИПЛОМНИЙ ПРОЕКТ (РОБОТА) БАКАЛАВРА

ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

Розроблення сервісу для автоматизованої системи обліку та тарифікації наданих послуг

Освітньо-кваліфікаційний рівень "бакалавр"
Напрям 6.050102 – "Комп'ютерна інженерія"

Керівник проекту:	(підпис)	доц. Барбарук В.М. (ініціали, прізвище)
Консультант з охорони праці:	(підпис)	ст.викл.Критська Я.О. (ініціали, прізвище)
Здобувач вищої освіти:	(підпис)	Газашвілі Д.В. (ініціали, прізвище)
Група:		КІ-14з

Сєвєродонецьк 2018

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень бакалавр
Напрям підготовки 6.050102 Комп'ютерна інженерія
(шифр і назва)
Спеціальність
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри
І.С. Скарга-Бандурова
«_____» _____2018 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) БАКАЛАВРА

Газашвілі Дмитру Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення сервісу для автоматизованої системи обліку та тарифікації наданих послуг

керівник проекту (роботи) Барбарук В.М., к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "14 " 05 2018р. №
2. Термін подання студентом роботи
3. Вихідні дані до роботи матеріали переддипломної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Огляд методів, підходів, технологій до проектування та розробки інформаційних систем. Розробка бази даних. Проектування та розробка програмної системи. Тестування та ілюстрація роботи системи. Охорона праці. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	ст.викл. кафедри КНІ Критська Я.О.		

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Огляд літератури з теми ДП і постановка задачі	14.05.18-19.05.18	
2	Розробка бази даних.	20.05.18-25.05.18	
3	Розробка програмної системи	26.05.18-02.06.18	
4	Тестування програмної системи	03.06.18-06.06.18	
5	Розробка розділу охорона праці	07.06.18-09.06.18	
6	Оформлення електронних плакатів	10.06.18-12.06.18	
7	Оформлення пояснювальної записки	13.06.18-15.06.18	

Здобувач вищої освіти _____ Газашвілі Д.В.
(підпис) (прізвище та ініціали)

Керівник _____ Барбарук В.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту (роботи) бакалавра: 99 с., 69 рис., 5 табл., 26 бібліографічних джерел посилань, 1 додаток.

Об'єкт розробки: процеси обробки замовлення на купівлю обладнання від абонентів системи.

Мета роботи: розробка сервісу для системи обліку та тарифікації наданих послуг.

В проекті виконано:

1. Огляд методів, підходів, технологій до проектування та розробки інформаційних систем, сформульована постановка задачі.
2. Проектування та розробку бази даних.
3. Вибір засобів розробки програмного забезпечення.
4. Проектування та розробку програмної системи.
5. Тестування роботи системи.
6. Здійснений аналіз потенційних небезпечних і шкідливих виробничих чинників проектованого об'єкта, що впливають на персонал.

Отримано наступні результати: розроблена система дозволяє обробляти замовлення на купівлю обладнання від абонентів системи. Оператор сервісу має можливість контролювати життєвий цикл заявки і формувати необхідні статистичні статуси, відповідно через різні рівні взаємодії з клієнтами та перевізниками; формувати розклад перевезень кур'єрів після узгодження часу доставки з клієнтом. У майбутньому розвитку сервісу є можливість реалізувати зв'язок зі сторонніми системами перевізників через FTP сервер для автоматизації процесу доставки товарів.

Практичне значення, галузь застосування роботи: розроблений сервіс дає можливість розширити та поліпшити якість послуг біллінгової системи.

Ключові слова: БІЛЛІНГ, БАЗА ДАНИХ, DDD, JAVA, SPRING, ORACLE, MYBATIS, TDD.

Умови одержання дипломного проекту: СНУ ім. В. Даля, пр. Центральний 59-А, м. Сєвєродонецьк, 93400.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ…….5
ВСТУП 6
1 ОГЛЯД МЕТОДІВ, ПІДХОДІВ, ТЕХНОЛОГІЙ ДО ПРОЕКТУВАННЯ ТА РОЗРОБКИ ІНФОРМАЦІЙНИХ СИСТЕМ 8
1.1 Порівняльний аналіз підходів до проектування інформаційних систем 8
1.2 Огляд технологій 12
1.3 Підходи до управління проектами щодо створення ПЗ 29
1.4 Постановка задачі дипломної роботи 39
2 ПРОЕКТУВАННЯ ТА РОЗРОБЛЕННЯ СЕРВІСУ 41
2.1 Огляд піраміди Domain Driven Design 41
2.1 Використання шаблону проектування Test-driven development 46
2.2 Розробка моделі БД 49
3 ТЕСТУВАННЯ ТА ІЛЮСТРАЦІЯ РОБОТИ СИСТЕМИ 56
3.1 Дизайн сервісу 56
3.2 Ілюстрація роботи головної сторінки 56
3.3 Ілюстрація роботи сторінки – відображення замовлення 58
3.4 Ілюстрація роботи сторінки «Передача замовлень кур’єру» 62
3.5 Тестування роботи системи 63
4 ОХОРОНА ПРАЦІ…….....67
4.1 Загальні питання з охорони праці 67
4.2 Аналіз стану умов праці 68
4.3 Виробнича санітарія 70
4.4 Гігієнічні вимоги до параметрів виробничого середовища 73
ВИСНОВКИ 80
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 82
Додаток А Комп’ютерна презентація 85

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ЖЦ – життєвий цикл

ІС – інформаційна система

ООП – об’єктно-орієнтоване програмування

ПЗ – програмне забезпечення

СУБД – система управління базами даних

JAVA – об’єктно-орієнтована, кросплатформена мова програмування

PL/SQL – (Procedure Language + Structured Query Language) мова програмування, яка використовується для доступу до баз даних Oracle

SPRING MVC – (Spring Model View Controler) архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення

MYBATIS – фреймворк, що автоматизує взаємодію між базами даних і об'єктами Java

DDD – підхід до проектування Domain Driven Design

TDD – шаблон програмування Test-driven development

ВСТУП

Незважаючи на більш ніж чвертьвікове існування "програмної інженерії" як окремої галузі наукового знання, багато компаній, що займаються розробкою програмного забезпечення, розглядають процес розробки ПЗ складних вбудованих систем як непередбачуваний і некерований. Втрата керованості робить суспільство все більш залежним від якості програмних систем, вимагає значних витрат на їх супровід, які в 2-3 рази перевищують витрати на розробку. Однією з основних причин, що лежить в основі цих проблем є зростаюча складність систем і, зокрема, їх динамічна поведінка. Традиційні підходи до проектування залишають досить велику частину робіт з уточнення вимог на етап реалізації, тим самим знижуючи якість готової системи і ускладнюючи її супровід.

Для полегшення процесу проектування існують, так звані, парадигми програмування. Це сукупність ідей і понять, що визначають стиль написання комп'ютерних програм (підхід до програмування). Це спосіб концептуалізації, що визначає організацію обчислень і структурування роботи, що виконується комп'ютером.

На сьогодняшній день набуває популярності підхід Domain-Driven Design. Це, так зване, проблемно-орієнтоване проектування, яке передбачає набір принципів і схем, що допомагають розробникам створювати витончені системи об'єктів. Це спосіб проектування, який фокусується передусім на предметній області, на об'єктах реального світу, їх поведінці і взаємодії, тобто фокусується на моделі та бізнес-логіці, а не на структурі даних. DDD робить модель явною, а роботу над цією моделлю ставить на перше місце в процесі розробки всієї системи, а так само прагне до створення таких моделей, які зрозумілі практично всім IT-фахівцям, а не тільки розробникам.

В данній дипломній роботі реалізована обробка замовлень на купівлю обладнання від абонентів системи. Оператор сервісу має можливість

контролювати життєвий цикл заявки і формувати необхідні статистичні статуси, відповідно через різні рівні взаємодії з клієнтами та перевізниками. Оператор також формує розклад перевезень кур'єрів після узгодження часу доставки з клієнтом.

Розробка сервісу була викликана необхідністю автоматизації обробки замовлень в системі VC Billing New і генерації звітів.

Даний сервіс є web-додатком. Актуальність досліджень в області питань побудови web-додатків обумовлена тим, що даний вид програмного забезпечення перспективний, як інструмент електронної комерції та надає широкі можливості соціальної взаємодії.

Web-додаток повинен відповідати наступним вимогам:

– виконуватися незалежно від операційної системи клієнта (при цьому накладається вимога кросбраузерності);

– являти собою розподілену інформаційну систему, підтримуючи велике число одночасних звернень;

– забезпечувати високу швидкість обробки інформації;

– гарантувати надійність обчислень і т.д.

1 ОГЛЯД МЕТОДІВ, ПІДХОДІВ, ТЕХНОЛОГІЙ ДО ПРОЕКТУВАННЯ ТА РОЗРОБКИ ІНФОРМАЦІЙНИХ СИСТЕМ

1.1 Порівняльний аналіз підходів до проектування інформаційних систем

Вибір методів проектування ІС визначається цілями проекту і значною мірою впливає на весь його подальший хід. Раціональний вибір можливий при розумінні кількох аспектів:

– цілей проекту;

– вимог до інформації необхідної для аналізу і прийняття рішень в рамках конкретного проекту;

– можливостей підходу з урахуванням вимог;

– особливостей розроблюваної/впроваджуваної інформаційної системи.

1.1.1 Структурний підхід

Розглянемо структурний підхід. Суть структурного підходу до розробки ІС полягає в його декомпозиції (розбитті): система розбивається на функціональні підсистеми, які в свою чергу діляться на підфункції, що підрозділяються на завдання і так далі [1]. Процес розбиття триває аж до конкретних процедур. При цьому система, що автоматизується, зберігає цілісне уявлення, в якому всі складові компоненти взаємопов'язані. При розробці системи "знизу-вверх" від окремих завдань до всієї системи цілісність втрачається, виникають проблеми при інформаційному стикуванні окремих компонентів. На початку підходу розробляється функціональна модель, за допомогою якої визначаються, аналізуються і фіксуються вимоги

до складу та структури функцій системи. Визначається, для яких цілей розробляється система, які функції вона буде виконувати [2]. На цій же моделі вказуються початкова інформація, проміжні та підсумкові результати роботи системи. На основі інформаційних потоків визначається склад і структура необхідних даних, що зберігаються в системі (будується інформаційна модель). Далі з урахуванням розроблених моделей створюються процедури реалізації функції, тобто алгоритми обробки даних і поведінки елементів системи. На заключній стадії встановлюється розподіл функцій по підсистемах (компонентам), необхідне технічне забезпечення і будується модель їх розподілу по вузлах системи.

1.1.2 Об’єктно-орієнтовані методології та Domain Driven Design

Важливе місце в аналізі системи та її проектуванні займають об'єктно-орієнтовані методології, засновані на об'єктній декомпозиції предметної області, що подається у вигляді сукупності об'єктів, які взаємодіють між собою за допомогою передачі повідомлень [1]. Даний підхід не є протиставленням структурному підходу. Більш того, фрагменти методологій структурного аналізу (базові моделі DFD, ERD, STD) використовуються при об'єктно орієнтованому аналізі для моделювання структури та поведінки самих об'єктів. У розробленій методології запропоновані комбіновані методи структурно-функціонального і об'єктно-орієнтованого аналізу. Залежно від конкретних вимог до проектування пропонуються найбільш оптимальні методи створення ІС. Вибір методів представлений у вигляді шаблонів (готових рішень). Використання шаблонів в розробленій методології є основним технологічним рішенням, технологією. Технологія побудови ІС з використання шаблонів реалізується за єдиним універсальним сценарієм і складається з наступних етапів (кроків):

– вибір стандарту ЖЦ, визначення етапів (стадій), процесів;

– вибір архітектури ІС;
– проектування моделей для створення ІС;
– оцінка якості моделей.

Відповідно до методології залежно від виду виконуваних робіт може здійснюватися перехід на один з попередніх процесів або етапів (стадій) проектування системи. З кожною ітерацією розроблені інформаційні моделі уточнюються, на їх основі будуються все більш повні версії ПЗ. Розроблена методологія проектування ІС дозволяє зробити оптимальний вибір необхідних методів та інструментальних засобів для побудови моделі ІС.

Окремо варто розглянути методологію доменно-орієнтованого проектування (Domain-Driven Design або DDD). Методологія DDD є спробою перенесення методик моделювання в програмну інженерію. На відміну від інших дисциплін аналіз предметної області та отримані на його основі моделі стають центральними складовими системи. Фактично готове програмне забезпечення, побудоване за принципами DDD, є діючою моделлю предметної області.

Ідеологія DDD заснована на припущенні, що істотні зміни в предметній області відбуваються значно рідше, ніж зміни вимог до програмного забезпечення. Таким чином, архітектура системи, заснована на моделі предметної області, буде більш стабільною. В ідеальному випадку якість системи буде визначатися коректністю моделі прикладної області та повнотою реалізації цієї моделі.

Додаток DDD дозволяє згладити розрив між технологіями, застосовуваними для зберігання даних і для реалізації логіки предметної області. Це спрощує реалізацію вимог, визначених у специфікаціях системи і дозволяє знизити вплив інженерних проблем на проектування системи.

Domain Driven Design (DDD) - предметно-орієнтоване проектування, яке передбачає набір принципів і схем, що допомагають розробникам створювати витончені системи об'єктів [3]. Це спосіб проектування, який фокусується передусім на предметній області, на реальних об'єктах, їх

поведінці і взаємодії, тобто фокусується на моделі та бізнес-логіці, а не на структурі даних. DDD робить модель явною, а роботу над цією моделлю ставить на перше місце в процесі розробки всієї системи, а так само прагне до створення таких моделей, які зрозумілі практично всім IT-фахівцям, а не тільки розробникам. Оскільки такі моделі розраховані на використання людьми без технічної освіти, зручно представляти їх різними способами. Як правило, модель предметної області може бути оформлена у вигляді UML-схеми, виражена у вигляді коду або описана на мові предметної області.

Підхід DDD особливо корисний у ситуаціях, коли розробник не є фахівцем в області продукту, що розробляється. Наприклад: програміст не може знати всі області, в яких потрібно створити ПЗ, але за допомогою правильного уявлення структури, за допомогою проблемно-орієнтованого підходу може без проблем спроектувати додаток, грунтуючись на ключових моментах і знаннях. Для розуміння і опису ключових деталей, специфічних для предметної області, необхідно використовувати несуперечливу чітку мову. У предметно-орієнтованому проектуванні - це не іменники і дієслова, а концепції та їх зв'язки. Точніше, мова описує призначення концепцій. Важливо розуміти і доносити до слухача, наскільки важлива і цінна та чи інша інформація. Важливо і те, як саме реалізується певне призначення, але, як правило, в кожному конкретному випадку можливо кілька варіантів реалізації. При роботі в предметній області всі повинні максимально регулярно і при будь-якій можливості користуватися такою мовою, щоб розуміти і пояснювати концепції та призначення. При спілкуванні на універсальній мові більш творчий характер набуває співробітництво з експертами-предметниками, користь такої взаємодії також підвищується. Необхідно відстежувати і усувати такі технічні та ділові неясності, які затемнюють зміст важливих концепцій [3]. З точки зору фахівця в предметній області такі концепції можуть бути «прихованими» або «уявними». І часто подібні ознаки характерні саме для варіантів реалізації, а не для самих концепцій предметної області. DDD не виключає опису

реалізації, але інформація про призначення моделі в цій методології цінується набагато вище.

1.2 Огляд технологій

1.2.1 Особливості мови програмування Java

Об'єктно-орієнтоване програмування - це метод програмування, в центрі уваги якого знаходяться дані (тобто об'єкти) і засоби доступу до них. За своєю суттю об'єктно-орієнтовані властивості мов Java і C++ збігаються.

Об'єктна орієнтація за минулі 30 років вже довела свою цінність і практичність. Без неї вже неможливо уявити собі сучасну мову програмування.

Основна відмінність між мовами Java і C++ полягає в механізмі множинного спадкоємства. Механізми відображення і серіалізації об'єктів, реалізовані в Java дозволяють розробнику створювати стійкі об'єкти і засоби для графічних користувацьких інтерфейсів на основі готових компонентів.

Мова Java володіє великою бібліотекою програм для передачі даних на основі протоколів TCP/IP (Transmission Control Protocol/Internet Protocol - протокол передачі гіпертексту) або FTP (File Transfer Protocol - протокол передачі файлів).

Програми, написані на мові Java, можуть відкривати об'єкти і отримувати до них доступ через мережу за допомогою URL-адрес (Uniform Resource Location - універсальний адресу ресурсу) так само просто, як і в локальній мережі [4].

Мова Java надає потужні та зручні засоби для роботи в мережі. Кожен, хто коли-небудь намагався писати програми для роботи в мережі інтернет на інших мовах програмування, здивований тим, як легко вирішуються на мові Java найважчі завдання, наприклад, відкриття мережевих з'єднань (sockets connection).

Налагоджений механізм, що складається з так званих сервлетів (servlets), дає можливість працювати з сервером дуже просто і ефективно.

Мова Java в першу чергу призначена для створення програм, які повинні надійно працювати на будь-яких платформах і під будь-яким навантаженням. Основну увагу в мові Java було приділено ранньому виявленню можливих помилок, динамічній перевірці (під час виконання програми), а також виключенню ситуацій, які можуть призвести до помилок.

Єдина значна відмінність мови Java від мови C++ полягає в моделі вказівників, прийнятій в мові Java, яка виключає можливість перезапису ділянки пам'яті і пошкодження даних.

Ця властивість дуже важлива. Компілятор мови Java виявляє такі помилки, які в інших мовах програмування виявляються тільки на етапі виконання програми. Крім того, програмісти, витративши багато часу на пошук помилки, що викликала пошкодження пам'яті через невірне значення вказівника, позбудуться подібної проблеми в Java, оскільки в цій мові програмування такі проблеми не виникають.

У таких мовах програмування, як Visual Basic і COBOL, вказівники явно не використовуються. Але для програмування на C/C++ потрібно вміти працювати з вказівниками для доступу до рядків, масивів, об'єктів і навіть файлів. Навіть багато структур даних у мові, що не має вказівників, реалізувати дуже складно. Програміст на мові Java може не турбуватися про невірне значення вказівника, неправильний розподіл або виток пам'яті.

Простота мови входить в ключові характеристики Java: розробник не повинен тривалий час вивчати мову, перш ніж він зможе на ній програмувати. Фундаментальні концепції мови Java швидко схоплюються, і програмісти з самого початку можуть вести продуктивну роботу. Розробниками Java було прийнято до уваги, що багато програмістів добре знайомі з мовою C++, тому Java, наскільки це можливо, наближена до C++.

В Java довелося відмовитися від:

- перевантаження операторів (але перевантаження методів в Java залишилися);
- множинного спадкування;
- автоматичного розширеного приведення типів.

Додалася автоматична збірка сміття, що спрощує процес програмування, але декілька ускладнює систему в цілому. В C і C++ управління пам'яттю викликало завжди масу проблем, тепер же про це не доведеться багато піклуватися.

Мова Java із самого початку проектувалася як об'єктно-орієнтована. Завданням розподілених систем клієнт-сервер відповідає об'єктно-орієнтована парадигма: використання концепцій інкапсуляції, успадкування та поліморфізму. Java надає ясну і дієву об'єктно-орієнтовану платформу розробки.

Програмісти на Java можуть використовувати стандартні бібліотеки об'єктів, що забезпечують роботу з пристроями введення/виведення, мережеві функції, методи створення графічних користувацьких інтерфейсів. Функціональність об'єктів цих бібліотек може бути розширена. Java розроблена для оперування в розподілених середовищах, це означає, що на першому плані повинні стояти питання безпеки. Засоби безпеки, вбудовані в мову, і система виконання Java дозволяють створювати додатки, на які неможливо "напасти" ззовні. У мережних середовищах додаткиё, написані на Java, захищені від вторгнення неавторизованого коду, що намагається впровадити вірус або зруйнувати файлову систему.

Java розроблена для підтримки додатків, впроваджуваних в гетерогенні мережеві середовища. У подібних середовищах, додатки повинні виконуватися на різних апаратних архітектурах, під керуванням різних операційних систем і у взаємодії з інтерфейсами різних мов програмування. Для забезпечення незалежності від платформи програм компілятор Java генерує байт-код - архітектурно-нейтральний проміжний формат програми, створюваний для ефективної передачі коду на різні апаратні і програмні

платформи. При виконанні програми байт-код інтерпретується виконуючою машиною Java. Один і той же Java байт-код буде виконуватися на будь-якій платформі. Архітектурна незалежність - лише складова частина переносимості. На відміну від C або C++ в Java не існує поняття "залежності від реалізації", коли йдеться про розмірності базових типів. Формати типів даних і операції над ними чітко визначені. Тим самим, програми залишаються незмінними на будь-якій платформі - не існує несумісності типів даних на апаратних і програмних архітектурах.

Архітектурна незалежність і переносимість програмного забезпечення Java забезпечується віртуальною машиною Java (Java Virtual Mashine - JVM) - абстрактною машиною, для якої компілятор Java генерує код. Спеціальні реалізації JVM для конкретних апаратних і програмних платформ надають уже конкретну віртуальну машину. JVM базується на стандарті інтерфейсу переносимих операційних систем (POSIX).

Продуктивність завжди заслуговує особливої уваги. Java досягає високої продуктивності завдяки спеціально оптимізованому байт-коду, який легко перекладається в машинний код. Автоматична збірка сміття виконується як фоновий потік з низьким пріоритетом, забезпечуючи високу ймовірність доступності необхідної пам'яті, що веде до збільшення продуктивності. Додатки, що вимагають великих обчислювальних ресурсів, можуть бути спроектовані так, щоб ті частини, які вимагають інтенсивних обчислень, були написані мовою асемблера і взаємодіяли з Java платформою. В основному користувачі відчувають, що додатки взаємодіють швидко, незважаючи на те, що вони інтерпретуються.

Java-інтерпретатор може виконувати Java байт-код на будь-якій машині, на якій встановлено інтерпретатор та система виконання. На платформі фаза збірки програми є простою і покроковою, тому процес розробки істотно прискорюється і спрощується, відсутні традиційні важкі етапи компіляції, збірки, тестування.

Більшості сучасних мережевих додатків зазвичай необхідно здійснювати кілька дій одночасно. У Java реалізований механізм підтримки легковагих процесів-потоків (ниток). Java надає засоби створення додатків з безліччю одночасно активних потоків.

Для ефективної роботи з потоками в Java реалізований механізм семафорів і засобів синхронізації потоків: бібліотека мови надає клас Thread, а система виконання надає засоби ёдиспетчеризації та засоби, що реалізують семафори. Важливо, що робота паралельних потоків з високорівневими системними бібліотеками Java не викличе конфліктів: функції, надані бібліотеками, доступні будь-яким потокам.

По ряду міркувань Java більш динамічна мову, ніж C++. Вона була розроблена спеціально для підстроювання під змінюване оточення. У той час як компілятор Java на етапі компіляції і статичних перевірок не допускає ніяких відхилень, процес складання та виконання суто динамічний. Класи зв'язуються тільки тоді, коли в цьому є необхідність. Нові програмні модулі можуть підключатися з будь-яких джерел, в тому числі, поставлятися по мережі. У випадку з браузером HotJava та іншими подібними додатками інтерактивний виконуваний код може бути завантажений звідки завгодно, що дозволяє виробляти прозорі модифікації додатків. В результаті можливе створення інтерактивних служб, які безболісно модифікуються, обслуговують велику кількість клієнтів і забезпечують розвиток електронного бізнесу через Internet. Якщо описані вище характеристики розглядати окремо, то їх можна знайти в багатьох програмних платформах. Радикальне нововведення полягає в способі, пропонованому Java і системою виконання, яка поєднує в собі всі характеристики для надання гнучкої і потужної системи програмування.

Розробка додатків на Java призводить до отримання програмного забезпечення, яке:

– є переносним на різні архітектури, операційні системи та графічні інтерфейси;

– безпечно;

– високопродуктивно.

Завдяки Java робота з розробки програмного забезпечення значно спрощується - всі старання спрямовані на досягнення кінцевої мети: вчасно отримати передовий продукт, який спирається на солідну основу Java.

1.2.2 Особливості використання Spring Framework

Spring Framework - це універсальний фреймворк з відкритим вихідним кодом для Java-платформи. Перша версія була написана Родом Джонсоном в 2003 році. Spring Framework є Java платформою, що забезпечує повну підтримку спільної роботи частин Java додатків. Він забезпечує вирішення багатьох завдань, з якими стикаються Java-розробники і організації, які хочуть створити інформаційну систему, засновану на платформі Java. З-за широкої функціональності важко визначити найбільш значущі структурні елементи, з яких він складається. Spring Framework має досить широку функціональність і активно використовується при розробці складних бізнес-додатків. Більшість Spring фреймворків може працювати незалежно один від одного, проте вони забезпечують більшу функціональність при спільному їх використанні [5].

Перелічимо основні переваги Spring:

– додаток не залежить від класів Spring, тобто не здійснюється успадкування від Spring і ніяк від нього не залежить, максимум можна реалізовувати який-небудь інтерфейс. Виходить, що додаток реалізує логіку, а за додатковими сервісами просто звертаємося до Spring, викликаючи його методи. Причому виклик робиться таким чином, що в основному викликаються теж інтерфейси, а реалізація у цих інтерфейсів може бути різна, тобто Spring пропонує кілька реалізацій для різних платформ,

наприклад, різна реалізація доступу до даних і т. д. Також Spring можна використовувати в маленьких додатках;

– інверсія контролю (або ін'єкція залежностей). Класи не створюють своїх залежностей, вони їх отримують (через метод setter можна отримати будь-який сервіс). Тобто один клас залежить від іншого класу. Наприклад, який-небудь сервіс виконує відправку пошти. Виходить, що сервіс залежить від сервісу відправки пошти. У IoC суть полягає в тому, що не створюємо сервіс відправки пошти (new MailService і далі виклик методів), а його отримуємо. По суті отримуємо на вхід, наприклад, через setter або як завгодно, Service і далі робимо з ним, що завгодно. Можемо отримати будь-який MailService.Spring аспектно-ориентований;

– використовується AspectJ. В Spring є своя реалізація виділення наскрізної функціональності в окремі модулі (АОП), але все ж краще використовувати AspectJ. Наприклад, функціональність, яка займається управлінням транзакціями виноситься в окремий модуль і доповнення до функціональності теж виноситься в окремий модуль. Далі визначається, які методи будуть брати участь в транзакції. Таким чином, це дозволяє локалізувати все, що пов'язано з наскрізною функціональністю в одному місці.

– Spring - це контейнер об'єктів. Це означає, що Spring зберігає в собі об'єкти, він їх створює і управляє їх життєвим циклом. По суті це означає, що ніде не викликаємо оператор new (через Spring не викликає конструктор, тобто не використовуємо new). Якщо ж знадобиться який-небудь об'єкт, то потрібно звертатися до контейнера Spring і він його видає. Як він його видає - це все можна налаштувати. Наприклад, або через new, або буде видавати Singleton, або буде видавати з пулу об'єктів. IoC працює завдяки тому, що всі об'єкти знаходяться в одному контейнері.

Фреймворк спрощує виконання і управління J2EE функціями в додатку (транзакції, повідомлення, веб). Архітектура Spring (рис. 1.1)включає такі блоки:

1. IoC – головний модуль;
2. AOP – Spring може підключитися до програми в будь-якій точці і виконати там потрібний код;
3. Service Abstraction – абстрактний завдяки тому, що викликаємо інтерфейси Spring-а і викликаємо його методи для роботи. За рахунок цього він абстрагує від усього іншого (Web remoting, EJB, JMS, Scheduling і т.д);
4. DAO – для роботи с БД;
5. ORM – JPA;
6. WEB – класи, які допомагають спростити розробку Web (авторизація, доступ до бінам Spring-а з web);
7. MVC – створює web.

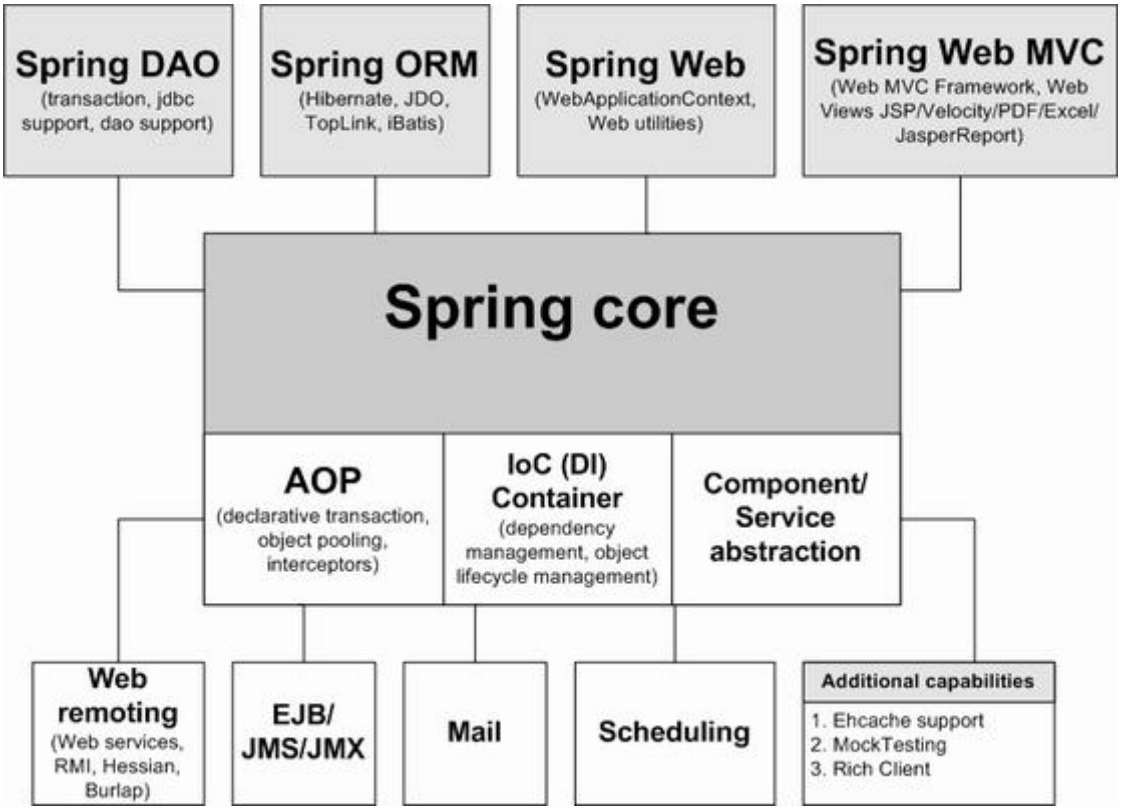


Рисунок 1.1 - Архітектура Spring

Spring включає в себе безліч класів, що реалізують найбільш затребуваний функціонал J2EE додатків. Spring істотно спрощує роботу з JDBC, ORM frameworks, Web services. Framework реалізує більш високий

рівень абстракції управління розсилкою пошти, кешуванням, виконання завдань за розкладом. Крім цього Spring включає в себе ще й Web framework.

До основних переваг Spring відносяться: простота; зручність тестування; використання Spring позитивно позначається на дизайні додатку і простоті коду.

В Spring реалізовано багато без чого не обходиться практично будь-яка Java програма. Перш ніж винаходити щось своє, варто подивитися, можливо, це вже є в Spring.

1.2.3 Характеристики фреймворку Java Server Faces

Java Server Faces (JSF) — це фреймворк для веб-додатків, написаний на Java. Він служить для того, щоб полегшувати розробку користувальницьких інтерфейсів для Java EE додатків. На відміну від інших MVC фреймворків, які управляються запитами, підхід JSF ґрунтується на використанні компонентів.

Стан компонентів для користувача інтерфейсу зберігається, коли користувач запитує нову сторінку і потім відновлюється, якщо запит повторюється.

На думку багатьох розробників найкращою архітектурою, яку можуть надати фреймворки, є Модель-Представлення-Контролер (Model-View-Controller, MVC). У додатках такого роду контролер відповідає за прийом даних від користувача і видачу відповідного представлення. Представлення формує для браузера HTML сторінку, а модель містить дані, отримані з веб-форм і ту інформацію, яку необхідно вивести на екран. Цей підхід дозволяє захистити рівень представлення даних від бізнес-логіки. Фреймворків, що підтримують MVC, існує величезна безліч, але JavaServer Faces (JSF) - є стандартом для додатків Java EE, і в цьому його перевага. По-перше, технологія активно розвивається Oracle і IBM, а по-друге, для неї на даний

момент створено безліч бібліотек, що дозволяють використовувати нестандартні UI компоненти, засновані najQuery. В JSF є три головні модулі:

– представлення - це файл * .JSF або * .XHTML, що відповідає за виведення даних у браузер і містить посилання на конкретні дані в моделі.

– модель - JavaBean, який зберігає ту чи іншу інформацію в приватних полях і надає для них функції по запису та отримання значень (сеттери/геттери) поряд з методами обробки такої інформації.

– контролер - це і є внутрішній механізм JSF, що дозволяє провести зв'язування першого з другим. JSF-додаток обычно містить два типи компонентів, причому обидва цілком прості в використанні.

Сторінки JSF формуються з XML тегів. Кожен тег представляє конкретний UI-компонент. Веб-розробнику не потрібно вдаватися в написання HTML розмітки або вставок на JavaScript, так як вони повністю генеруються компонентними тегами JSF. Кожен компонент по-суті незалежний і містить певну поведінку. Динамічні дані на JSF сторінках моделюються за допомогою спеціальних POJO, званих керованими компонентами (JSF Backing Beans). POJO - це простий Java-об'єкт, що не успадкований від якогось специфічного об'єкта і не реалізує жодних службових інтерфейсів понад ті, які потрібні для бізнес-моделі. Компонентні моделі UI і POJO дозволяють JSF заручиться підтримкою різних середовищ розробки. Фактично, багато IDE для Java підтримують інтерактивні drag-and-drop побудовники UI-інтерфейсу для JSF ("графічне програмування"). Компонентна модель JSF також дозволяє розробляти бібліотеки компонентів, значно розширює функціональність фреймворку. Серед таких розробок величезною популярністю користуються PrimeFaces, IceFaces і деякі інші проекти. Що ж стосовно недоліків, недоробок, багів і інших неприємностей - їх не надто багато. З розвитком платформи Java EE багато з них були усунені або усуваються. Програмування JSF стало набагато простіше з приходом техніки анотацій, що дозволила відмовитися від складного конфігурування компонентів за допомогою XML.

За угодою ім'я керованого bean-компонента збігається з ім'ям класу, за винятком того, що перша буква наводиться до нижнього регістру. Ім'я керованого bean-компонента також можна вказати явно за допомогою атрибуту name анотації ManagedBean, наприклад:ManagedBean (name = "home"). Для керованих bean-компонентів можливе використання атрибуту eager, якщо атрибут eager має значення true, то JSF створює цей керований bean-компонент при старті і поміщає його в область видимості додатку. Також за допомогою анотації @ ManagedProperty можно задати властивості керованого bean-компонента. У таблиці 1.2 наведено повний список наявних в JSF 2 анотацій керованих bean-компонентів:

Таблиця 1.2 – Список анотацій керованих bean-компонентів

Анотація керованого bean-компонента	Опис	Атрибути
@ManagedBean	Реєструє екземпляр цього класу в якості керованого bean-компонента і поміщає його в область видимості, зазначену однією з анотацій @ ... Scoped. Якщо область видимості не вказана, JSF поміщає bean-компонент в область видимості запиту, а якщо не вказано ім'я, JSF генерує ім'я, конвертуючи першу літеру імені класу в нижній регістр; наприклад, для класу з ім'ям UserBean JSF створює керований bean-компонент з ім'ям userBean. Атрибути eager і name є необов'язковими. Цю анотацію можна використовувати тільки для Java-класів, що реалізують конструктор без аргументів.	eager, name

Продовження таблиці 1.2

Анотація керованого bean-компонента	Опис	Атрибути
@ManagedProperty	Задає властивості керованого bean-компонента. Цю анотацію необхідно поміщати перед декларацією змінних-членів класу. Атрибут name вказує ім'я властивості, за замовчуванням воно збігається з ім'ям змінної-члена. Атрибут value є значенням властивості, їм може бути або рядок, або JSF-вираз, таке як #{...}.	value, name
@ApplicationScoped	Поміщає керований bean-компонент в область видимості додатку.	
@SessionScoped	Поміщає керований bean-компонент в область видимості сеансу.	
@RequestScoped	Поміщає керований bean-компонент в область видимості запиту.	
@ViewScoped	Поміщає керований bean-компонент в область видимості представлення.	
@NoneScoped	Вказує, що керований bean-компонент не має області видимості. Керовані bean-компоненти без області видимості корисні, коли на них посилаються інші bean-компоненти.	
@CustomScoped	Поміщає керований bean-компонент в спеціальну область видимості. Спеціальна область видимості являє собою карту, доступну авторам сторінки. Спеціальні області видимості дозволяють програмно задавати видимість і час життя bean-компонентів, які знаходяться в ній. Атрибут value вказує на цю карту.	

1.2.4 Характеристики Object-relational mapping

ORM або Object-relational mapping (об'єктно-реляційне відображення) - це технологія програмування, яка дозволяє перетворювати несумісні типи моделей в ООП, зокрема, між сховищем даних та об'єктами програмування (рис.1.2).

Рисунок 1.2 – Схема роботи ORM

ORM використовується для спрощення процесу збереження об'єктів в реляційну базу даних та їх вилучення. При цьому ORM сама піклується про перетворення даних між двома несумісними станами. Більшість ORM-інструментів значною мірою покладаються на метадані бази даних і об'єктів, так що об'єктам нічого не потрібно знати про структуру бази даних, а базі даних - нічого про те, як дані організовані у додатку. ORM забезпечує повне розділення завдань в добре спроектованих додатках, при якому і база даних, і додаток можуть працювати з даними кожен у своїй вихідній формі. Використання ORM вирішує проблему так званої парадигми «невідповідності», яка свідчить про те, що об'єктні та реляційні моделі не дуже добре працюють разом. Реляційні бази представляють дані в табличному форматі, в той час як об'єктно-орієнтовані мови представляють їх як зв'язаний граф об'єктів. Основні проблеми та невідповідності виникають під час збереження цього графа об'єктів в реляційну базу або його завантаження:

– реляційна модель може бути набагато детальніше, ніж об'єктна, тобто для зберігання одного об'єкта в реляційній базі даних використовується декілька таблиць;

– реляційні СУБД не мають нічого схожого на спадкування - природну парадигму об'єктно-орієнтованих мов програмування;

– в СУБД визначений тільки один параметр для порівняння записів - первинний ключ. У той час як ООП надає як перевірку ідентичності об'єктів (a == b), так і їх рівності (a.equals (b));

– для зв'язку об'єктів СУБД використовує поняття зовнішніх ключів, в об'єктно-орієнтованих мовах зв'язок між об'єктами може бути тільки односпрямованим [6]. Якщо ж потрібно організувати двонаправлені відносини, то доведеться визначити дві односпрямовані асоціації. Крім того, немає можливості визначити кратність відносини, дивлячись на модель предметної області;

– принцип доступу до даних в ООП кардинально відрізняється від доступу до даних в БД. Для доступу до даних в ООП використовуються послідовні переходи від батьківського об'єкта до властивостей дочірніх елементів і ініціалізації об'єктів за необхідності. Такий підхід вважається не ефективним способом отримання даних з реляційних баз даних. Як правило, кількість запитів до БД має бути зведено до мінімуму, необхідні сутності повинні по можливості завантажуватися відразу з використанням JOIN-ів.

Ключовою особливістю ORM є відображення, яке використовується для прив'язки об'єкта до його даних в БД. ORM як би створює «віртуальну» схему бази даних у пам'яті і дозволяє маніпулювати даними вже на рівні об'єктів. Відображення показує як об'єкт і його властивості пов'язані з однією або декількома таблицями і їх полями в базі даних. ORM використовує інформацію цього відображення для управління процесом перетворення даних між базою і формами об'єктів, а також для створення SQL-запитів для вставки, оновлення та видалення даних у відповідь на зміни, які додаток вносить в ці об'єкти.

Використання ORM в проекті позбавляє розробника від необхідності роботи з SQL і написання великої кількості коду, часто одноманітного і схильного до помилок. Весь генерований ORM код імовірно добре перевірений і оптимізований, тому не потрібно в цілому замислюваться про його тестування. Це безсумнівно є плюсом, але в той же час не варто забувати і про мінуси. Основний з них - це втрата продуктивності. Це відбувається тому, що більшість ORM призначені для обробки широкого спектру сценаріїв використання даних, набагато більшого, ніж будь-який окремий додаток коли-небудь зможе використовувати. Робота з БД за допомогою грамотно написаного SQL-коду буде набагато ефективніше, але не варто забувати і про такий параметр, як час - те, що з легкістю пишеться з використанням ORM за тиждень, можна реалізовувати місяць власними зусиллями. Крім того, більшість сучасних ORM дозволяють програмісту при необхідності самому задавати код SQL-запитів. Без сумнівів, для невеликих проектів використання ORM буде куди більш виправдане, ніж розробка власних бібліотек для роботи з БД.

Hibernate - бібліотека для мови програмування Java, призначена для вирішення завдань об'єктно-реляційного відображення (object-relational mapping - ORM). Дана бібліотека надає легкий у використанні каркас (фреймворк) для відображення об'єктно-орієнтованої моделі даних в традиційні реляційні бази даних. Hibernate значно зменшує час розробки додатків, що працюють з базами даних, піклується про зв'язок Java класів з таблицями бази даних (і типів даних Java в типи даних SQL), надає засоби для автоматичної побудови запитів і отримання даних. Метою Hibernate є звільнення розробника від значного обсягу порівняно низькорівневого програмування щодо забезпечення зберігання об'єктів у реляційній базі даних. Розробник може використовувати Hibernate як у процесі проектування системи класів і таблиць «з нуля», так і для роботи з вже існуючою базою даних.

Hibernate не тільки вирішує завдання зв'язку класів Java з таблицями бази даних (і типів даних Java з типами даних SQL), але й також надає засоби для автоматичної генерації та поновлення набору таблиць, побудови запитів і обробки отриманих даних і може значно зменшити час розробки, який зазвичай витрачається на ручне написання SQL- і JDBC-коду. Hibernate автоматизує генерацію SQL-запитів і звільняє розробника від ручної обробки результуючого набору даних і перетворення об'єктів, максимально полегшуючи перенесення додатку на будь-які бази даних SQL.

Hibernate забезпечує прозору підтримку збереження даних (persistence) для «POJO» (тобто для стандартних Java-об'єктів); єдина сувора вимога для класу, що зберігається - наявність конструктора за замовчуванням (без параметрів). Для коректної поведінки в деяких додатках потрібно також приділити увагу методам equals () і hashCode ().

Mapping (зіставлення, проектування) Java-класів з таблицями БД здійснюється за допомогою конфігураційних XML-файлів або Java-анотацій. При використанні файлу XML Hibernate може генерувати скелет вихідного коду для класів тривалого зберігання (persistent). У цьому немає необхідності, якщо використовується анотація. Hibernate може використовувати файл XML або анотації для підтримки схеми бази даних.

Забезпечуються можливості по організації відносини між класами «один-до-багатьох» і «багато-до-багатьох. Hibernate також може управляти рефлексивними відносинами, де об'єкт має зв'язок «один-до-багатьох» з іншими екземплярами свого власного типу даних.

Hibernate підтримує відображення для користувача типів значень. Це робить можливими такі сценарії:

– перевизначення типу за замовчуванням SQL, Hibernate вибирає при відображенні стовпця властивості;

– проектування перераховується типу Java на поле БД ніби вони є звичайними властивостями;

– проектування однієї властивості в кілька колонок.

Колекції об'єктів даних, як правило, зберігаються у вигляді колекцій Java-об'єктів, таких, як набір (Set) і список (List). Підтримуються узагальнені класи (Generics), введені в Java 5. Hibernate може бути налаштований на «лініві» (відкладені) завантаження колекцій. Відкладені завантаження є варіантом за умовчанням, починаючи з Hibernate 3.

Пов'язані об'єкти можуть бути налаштовані на каскадні операції. Hibernate забезпечує використання SQL-подібної мови Hibernate Query Language (HQL), яка дозволяє виконувати SQL-подібні запити, записані поряд з об'єктами даних Hibernate.

У Java співтоваристві Hibernate framework де-факто вважається стандартом для зручної роботи з базою даних. Розробнику важко вибрати інший фреймфорк, бо деколи він не знає про існування альтернатив. MyBatis альтернативою JPA.

Основна відмінність MyBatis від Hibernate - це те як проводиться мапінг об'єктів. Hibernate мапіт таблиці БД на сутності, даючи доступ до даних. Для отримання даних Hibernate генерує SQL запити, а запити, що генеруються з'їдають багато часу, стають громіздкими і не керованими. MyBatis мапітся не так на таблиці, а на SQL запити. За формування запитів відповідає розробник і тільки від нього буде залежати як швидко буде працювати додаток.

Ідея така - описується перетворення або маппінг таблиць бази даних або запитів в атрибути Java класів за допомогою XML файлу і, роблячи виклики відповідних методів MyBatis, просто заповнює даними ці Java класи.

1.2.5 Apache FOP

FOP – це безкоштовний Java додаток, поширюваний через Internet організацією Apache Software. FOP – це Java-додаток, який прочитує дерево об'єктів форматування, після чого перетворить його в документ PDF. Його

можна представляти і в декількох інших форматах, включаючи текстовий формат, MIF, PCL, AWT, а також виведення безпосередньо на принтер. FOP приймає дерева FO у вигляді документів XML - FO, які раніше створювалися за допомогою текстового редактора або засобу XSLT. FOP підтримує і безпосередню передачу даних від синтаксичних аналізаторів SAX і DOM без попереднього збереження у вигляді файлу FO.

1.3 Підходи до управління проектами щодо створення ПЗ

Управління проектами - відповідно до визначення національним стандартом ANSI PMBoK (Project Management Body of Knowledge - звід знань з управління проектами) - це область діяльності, в ході якої визначаються і досягаються чіткі цілі проекту при балансуванні між обсягом робіт, ресурсами (такими як гроші, праця, матеріали, енергія, простір та ін.), часом, якістю та ризиками. Ключовим фактором успіху проектного управління є наявність чіткого заздалегідь визначеного плану, мінімізації ризиків і відхилень від плану, ефективного управління змінами (на відміну від процесного, функціонального управління, управління рівнем послуг).

Продуктами проекту можуть бути продукція підприємства або організації (результати наукових і маркетингових досліджень, проектно-конструкторська і технологічна документація на новий виріб, розроблені для замовника) і рішення різних внутрішніх виробничих завдань (наприклад, підвищення якості продукції та ефективності організації праці, оптимізація фінансових потоків).

Управління проектами є частиною системи менеджменту підприємства.

Щоб проектне управління, як інструмент підвищення ефективності бізнесу, приносило максимальну результативність, кожна організація повинна вибрати для себе ту методологію управління проектами, яка найкращим чином підходить під специфіку її бізнесу.

Для того щоб не помилитися з вибором методології, необхідно розуміти плюси і мінуси кожної з них. Розглянемо найбільш популярні методологій управління проектами.

PMI PMBOK. Згідно з дослідженням близько 41% компаній у світі використовують дану методологію в якості основної (рис. 1.3).

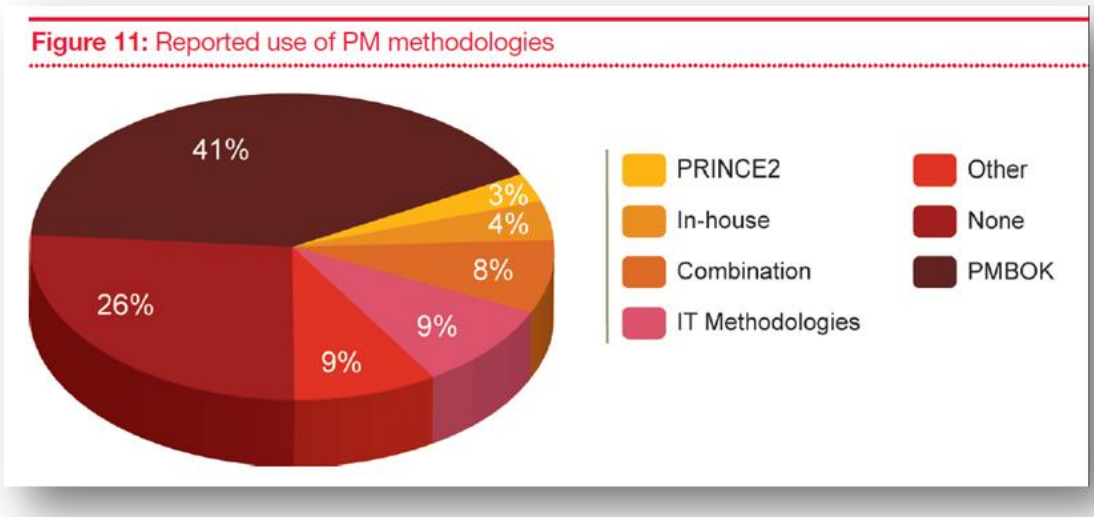


Рисунок 1.3 – Показники досліджень PMI PMBOK

Отже, стандарт PMI PMBOK5 є найбільш затребуваною методологією управління проектами зараз.

Основна унікальність даного стандарту полягає в тому, що в його основу покладено процесний підхід. Весь алгоритм управління будь-яким проектом розділений на 5 груп процесів (Ініціація, Планування, Виконання, Моніторинг та Контроль і Завершення), які в свою чергу розділені на 47 процесів.

Таким чином, менеджер проекту може в залежності від складності проекту конфігурувати свій процес, складаючи його як пазл з цих 47 процесів.

Звичайно, дана методологія розроблена під великі проекти. Вона розрахована на проекти від 3 років тривалості і від 1000 чоловік проектної

команди. Однак, даний стандарт дуже легко адаптувати під специфіку конкретного проекту та організації.

Мінусом даного стандарту є його громіздкість і великі витрати на планування та розробку проектної документації.

IPMA. Ядром методології IPMA є модель IPMA Delta, яка складається з трьох блоків (рис. 1.4):

– організація;

– індивідуальні компетенції;

– проекти.

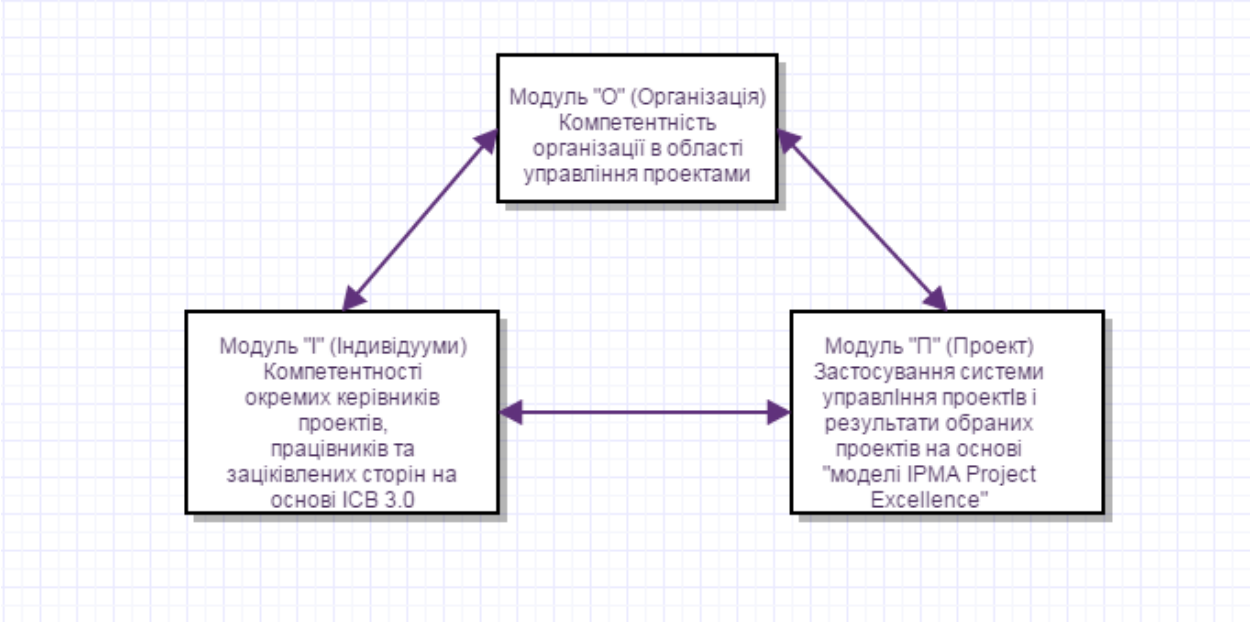


Рисунок 1.4 – Огляд методології IPMA

Дана методологія включає в себе 5 рівнів зрілості організацій в галузі управління проектами:

– початковий: досягнення в галузі управління проектами на рівні окремих співробітників. Деякі співробітники працюють задовільно, добре і навіть відмінно, але немає єдиного стандарту для всієї організації - управління портфелями проектів та програмами в цілому незадовільно. Організація не має формальних стандартів і процесів у цій галузі;

– певний: існують певні стандарти управління проектами, програмами та портфелями проектів (ППП). Структури і процеси управління проектами використовуються епізодично на окремих проектах;

– стандартизований: існують процеси, структури та стандарти управління ППП, які в основному застосовуються в організації (немає повного охоплення та інтеграції);

– керований: існують стандарти, структури та процеси управління ППП, які застосовуються у всій організації і контролюються з боку керівництва (повне охоплення та інтеграція);

– оптимизований: існують всі необхідні стандарти, структури та процеси управління ППП, які застосовуються у всій організації, контролюються керівництвом і постійно удосконалюються.

Agile. Дане сімейство методологій гнучкої розробки проектів використовує близько 9% організацій.

Самою відомою і популярною методологією даного сімейства зараз є SCRUM (рис. 1.5).

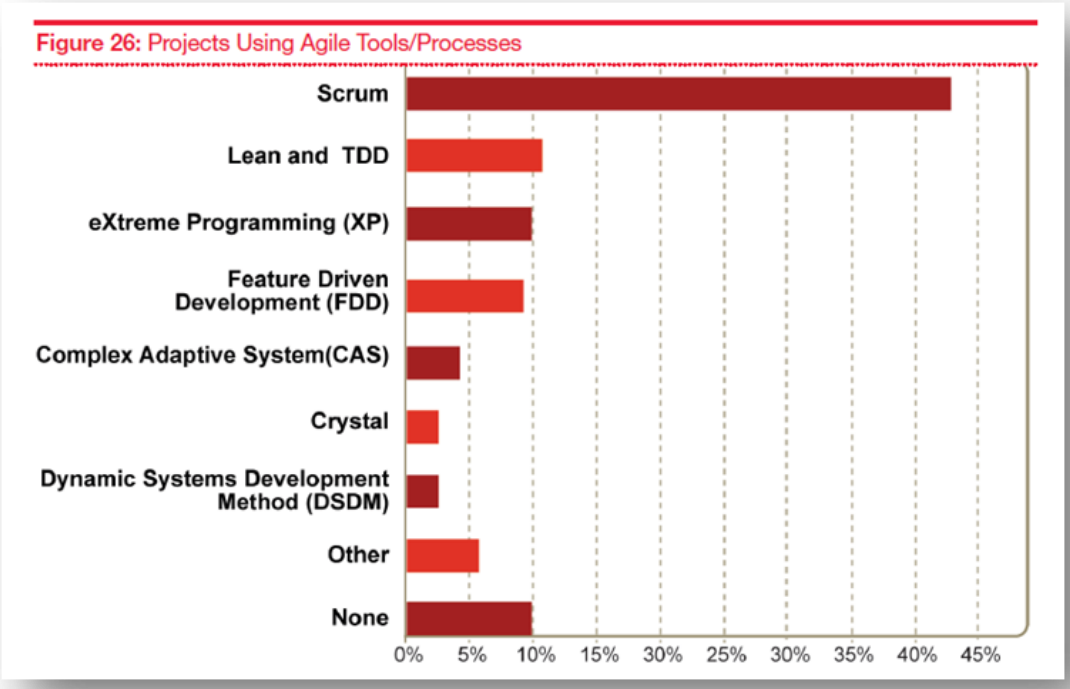


Рисунок 1.5 – Огляд відомих методологій

Методологія Agile була розроблена для управління IT-проектами в області розробки програмного забезпечення і базується на принципах протилежних класичному waterfall-підходу, який проповідує PMBOK. Порівняння методологій зображено на рисунку 1.6.

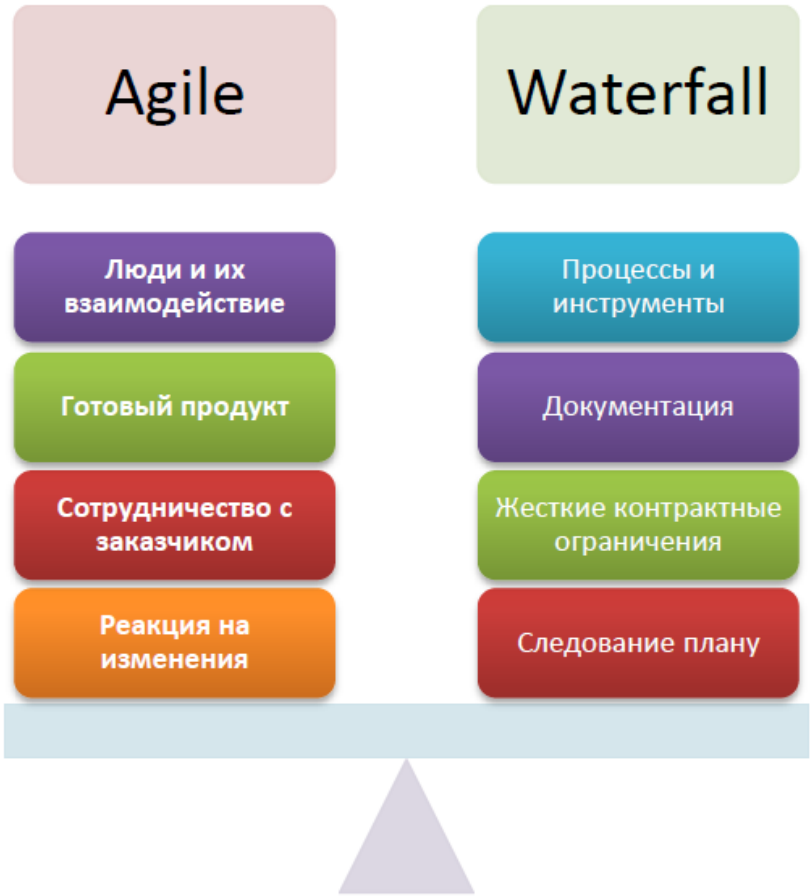


Рисунок 1.6 – Порівняння методологій Agile та Waterfall

Основні принципи методології Agile:

- короткий ітеративний цикл розробки становить 1-6 тижнів;
- фокус на продукті, а не проектный документації;
- фокус на комунікаціях в команді;
- повне технічне завдання не пишеться на весь проект, а тільки на майбутній спринт (реліз);
- гнучкий процес внесення змін в проект;

– у проектній команді повинен бути присутнім так званий product owner, який визначає вимоги продукту;

– команда розробників повинна розташовуватися в одній кімнаті разом з власником продукту;

– у команді повинен бути присутнім так званий SCRUM-майстер, який відрізняється за функціоналом від традиційного менеджера проекту.

Звичайно ж, крім явних плюсів в методології Agile присутні і явні мінуси, такі як:

– відсутність докладної документації;

– непередбачувані терміни і бюджет розробки;

– значні ризики проекту.

Кожна організація повинна визначити для себе, чи підходить для неї гнучка методологія управління проектами чи все-таки доцільно використовувати традиційний «водоспад» (PMBOK).

У найближчі роки тренд в сторону переходу на Agile-методологію збережеться і тим організаціям, які хочуть бути ефективними в галузі управління проектами, варто вже зараз звернути увагу на даний підхід до проектного управління [7].

MSF. Microsoft Solutions Framework (MSF) - це методологія розробки програмного забезпечення, створена корпорацією Microsoft в 1994 році на основі свого багаторічного досвіду роботи в IT-індустрії.

По суті, дана методологія входить в сімейство Agile-методів гнучкої розробки програмного забезпечення.

Дана методологія складається з двох взаємопов'язаних framework'ов: Microsoft Solutions Framework (MSF) і Microsoft Operations Framework (MOF).

Дана модель складається з 5 процесів управління проектами:

– вироблення концепції (Envisioning);

– планування (Planning);

– розробка (Developing);

– стабілізація (Stabilizing);

– впровадження (Deploying).

RUP. Методологія Rational Unified Process (RUP) була розроблена корпорацією IBM і призначена для управління проектами в області розробки програмного забезпечення. Стадії розробки данної методології зображено на рисунку 1.7.

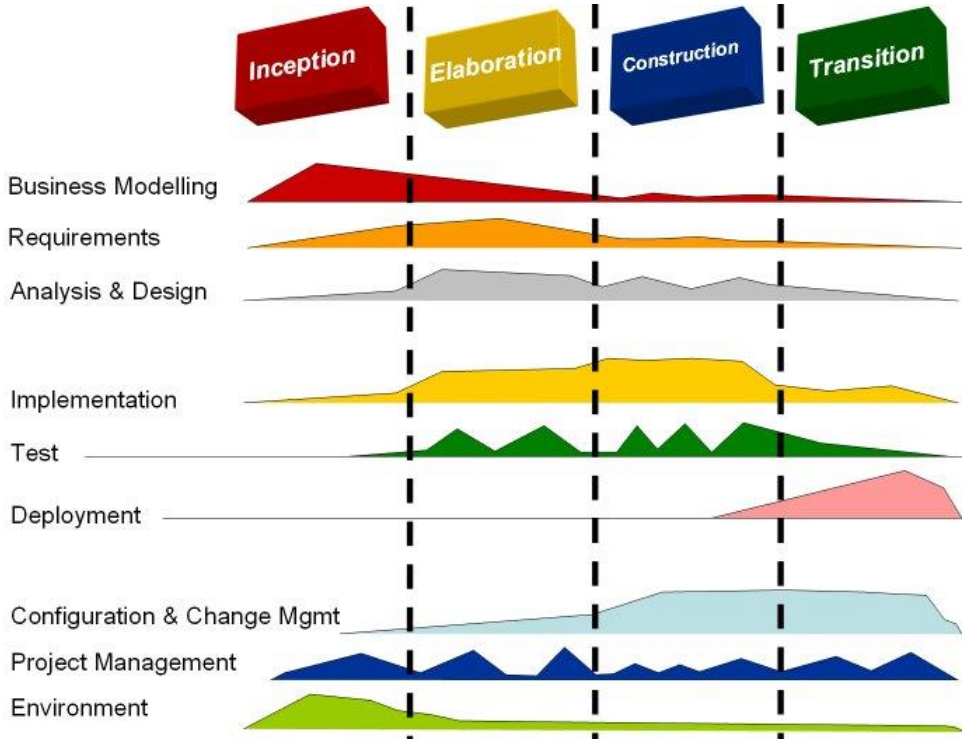


Рисунок 1.7 – Стадії розробки метології RUP.

Серед менеджерів IT-проектів існує думка, що RUP є «важким» і застарілим інструментом для розробки програмного забезпечення.

Складно однозначно погодитися або спростувати дану думку. Тільки використовуючи цю методологію на реальних проектах можливо зрозуміти наскільки вона ефективна для компанії.

Ad-hoc. Звичайно не потрібно забувати про те, що кожна компанія має право розробити власну методологію проектного управління, як і вчинили близько 4% компаній у світі.

Також близько 8% компаній використовують комбінацію з вже існуючих методологій.

Тому важливо вибрати для себе ту методологію, яка підходить саме вашій компанії, або розробити свою.

При написанні дипломної роботи використовувався підхід проектування ***Agile Scrum***. Це набір принципів, на яких будується процес розробки, що дозволяє в жорстко фіксовані і невеликі за часом ітерації, звані спринт (sprints), надавати кінцевому користувачеві працююче ПЗ з новими можливостями, для яких визначений найбільший пріоритет. Можливості ПЗ до реалізації в черговому спринті визначаються на початку спринту на етапі планування і не можуть змінюватися на всьому його протязі. При цьому суворо фіксована невелика тривалість спринту надає процесу розробки передбачуваність і гнучкість.

Спринт [7] - це ітерація, в ході якої створюється функціональне зростання програмного забезпечення. Жорстко фіксована за часом. Тривалість одного спринту від 2 до 4 тижнів. Для оцінки обсягу робіт у спринті використовується попередня оцінка, яка вимірюється в story point. Попередня оцінка фіксується в беклоге проекту. Протягом спринту ніхто не має права змінювати список вимог до роботи, внесених до беклог спринту.

Беклог проекту - це список вимог до функціональності, упорядкований за їх ступенем важливості, що підлягають реалізації. Елементи цього списку називаються «побажаннями користувача» (user story) або елементами беклога (backlog items). Беклог проекту відкритий для редагування для всіх учасників скрам процесу. Беклог спринту містить функціональність, обрану власником проекту з беклога проекту. Всі функції розбиті за завданнями, кожна з яких оцінюється скрам-командою. Щодня команда оцінює обсяг роботи, який потрібно виконати для завершення спринту.

Для відображення кількості зробленої роботи і роботи, яка ще залишилася, використовується діаграма Burn down Chart (рис. 1.8), яка оновлюється щодня для того, щоб у простій формі показати зрушення в роботі над спринтом.

Планування спринту (Sprint Planning Meeting) відбувається на початку нової ітераці. Далі з беклога проекту вибираються завдання, зобов'язання щодо виконання яких за спринт приймає на себе команда. На основі вибраних задач створюється беклог спринту. Кожне завдання оцінюється в ідеальних людино-годинах. Рішення завдання не повинно займати більше 12 годин або одного дня. При необхідності завдання розбивається на підзадачі.

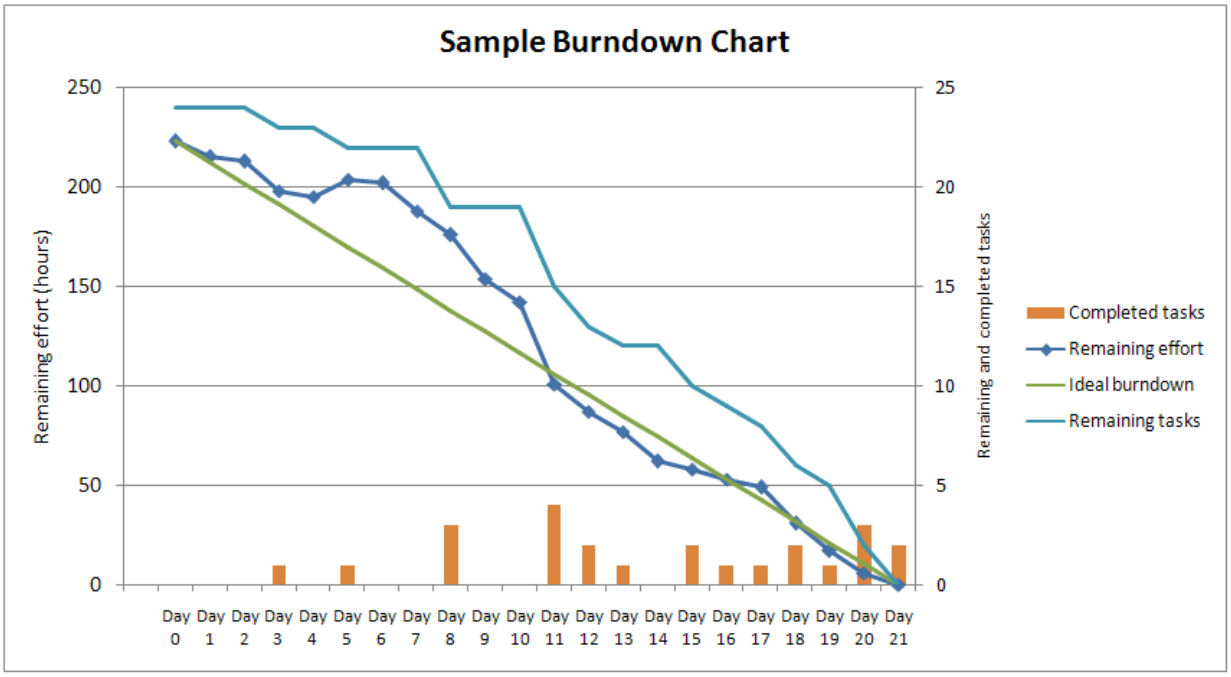


Рисунок 1.8 – Приклад діаграми Burn down Chart

Всією командою обговорюється і визначається, яким чином буде реалізований обсяг робіт.

Основними ролями в методології Scrum є:

– скрам-майстер (Scrum Master) - проводить наради (Scrum meetings), стежить за дотриманням всіх принципів скрам, вирішує протиріччя і захищає команду від відволікаючих чинників. Дана роль не передбачає нічого іншого крім коректного ведення скрам-процесу. Керівник проекту швидше відноситься до власника проекту і не повинен фігурувати як скрам-майстер;

– власник продукту (Product Owner) - представляє інтереси кінцевих користувачів та інших зацікавлених в продукті сторін;

– скрам-команда (Scrum Team) - крос-функціональна команда розробників проекту, що складається з фахівців різних профілів: тестувальників, архітекторів, аналітиків, програмістів і т. д. Розмір команди в ідеалі становить 7 ± 2 людини. Команда є єдиним повністю залученим учасником розробки і відповідає за результат як єдине ціле. Ніхто крім команди не може втручатися в процес розробки протягом спринту.

Щоденно проводиться так званий Daily Scrum meeting, який починається кожен день в один і той же час і характеризується такими пунктами:

– триває не більше 15 хвилин;

– проводиться в одному і тому ж місці протягом спринту;

– протягом наради кожен член команди відповідає на 3 питання:

1. що зроблено з моменту попередньої щоденної наради?
2. що буде зроблено з моменту поточної наради до наступної?
3. які проблеми заважають досягненню цілей спринту? (Над вирішенням цих проблем працює скрам майстер.)

Огляд підсумків спринту (Sprint review meeting) проводиться в кінці спринту за таким правилами:

– команда демонструє приріст функціональності продукту всім зацікавленим особам;

– привертається максимальна кількість глядачів;

– всі члени команди беруть участь у демонстрації (одна людина на демонстрацію або кожен показує, що зробив за спринт);

– незавершена функціональність не демонструється;

– нарада обмежена чотирма годинами залежно від тривалості ітерації і приросту функціональності продукту.

1.4 Постановка задачі дипломної роботи

Метою роботи є розроблення сервісу для біллінгової системи VC Billing New.

Система VC Billing New працює з абонентами мобільного зв’язку:

– слідкує за використанням трафіку, а також балансу абонента;

– відповідає за назначення обо відміну послуг;

– виконує зміну тарифного плану абонента та ін.

Сервіс призначений для обробки замовлень на купівлю обладнання від абонентів системи. Оператор сервісу має можливість контролювати життєвий цикл заявки і формувати необхідні статистичні звіти. Система призначена для роботи із заявками – проведення їх через різні статуси, і відповідно через різні рівні взаємодії з клієнтами та перевізниками. Оператор також формує перевезення кур’єрів після узгодження часу доставки з клієнтом. Наразі працює три точки продажу, з яких клієнт має можливість забрати замовлення самотужки.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- вивчити принципи роботи технології Spring Framework;
- вивчити особливості роботи JSF Primefaces та застосувати їх на практиці;
- провести порівняльний аналіз ORM Hibernate та MyBatis;
- проаналізувати підхід проектування DDD та застосувати його на практиці;
- вивчити основні принципи підходу проектування Scrum Agile;
- реалізувати формування та завантаження документів за допомогою технології FOP;
- використати на практиці шаблон програмування Test-driven development (TDD);
- розробити систему на основі вивчених технологій;

- провести інтеграцію сервісу в систему VC Billing New;
- реалізувати модульне тестування (Unit tests) для всіх частин програми;
- розробити тест-кейси та провести тестування основного функціоналу системи.

2 ПРОЕКТУВАННЯ ТА РОЗРОБЛЕННЯ СЕРВІСУ

2.1 Огляд піраміди Domain Driven Design

Предметною областю дипломної роботи є біллінг. Білінг — це автоматизована система обліку наданих послуг, їх тарифікації і виставляння рахунків для оплати. Функції білінгу на підприємстві групуються в три основні блоки: розрахункові операції, інформаційне обслуговування, фінансове обслуговування.

При проектуванні дипломної роботи використовувалася методологія DDD спільно із підходом керування проектами Scrum Agile. При такому синтезі технологій була розроблена піраміда, яку дуже зручно використовувати при аналізі нового функціоналу або при початковій розробці системи. На рисунку 2.1 забражена піраміда DDD, яка складається з таких блоків:

1. BOM – бізнес-об’єкты системи;
2. DTO – модель представлення таблиць бази данних;
3. Translator – клас, який здійснює трансляцію BOM в DTO і навпаки;
4. Domain Service – реалізація бізнес – методів (оперує BOM);
5. Connector – реалізація методів роботи з базою даних;
6. UI – реализація графічного інтерфейсу.

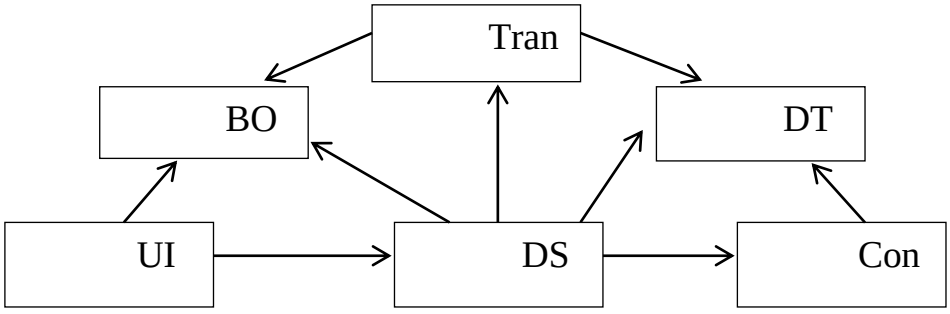


Рисунок 2.1 – Піраміда DDD

Кожній ітерації підходу Scrum відповідає сукупність кроків написання піраміди:

– створення BOM/DS – обговорюється спільно з бізнес-аналітиком;

– написання BOM, створення DS з mock-даними, створення UI - виконується розробниками, відбувається узгодження UI із замовником;

– Back End - підключення до бази даних, написання методів для роботи з БД.

Таким чином, при зміні методу зберігання даних, модифікаціям піддасться тільки частина Back End, при цьому модулі BOM/DS/UI залишаться без змін. Тобто, ці зміни ніяк не зачеплять раніше узгоджений із замовником UI та реалізовані вимоги.

Основними перевагами стилю DDD є:

– обмін інформацією. Всі учасники групи розробки можуть використовувати модель предметної області та описувані нею сутності для передачі відомостей і вимог предметної області за допомогою спільної мови предметної області, не вдаючись до технічного жаргону;

– розширюваність. Модель предметної області часто є модульною і гнучкою, що спрощує оновлення і розширення при зміні умов і вимог;

– зручність тестування. Об'єкти моделі предметної області характеризуються слабкою зв'язаністю, що полегшує їх тестування;

При такому підході використовується багатошарова архітектура. Розглянемо загальні принципи проектування з використанням багатошарової архітектури:

– абстракція. Багатошарова архітектура представляє систему як єдине ціле, забезпечуючи при цьому достатньо деталей для розуміння ролей, відповідальностей і відносин;

– інкапсуляція. Під час проектування немає необхідності робити будь-які припущення про типи даних, методи і властивості або реалізації, оскільки всі ці деталі приховані в рамках шару;

– чітко визначені функціональні шари. Поділ функціональності між шарами дуже чіткий. Верхні шари, такі як шар уявлення, посилають команди нижнім, таким як бізнес-шар і шар даних, і можуть реагувати на події, що виникають в цих шарах, забезпечуючи можливість передачі даних між шарами вгору і вниз;

– чітко визначені межі відповідальності для кожного шару і гарантоване включення в шар тільки функціональності, безпосередньо пов'язаної з його завданнями, допоможе забезпечити максимальну зв'язність в рамках шару;

– можливість повторного використання;

– відсутність залежностей між нижніми і верхніми шарами забезпечує потенційну можливість їх повторного використання в інших сценаріях.

При розробці системи була отримана діаграма бізнес-об’єктів, зображена на рисунку 2.2.

Моделювання предметних областей - ключовий розділ проектування програмного забезпечення. У моделях предметних областей розробники виражають складні функції своїх програм, реалізуючи їх потім у такому вигляді, який відповідає реальним потребам користувачів.

При розробці ПЗ на початку ітерації, згідно з метологією Scrum, була отримана дошка із завданнями. Були розписани завдання, наприклад: «Розробка функціональності друку звітів». Функціональність завдання була описана в програмному додатку Jira Confluence (рис. 2.3). Також була розроблена піраміда DDD, яка була узгодженя з Solution Architect проекту (рис. 2.4).

Jira Confluence це проста, потужна wiki система, яка дозволяє створювати сторінки і документи, обмінюватися ними та іншим контентом між учасниками проекту, а також із замовником та його командою.

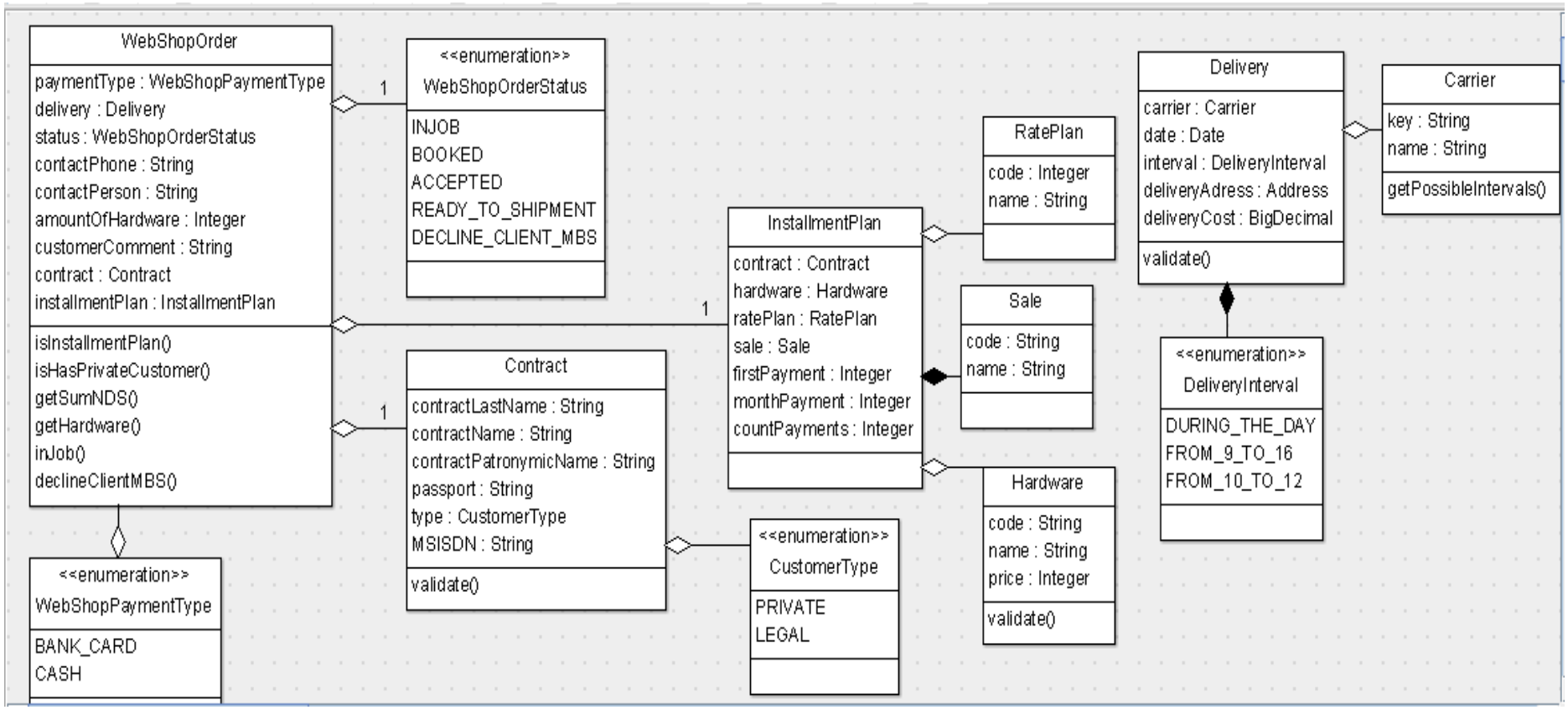
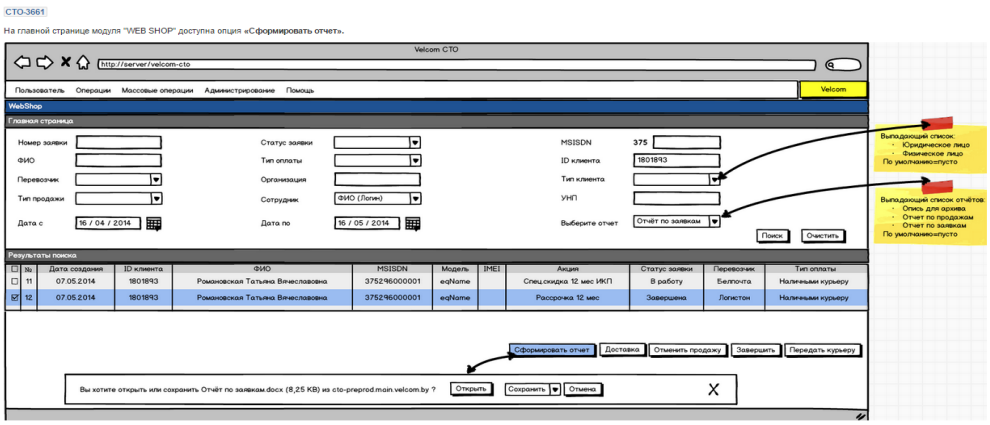


Рисунок 2.2 - BOM - діаграма

CTO-3661 Добавление функциональности формирования отчетов на главной странице



Для того чтобы воспользоваться опцией **"Сформировать отчёт"** необходимо :

1) осуществить поиск заявок по заданным критериям

2) выбрать заявку/заявки в статусе **"Завершена"** или **"Завершена УБП"**

3) выбрать отчёт из выпадающего списка

- Опись для архива
- Отчёт по продажам
- Отчёт по заявкам (шаблон документа отчет по заявкам.docx)

3) нажать на кнопку **"Сформировать отчёт"** - после чего в стандартном диалоге браузера будет возможность **Открыть** или **Сохранить** выбранный отчёт в формате **.docx**

Примечание

Поле **"Сотрудник"** - выпадающий список с опцией Autocomplete. Содержит список пользователей, которые последние меняли статус заявки в формате [ФИО пользователя и его логин]

Рисунок 2.3 – Опис завдання в Jira Confluence

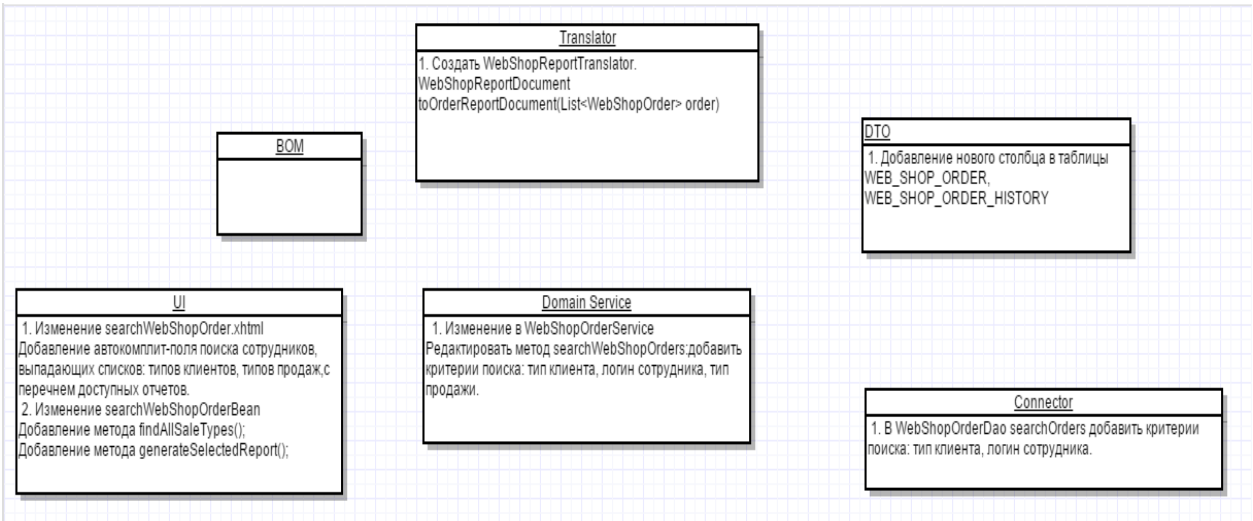


Рисунок 2.4 – Піраміда DDD для розробки завдання

2.1 Використання шаблону проектування Test-driven development

При написанні дипломної роботи виникла необхідність покриття функціоналу тестами для перевірки працездібності окремих модулів. Для цього використовувався шаблон програмування TDD. Розробка через тестування (test-driven development, TDD) – це техніка розробки програмного забезпечення, яка ґрунтується на повторенні дуже коротких циклів розробки: спочатку пишеться тест, що покриває бажану зміну, потім пишеться код, який дозволить пройти тест, і під кінець проводиться рефакторинг нового коду до відповідних стандартів (рис. 2.5).

Розробка через тестування виражається в простому правилі: спочатку тести, а потім код. Виявляється, що висловити задачу в тестах навіть простіше, ніж пояснити її колезі. Коли є тести, код писати дуже просто, завдання зводиться до того, щоб задовольнити описані вами умови.

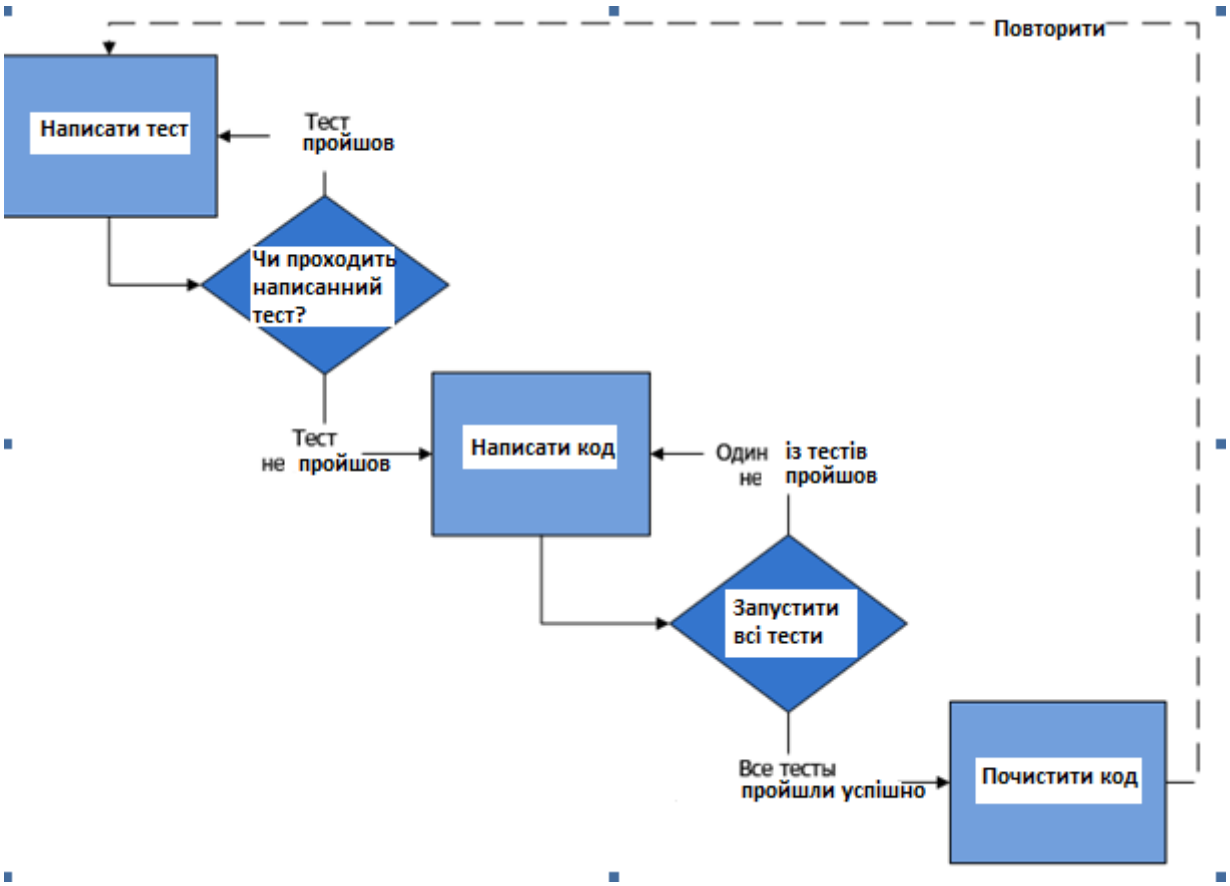


Рисунок 2.5 – Ілюстрація використання шаблону TDD

Зазвичай TDD - процес виглядає так:

– береться опис завдання з голови або паперу і починається процес роздуму над тим, з чого починати його рішення;

– описується невелика ділянка коду, яка перевіряє результат виконання ще неіснуючої функції (методу, класу etc). Якщо функція може приймати багато аргументів або мати кілька варіантів використання, то обирається лише 2-3 приклади, уникаючи докладного опису всіх варіантів;

– виконується запуск тестів. Тести – червоні;

– далі пишеться код, задовольняючи умовам написаних тестів;

– виконується запуск тестів. Тести – зелені;

– перехід до першого або до другого кроку.

З практичної точки зору, основою TDD є цикл "red/green/refactor" (рис. 2.6). У першій фазі програміст пише тест, у другій - код, необхідний для того, щоб тест працював, у третій, при необхідності, проводиться рефакторинг. Послідовність фаз дуже важлива. Відповідно до принципу "Test First" слід писати тільки такий код, який абсолютно необхідний, щоб тести виконувалися успішно.

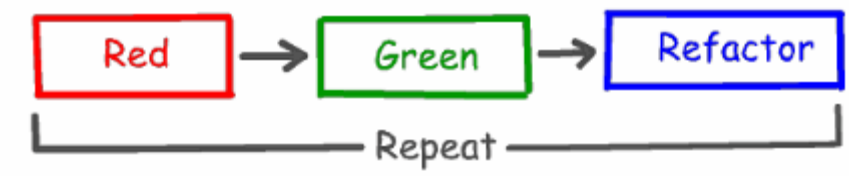


Рисунок 2.6 – Цикл виконання шаблону TDD

Значення традиційної ролі модульного тестування останнім часом в TDD-співтоваристві зазвичай не підкреслюється. Часто можна зустріти твердження, що сенс TDD взагалі не в тестуванні, а в зниженні кількості помилок. В реальних програмах ніяка кількість модульних тестів не в змозі гарантувати повну відсутність помилок. Значення тестів для рефакторинга переоцінити неможливо. Навіть самий невеликий рефакторинг, як відомо, вимагає наявності написаних тестів. Звичайно, деякі види рефакторингів,

такі, як, наприклад, перейменування полів, цілком можливо застосовувати і без тестів. Але набір модульних тестів, що покривають більшу частину додатку, дозволяє модифікувати систему значно більш агресивно і зі значно більш передбачуваними результатами. Саме поєднання рефакторинга і тестів дозволяє швидко змінювати систему.

При розробці ПЗ було дуже зручно використовувати методологію TDD, так як з її допомогою дуже легко перевірити функціональність написаного коду. Також перевагою модульного тестування є те, що один тест відповідає за один метод і не стосується інших методів, це дозволяє чітко визначити помилку та її місцезнаходження. На рисунках 2.7, 2.8 показаний приклад проведення модульного тестування в рамках поставленого завдання (рис. 2.3). Для цього потрібно було провести рефакторинг методу «searchWebShopOrders» класу WebShopServiceImpl.

```
@Override
@Transactional(value = "cto", readOnly = true)
public WebShopHolder searchWebShopOrders( String orderId, PhoneNumber msisdn,WebShopOrderStatus status,
    Carrier carrier,Date fromDate,Date toDate,String csPub,String csName, String personalNo,WebShopPaymentType type,
    Sale sale,CustomerType csType,WebShopPickupPoint pickupPoint,Date completionDate,Integer isForReport) throws Exception {

  WebShopHolder result = new WebShopHolder();
  for (WebShopOrderDto dto : findOrders(orderId,msisdn,status,carrier,fromDate,
    toDate,csPub, csName,personalNo,type,sale,csType,pickupPoint,completionDate,isForReport)) {
    WebShopOrder webShopOrder = new WebShopOrder(dto.getIdWebShopOrder().toString());
    try {
      result.addLoadedOrder(webShopOrder);
    } catch (Exception e) {
      log.error("Faild to load order with id " + webShopOrder.getId(), e);
      result.addNotLoadedOrderId(webShopOrder.getId());
    }
  }
  return result;
}
```

Рисунок 2.7 – Використання методології TDD: реалізація методу

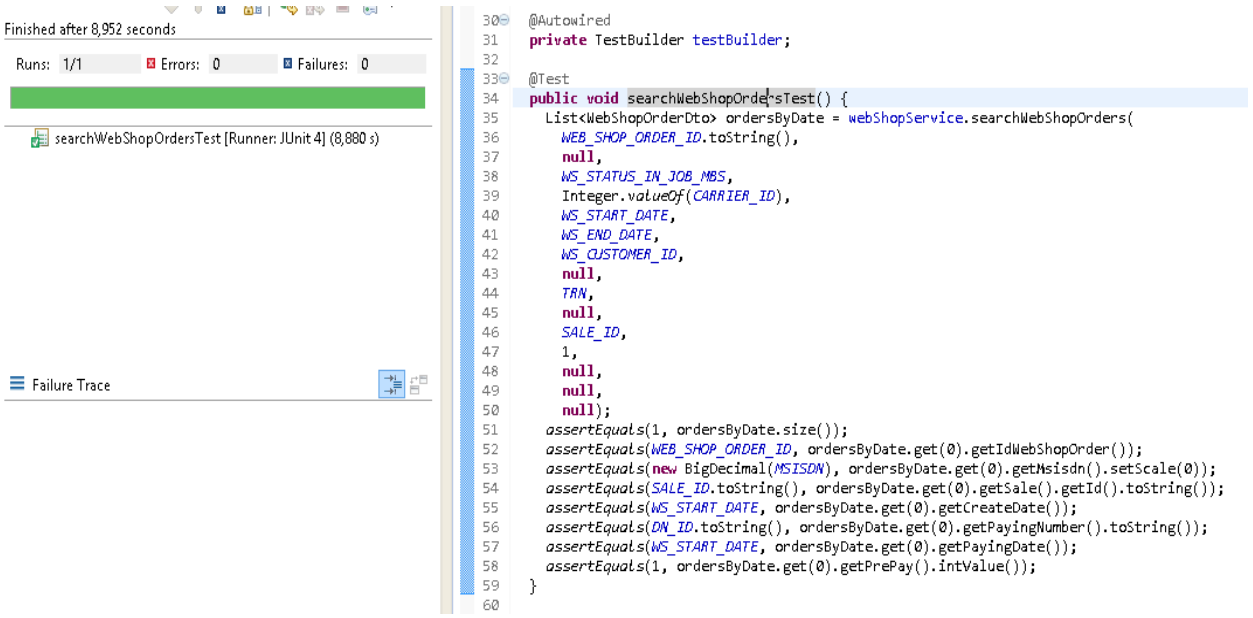


Рисунок 2.8 – Використання методології TDD: unit test та результат його виконання

2.2 Розробка моделі БД

При написанні дипломного проекту була розроблена схема бази даних для сервісу та з’інтегрована з уже існуючою базою даних проекту VC Billing New. Використовувалась СУБД Oracle 10g. На рисунку 2.9 зображена локальна схема БД сервісу, яка була змодельована у графічному редакторі Gliffy [8].

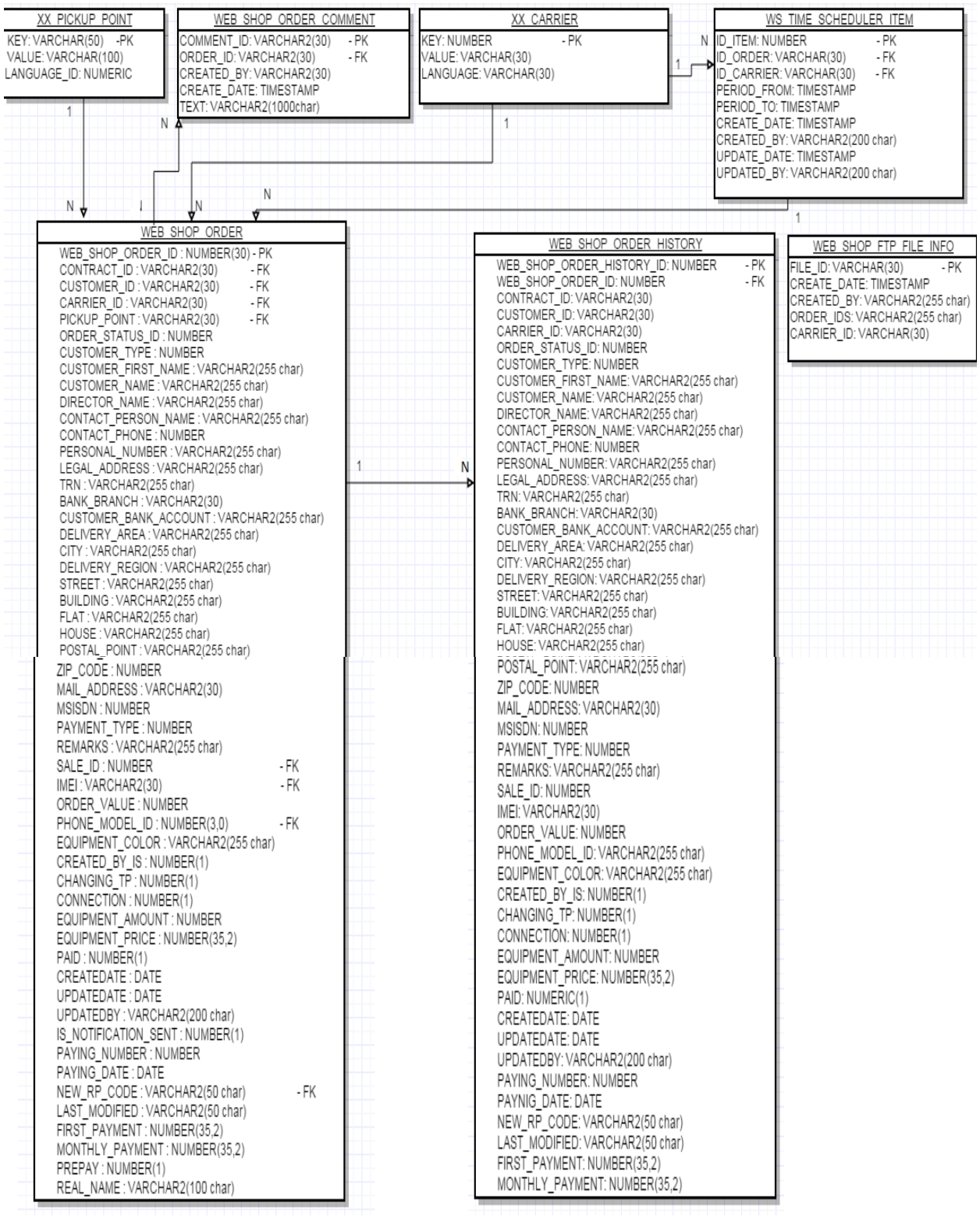


Рисунок 2.9 – Локальна схема БД

Таблиця WEB_SHOP_ORDER – описує заявку на придбання товару в Інтернет магазині (рис. 2.10).

COLUMN_NAME	DATA_TYPE	
BANK_BRANCH	VARCHAR2(30 BYTE)	№ банку
CUSTOMER_BANK_ACCOUNT	VARCHAR2(255 CHAR)	№ банк. акаунту
DELIVERY_AREA	VARCHAR2(255 CHAR)	область доставки
CITY	VARCHAR2(255 CHAR)	місто доставки
STREET	VARCHAR2(255 CHAR)	вулиця доставки
ZIP_CODE	NUMBER	індекс
MAIL_ADDRESS	VARCHAR2(50 BYTE)	email
MSISDN	NUMBER(38,0)	номер абонента
PAYMENT_TYPE	NUMBER(38,0)	тип олати
REMARKS	VARCHAR2(255 CHAR)	коментарі
SALE_ID	NUMBER(38,0)	тип продажу
IMEI	VARCHAR2(30 BYTE)	id обладнання
ORDER_VALUE	NUMBER(38,0)	сума замовлення
PHONE_MODEL_ID	VARCHAR2(255 CHAR)	модель обладнання
EQUIPMENT_COLOR	VARCHAR2(255 CHAR)	колір обладнання
CREATED_BY_IS	NUMBER(1,0)	стровення оператором
CHANGING_TP	NUMBER(1,0)	чи є зміна ТП
CONNECTION	NUMBER	чи є підключення
EQUIPMENT_AMOUNT	NUMBER(38,0)	кількість обладнання
EQUIPMENT_PRICE	NUMBER(35,2)	сума за обладнання
PAID	NUMBER(1,0)	чи оплачене замовлення

Рисунок 2.10 – Опис таблиці WEB_SHOP_ORDER

Таблиця WEB_SHOP_ORDER_HISTORY – існує для зберігання історії при зміні деякої інформації по заявці. Являється повним відображенням таблиці WEB_SHOP_ORDER.

Таблиця WEB_SHOP_FTP_FILE_INFO – містить інформацію по документам, доступним для завантаження (рис. 2.11).

COLUMN_NAME	DATA_TYPE	
FILE_ID	VARCHAR2(30 BYTE)	номер файла
CREATE_DATE	TIMESTAMP(6)	дата створення
CREATED_BY	VARCHAR2(255 CHAR)	хто створив
ORDER_IDS	VARCHAR2(255 CHAR)	№ заявок, що друкуються в файлі
CARRIER_ID	VARCHAR2(30 BYTE)	№ перевізника
FILE_UPLOAD	BLOB	контент файлу

Рисунок 2.11 – Опис таблиці WEB_SHOP_FTP_FILE_INFO

Таблиця XX_PICK_UP_POINT – містить інформацію по пунктам самовивозу товару (рис. 2.12).

COLUMN_NAME	DATA_TYPE	
VALUE	VARCHAR2(1000)	назва складу самовивозу
LANGUAGE_ID	NUMBER	мова
KEY	VARCHAR2(1000)	№ складу

Рисунок 2.12 – Опис таблиці XX_PICK_UP_POINT

Таблиця WEB_SHOP_ORDER_COMMENT – містить інформацію про коментарі, залишені оператором при обробці замовлення (рис. 2.13).

COLUMN_NAME	DATA_TYPE	
COMMENT_ID	VARCHAR2(30 BYTE)	№ запису
ORDER_ID	VARCHAR2(30 BYTE)	№ заявки
CREATED_BY	VARCHAR2(30 BYTE)	дата створення
CREATE_DATE	TIMESTAMP(6)	ким створений запис
TEXT	VARCHAR2(1000 CHAR)	текст коментару

Рисунок 2.13 – Опис таблиці WEB_SHOP_ORDER_COMMENT

Таблиця XX_CARRIER – містить інформацію про можливих перевізників для доставки замовлення (рис. 2.14).

COLUMN_NAME	DATA_TYPE	
VALUE	VARCHAR2(1000)	назва перевізника
LANGUAGE_ID	NUMBER	мова
KEY	VARCHAR2(1000)	№ перевізника

Рисунок 2.14 – Опис таблиці XX_ CARRIER

На рисунку 2.16 зображена частина схеми бази даних проекту VC Billing New.

Таблиця WS_TIME_SCHEDULER_ITEM – містить інформацію по роботі перевізників, а саме графік перевезень товарів (рис. 2.17).

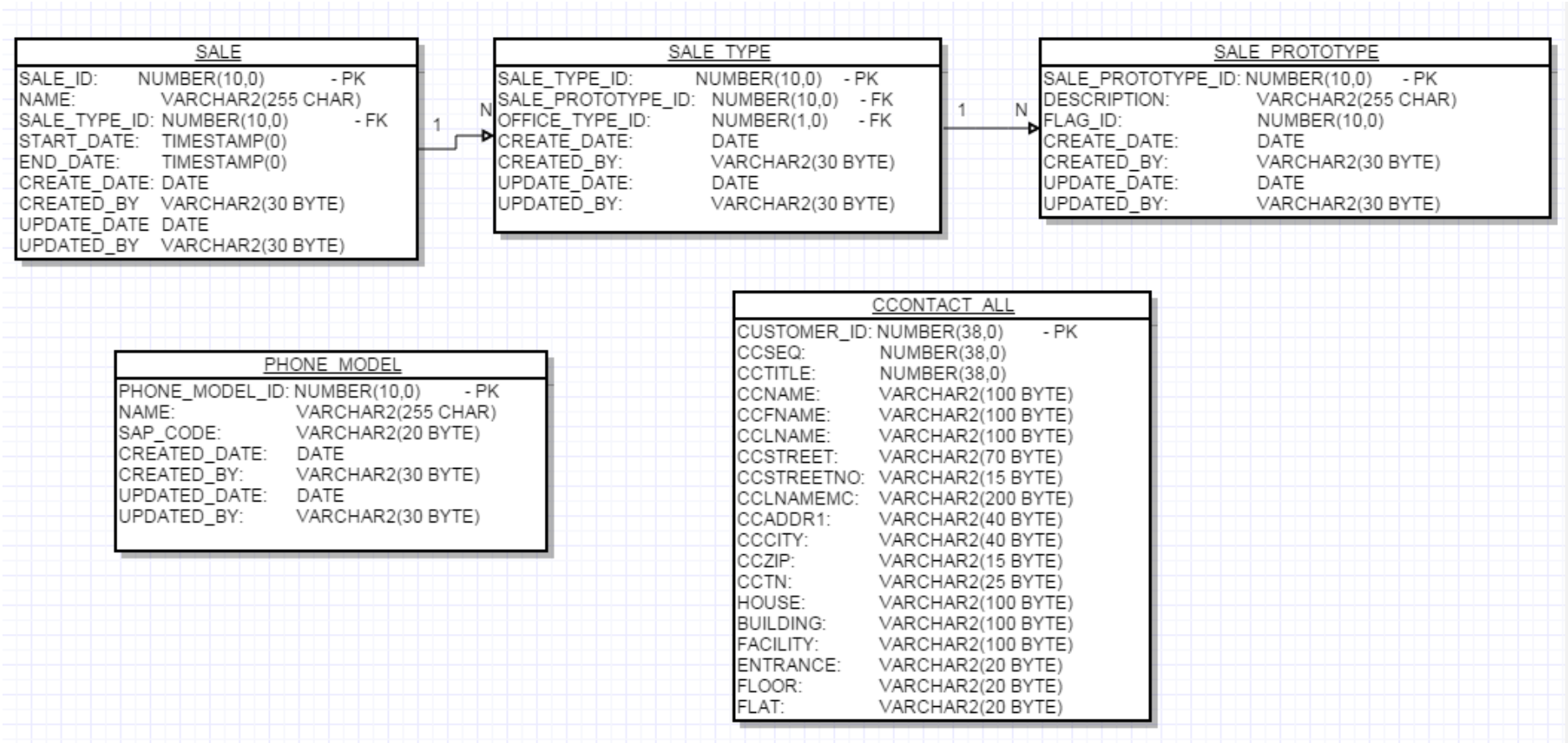


Рисунок 2.16 – Частина схеми БД проекту VC Billing New, що використовується в БД системи

COLUMN_NAME	DATA_TYPE	
ID_ITEM	NUMBER	№ запису
ID_ORDER	VARCHAR2(30 BYTE)	№ заявки
ID_CARRIER	VARCHAR2(30 BYTE)	перевізник
PERIOD_FROM	TIMESTAMP(6)	дата початку періоду
PERIOD_TO	TIMESTAMP(6)	дата кінця періоду
CREATE_DATE	TIMESTAMP(6)	дата створення запису
CREATED_BY	VARCHAR2(200 CHAR)	ким був створений запис
UPDATE_DATE	TIMESTAMP(6)	дата зміни запису
UPDATED_BY	VARCHAR2(200 CHAR)	ким був змінений запис

Рисунок 2.17 – Опис таблиці WS_TIME_SCHEDULER_ITEM

Зображені таблиці, які використовуються в локальньній схемі БД сервісу, що розробляється. Таблиця PHONE_MODEL – містить інформацію про товар, який продається в Інтернет магазині (рис. 2.18).

COLUMN_NAME	DATA_TYPE	
PHONE_MODEL_ID	NUMBER(10,0)	№ моделі обладнання
NAME	VARCHAR2(255 CHAR)	назва
SAP_CODE	VARCHAR2(20 BYTE)	код
CREATED_DATE	DATE	дата створення
CREATED_BY	VARCHAR2(30 BYTE)	ким створено
UPDATED_DATE	DATE	дата зміни
UPDATED_BY	VARCHAR2(30 BYTE)	ким було змінено

Рисунок 2.18 – Опис таблиці PHONE_MODEL

Таблиця CCONTACT_ALL – містить інформацію про абонентів даної білінгової системи, які являються потенційними клієнтами Інтернет магазину. Опис таблиці зображено на рисунку 2.19.

Таблиці SALE, SALE_TYPE, SALE_PROTOTYPE – містять інформацію про типи продажу товарів (акції, розстрочка). Опис таблиць зображено на рисунку 2.20.

COLUMN_NAME	DATA_TYPE	
PATRONYMIC_NAME	VARCHAR2(100)	ПІБ клієнта
HOUSE	VARCHAR2(100)	будинок
BUILDING	VARCHAR2(100)	будівля
FACILITY	VARCHAR2(100)	будова
ENTRANCE	VARCHAR2(20)	в'їзд
FLOOR	VARCHAR2(20)	поверх
FLAT	VARCHAR2(20)	квартира
POST_OFFICE_BOX	VARCHAR2(20)	поштове віділення
AREA_ID	NUMBER(38)	область
CITY_TYPE_ID	NUMBER(38)	тип міста
STREET_TYPE_ID	NUMBER(38)	тип вулиці
IMSI	VARCHAR2(24)	індекс
DEF_NOTIF_CO...	CHAR(1)	чи є нотиф. по створенню
DEF_INSTADDR...	CHAR(1)	чи є нотифікац. по зміні даних
CUSTOMER_ID	NUMBER(38)	номер клієнта

Рисунок 2.19 – Опис таблиці CCONTACT_ALL

COLUMN_NAME	DATA_TYPE	
SALE_ID	NUMBER(10,0)	№ акції
NAME	VARCHAR2(255 CHAR)	назва
SALE_TYPE_ID	NUMBER(10,0)	№ типу акції
START_DATE	TIMESTAMP(0)	дата початку дії
END_DATE	TIMESTAMP(0)	дата закінчення дії
CREATE_DATE	DATE	дата створення
CREATED_BY	VARCHAR2(30 BYTE)	ким було створено
UPDATE_DATE	DATE	дата зміни
UPDATED_BY	VARCHAR2(30 BYTE)	ким було змінено

COLUMN_NAME	DATA_TYPE	
SALE_TYPE_ID	NUMBER(10,0)	№ типу акції
SALE_PROTOTYPE_ID	NUMBER(10,0)	№ типу продажу
OFFICE_TYPE_ID	NUMBER(1,0)	№ типу офісу для продажу
CREATE_DATE	DATE	дата створення
CREATED_BY	VARCHAR2(30 BYTE)	ким створено

а) б)

COLUMN_NAME	DATA_TYPE	
SALE_PROTOTYPE_ID	NUMBER(10,0)	№ типу продажу
DESCRIPTION	VARCHAR2(255 CHAR)	опис
FLAG_ID	NUMBER(10,0)	прапорець на розстрочку
CREATE_DATE	DATE	дата створення
CREATED_BY	VARCHAR2(30 BYTE)	ким створено

в)

Рисунок 2.20 – Опис таблиць:

а) таблиця SALE; б) таблиця SALE_TYPE; в) таблиця SALE_PROTOTYPE

3 ТЕСТУВАННЯ ТА ІЛЮСТРАЦІЯ РОБОТИ СИСТЕМИ

3.1 Дизайн сервісу

Сервіс є частиною біллінгової системи VC Billing New, тому для початку роботи необхідно перш за все, пройти процедуру авторизації. Діаграма переходів між сторінками модуля зображена на рисунку3.1.

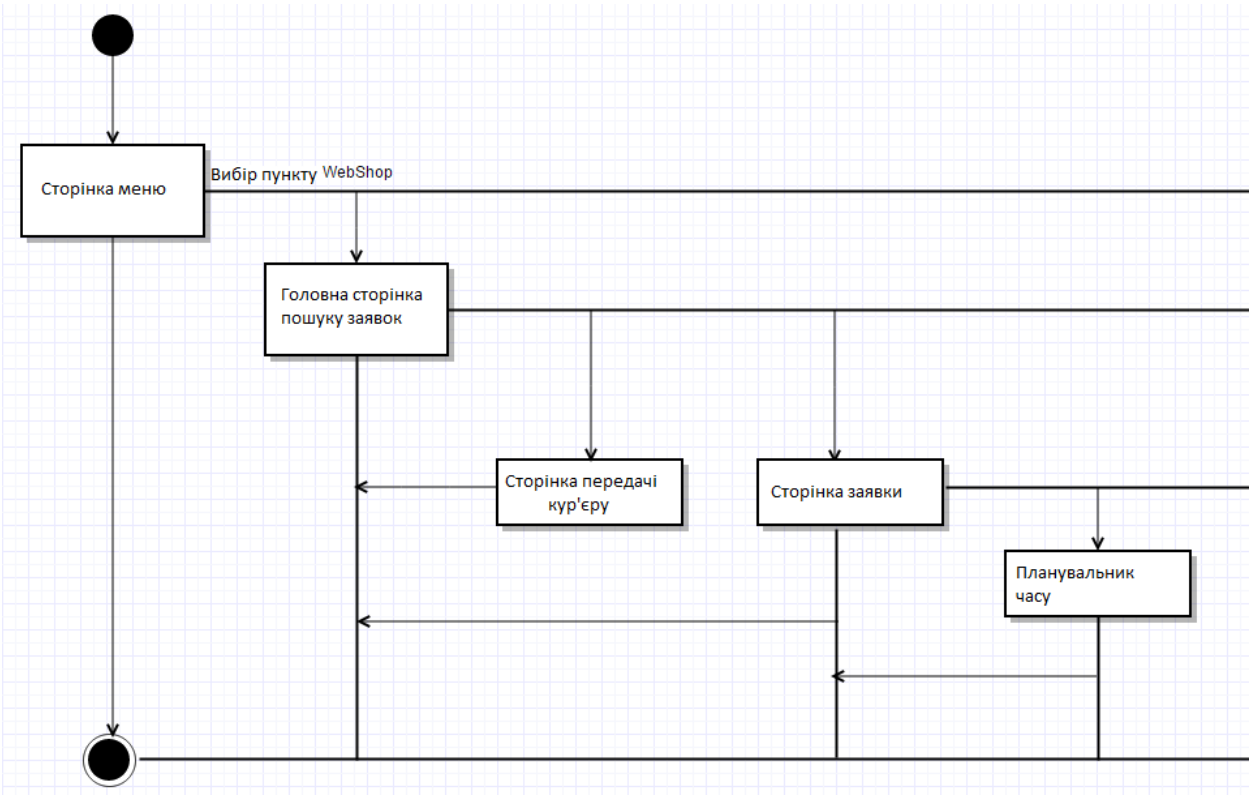


Рисунок 3.1 – Діаграма переходу між сторінками

3.2 Ілюстрація роботи головної сторінки

Безпеку аутентифікації гарантує використання технології Spring Security. При невірному вводі логіну або пароля, користувач отримує повідомлення про помилку входу в систему (рис. 3.2).

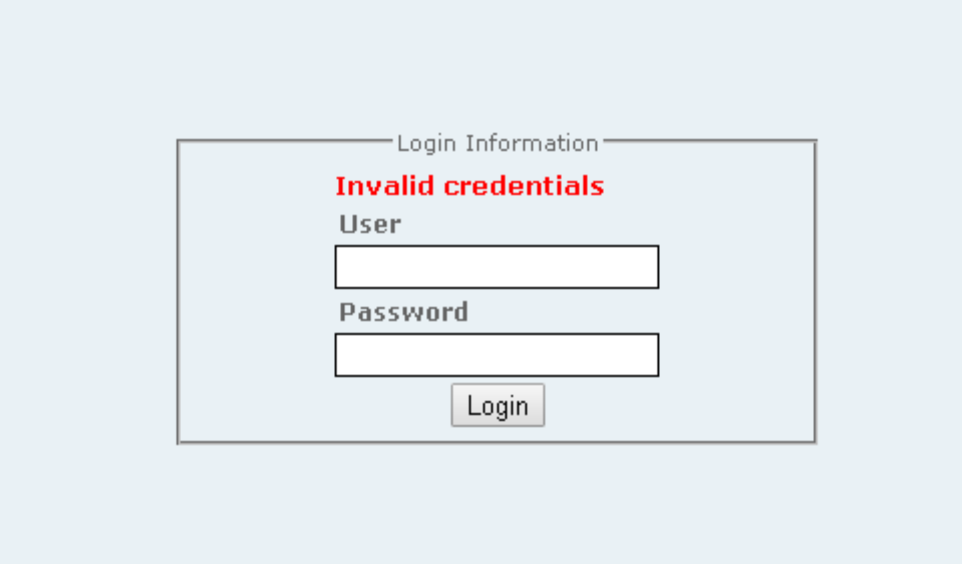


Рисунок 3.2 – Повідомлення про помилку входу в систему

При успішній аутентифікації користувач обирає в меню «Интернет магазин» і потрапляє на сторінку пошуку замовлень (рис.3.3).

Рисунок 3.3 – Сторінка пошуку замовлень по фільтру

При пошуку замовлень можна використовувати фільтр по необхідним полям:

– № замовлення;
– назва клієнта;
– спосіб доставки;
– тип продажу;
– статус замовлення;
– тип оплати;

– MSISDN абонента;

– тип клієнта;

– співробітник, що створив замовлення.

Також є кнопка «Сформувати звіт», яка стає активною після вибору необхідних для звіту замовлень. Приклад звіту по замовленням зображений на рисунку 3.4.

ОТЧЕТ
ПО ЗАЯВКАМ, ОБРАБОТАННЫМ ЗА ПЕРИОД

№	Статус	ID заявки	ID клиента	ФИО/Наименование	Тип продажи	Модель	Способ доставки	Способ оплаты	Количество товара	Полная стоимость	Первоначальный взнос
1	В работу	108099	10902321	Лысенко Татьяна Владик	Цена со скидкой	Alcatel 2000Xчерный	Белпочта	Наличными кур	1	449000	0
2	В работу	108100	1793657	Богородов Владимир Юр	Специальная расср	LG D855 (G3) 16Гб белый	Белпочта	Наличными кур	1	8587000	1999000
3	В работу	108102	DUMMY	тест тест	Полная стоимость	Samsung SM-G530H бель	Белпочта	WebPay	1	3197000	0
4	В работу	108103	DUMMY	тест тест	Полная стоимость	Samsung SM-G530H бель	Белпочта	WebPay	1	3197000	0
5	В работу	108104	DUMMY	тест тест	Полная стоимость	Samsung SM-G530H бель	Белпочта	WebPay	1	3197000	0
6	В работу	108122	DUMMY	тест тест	Полная стоимость	Samsung SM-G530H бель	Белпочта	WebPay	1	3197000	0
7	В работу	108142	DUMMY	Иттест Иттест	Полная стоимость	Samsung SM-G530H бель	Белпочта	WebPay	1	3987000	0
8	В работу	108143	DUMMY	тест тест	Полная стоимость	Samsung SM-G530H бель	Белпочта	WebPay	1	3987000	0
9	В работу	108144	DUMMY	тест тест	Полная стоимость	Samsung SM-G530H бель	Белпочта	WebPay	1	3987000	0
10	В работу	108145	DUMMY	fff fff	Полная стоимость	Samsung SM-G530H бель	Самовывоз	WebPay	1	3987000	0
11	В работу	108146	DUMMY	fff fff	Полная стоимость	Samsung SM-G530H бель	Самовывоз	WebPay	1	3987000	0
12	В работу	108147	DUMMY	fff fff	Полная стоимость	Samsung SM-G530H бель	Самовывоз	WebPay	1	3987000	0
13	В работу	108148	12822095	Иттест Иттест НЕ МЕНЯТ	Специальная расср	Samsung SM-G530H бель	Белпочта	WebPay	1	399000	399000
14	В работу	108149	DUMMY	тест тест	Полная стоимость	Samsung SM-G530H бель	Белпочта	WebPay	1	3987000	0
15	В работу	108150	12822095	Иттест Иттест НЕ МЕНЯТ	Специальная расср	Samsung SM-G530H бель	Белпочта	Наличными кур	1	399000	399000
16	В работу	108151	12822095	Иттест Иттест НЕ МЕНЯТ	Быстрая рассрочка	Samsung SM-G530H бель	Белпочта	WebPay	1	950000	950000

Рисунок 3.4 – Звіт по замовленням

3.3 Ілюстрація роботи сторінки – відображення замовлення

При натисканні на номер замовлення – потрапляємо на сторінку заявки. На рисунку 3.5 зображена заявка у статусі «Готова к отгрузке». На сторінці заявки відображена інформація щодо замовлення, а також можливість переводу заявки в інший статус або друк необхідних документів.

При натисканні на кнопку «Печать документов» користувач отримує вікно вибору документу: накладна або ярлик кур’єра. На рисунку 3.6 забражено вікно прінтингу.

Приклад документа «Ярлик кур’єра» зображено на рисунку 3.7. Документ завантажується в форматі .rtf. Формування документу на основі шаблону, відбувається за домогою технології FOP.

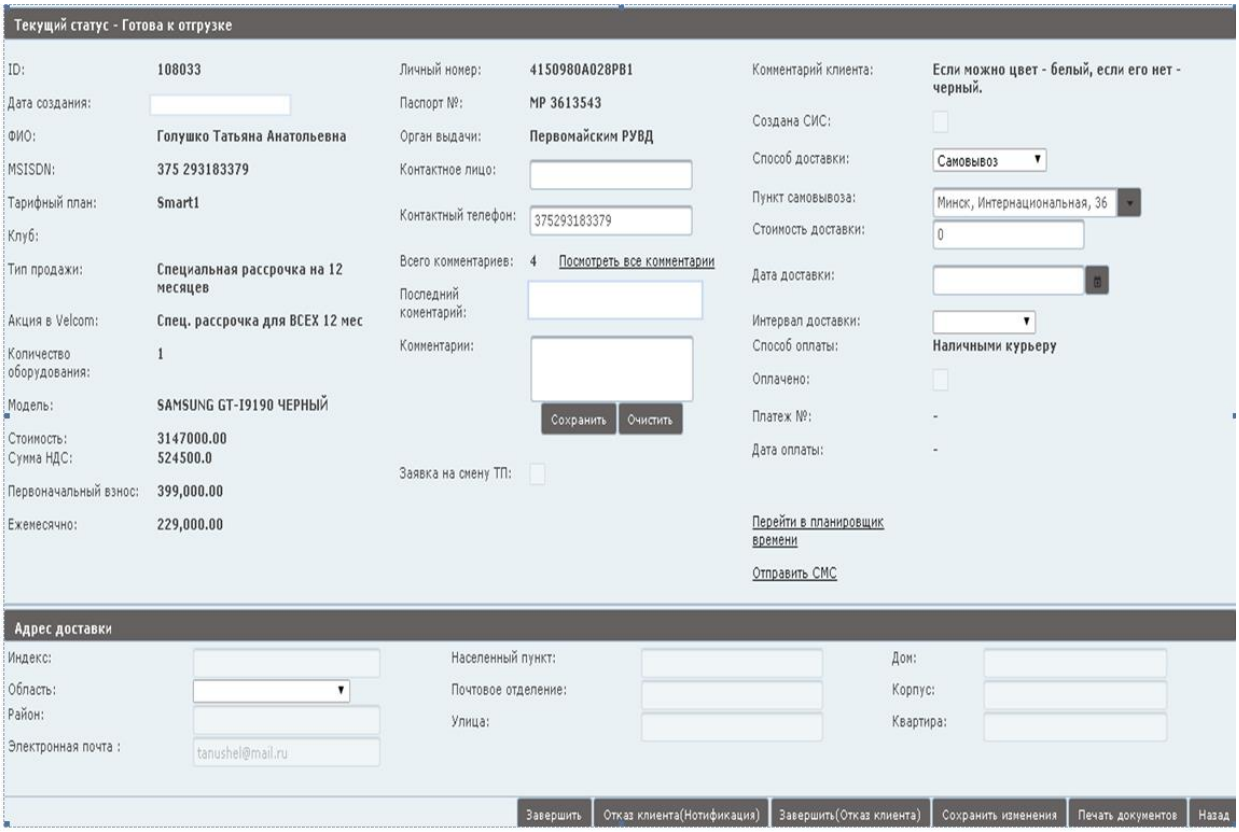


Рисунок 3.5 – Заявка у статусі «Готова к отгрузке»

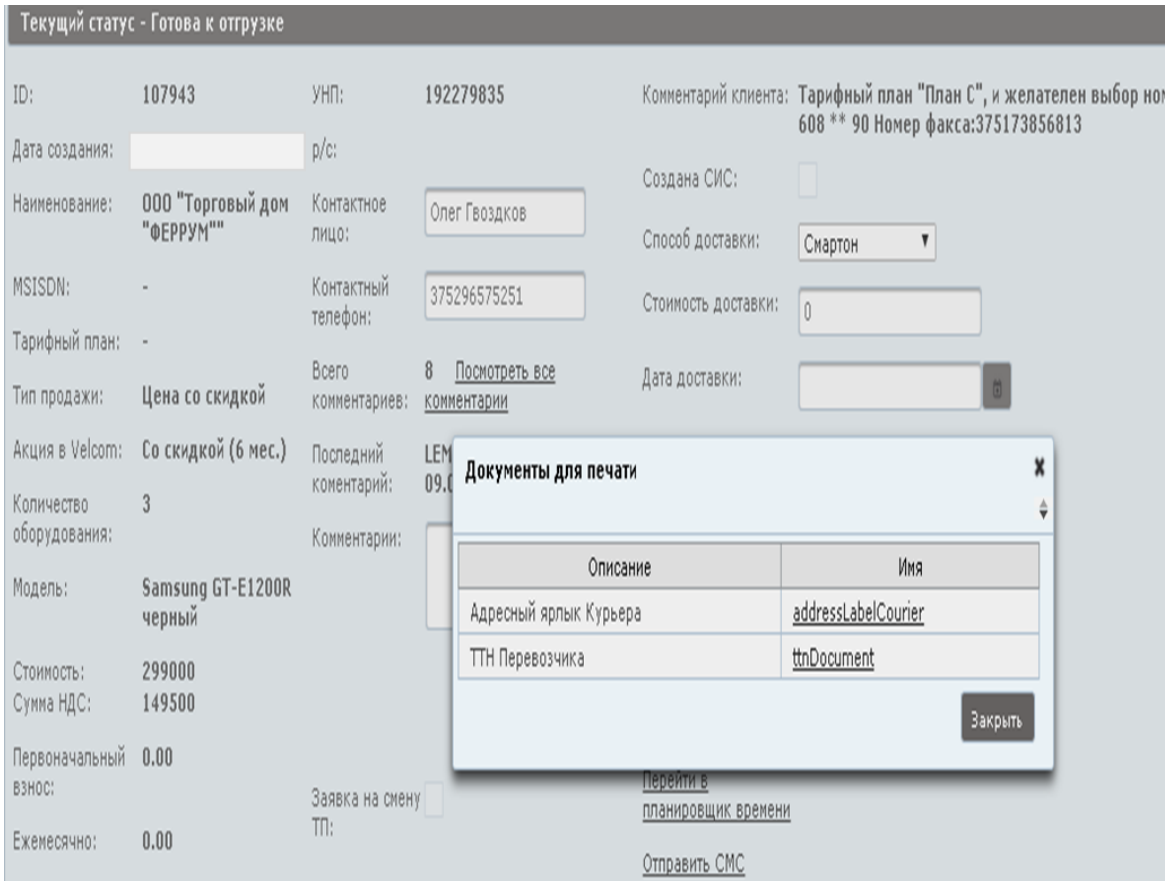
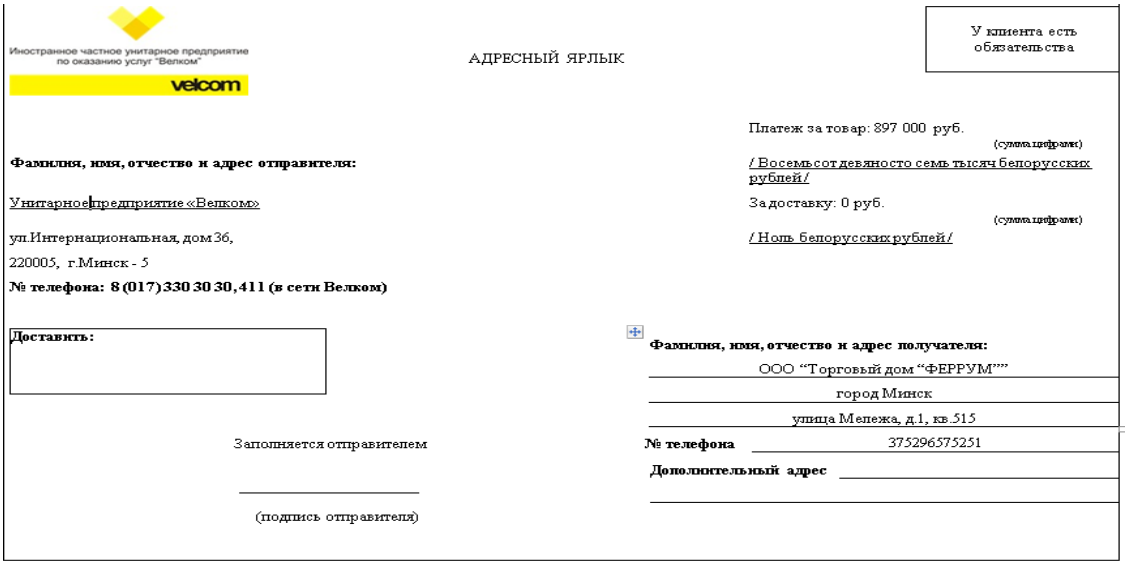


Рисунок 3.6 – Вікно прінтингу документів



Иностранное частное унитарное предприятие по оказанию услуг "Велком"

velcom

АДРЕСНЫЙ ЯРЛЫК

У клиента есть обязательства

Платеж за товар: 897 000 руб.
(сумма цифрами)
/ Восемьсот девяносто семь тысяч белорусских рублей /

За доставку: 0 руб.
(сумма цифрами)
/ Ноль белорусских рублей /

Фамилия, имя, отчество и адрес отправителя:

Унитарное предприятие «Велком»

ул.Интернациональная, дом 36,

220005, г.Минск - 5

№ телефона: 8 (017) 330 30 30, 411 (в сети Велком)

Доставить:

Заполняется отправителем

(подпись отправителя)

Фамилия, имя, отчество и адрес получателя:

ООО "Торговый дом "ФЕРРУМ""

город Минск

улица Мележа, д.1, кв.515

№ телефона 375296575251

Дополнительный адрес

Рисунок 3.7 – Приклад документа для друку «Ярлик кур'єра»

Документ «Накладна» завантажується в форматі .xlsx. Приклад документу зображений на рисунку 3.8.

Заказчик автомобильной перевозки (плательщик) Унитарное предприятие Велком, 220030, г.Минск,ул. Интернациональная, 36-2
(наименование, адрес)

Грузоотправитель Унитарное предприятие Велком, 220030, г.Минск,ул. Интернациональная, 36-2
(наименование, адрес)

Грузополучатель ООО "Торговый дом "ФЕРРУМ""город Минск, улица Мележа, д.1, кв.515
(наименование, адрес)

Основание отпуска (дата и номер договора или другого документа) Пункт погрузки г.Минск,ул. Интернациональная, 36-2 (адрес) Пункт разгрузки Минская область Мински (адрес)

Переадресовка нет
(наименование, адрес нового грузополучателя, фамилия, инициалы, подпись уполномоченного должностного лица)

I. ТОВАРНЫЙ РАЗДЕЛ

Наименование товара	Единица измерения	Количество	Цена, руб.	Стоимость, руб.	Ставка НДС, %	Сумма НДС, руб.	Стоимость с НДС, руб.	Кол-во грузовых мест	Масса груза	Примечание
1	2	3	4	5	6	7	8	9	10	11
Итого	X	3	249167	747500	20	149500	897000			X

Товар, оборудование, согласно приложению к _____ от года в количестве 1 листов.

Количество ездок (заездов)

Всего сумма НДС Сто сорок девять тысяч пятьсот белорусских рублей
(прописью)

Всего стоимость с НДС Восемьсот девяносто семь тысяч белорусских рублей
(прописью)

Рисунок 3.8 - Приклад документу «Накладна»

При виборі кнопки «Отказ клиента (Нотификация)», на відповідну електронну адресу оператора приходить лист про відмову клієнту від замовлення.

Зі сторінки заявки можно також відправити СМС клієнту. Відправка повідомлень здійснюється за допомогою API бібліотеки «SMS - центр». Форма відправлення СМС зображена на рисунку 3.9.

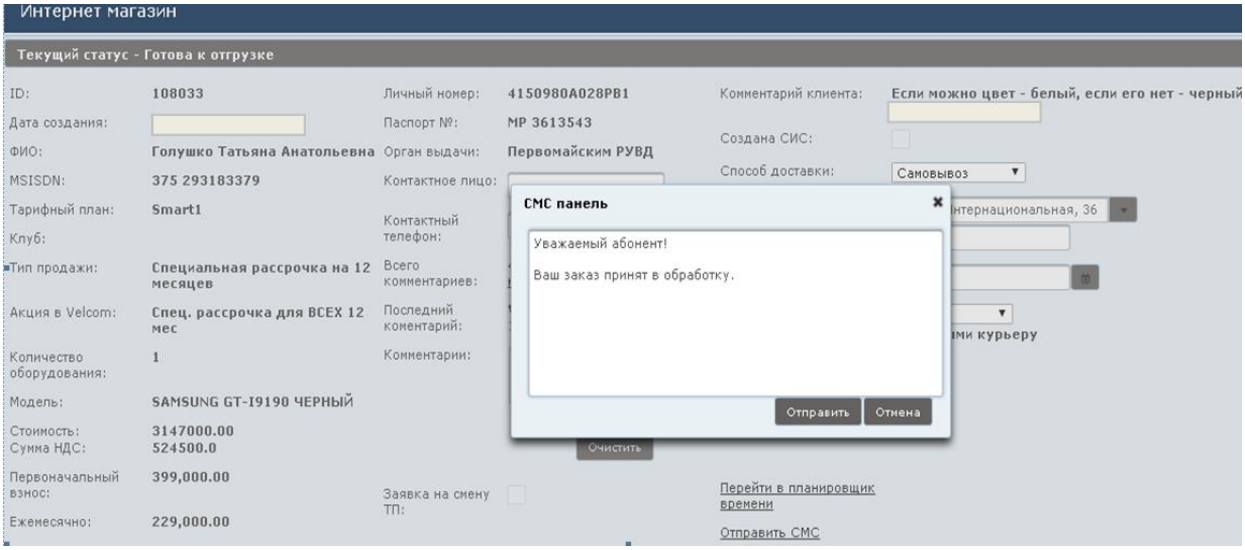


Рисунок 3.9 – Форма відправлення СМС клієнту

Також зі сторінки заявки можна перейти у планувальник часу для перевізників. Для цього необхідно натиснути на лінк «Перейти в планировщик времени». Функціонал виконаний за допомогою компонента Primefaces p:schedule. Інтерфейс функціоналу зображений на рисунку 3.10.

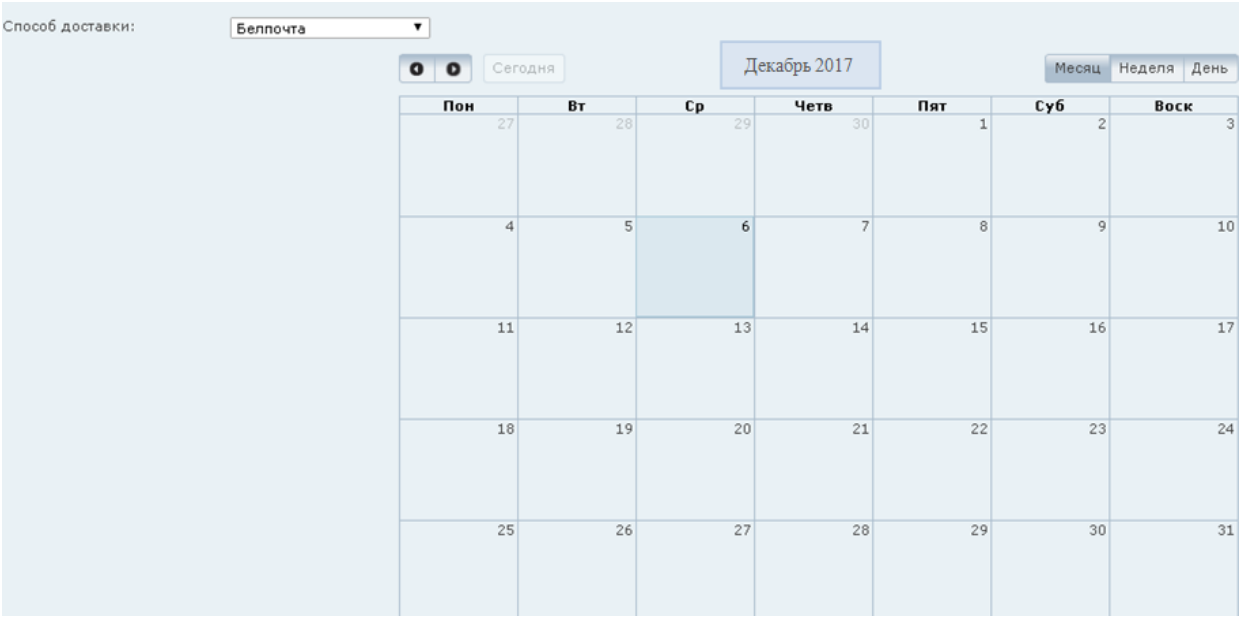


Рисунок 3.10 – Планувальник часу для перевізників

3.4 Ілюстрація роботи сторінки «Передача замовлень кур'єру»

У замовлень, які перебувають в статусі «Готова к отгрузке», вже уточнені всі деталі з клієнтом, і для успішної доставки залишається лише обрати тип перевізника і перевести заявку у статус «Передана курьеру». Для цього з головної сторінки пошуку заявок обираємо фільтр по полю «Статус», відмічаємо потрібні заявки і натискаємо на кнопку «Передать курьеру». Перехід на сторінку передачі кур'єру зображений на рисунку 3.11.

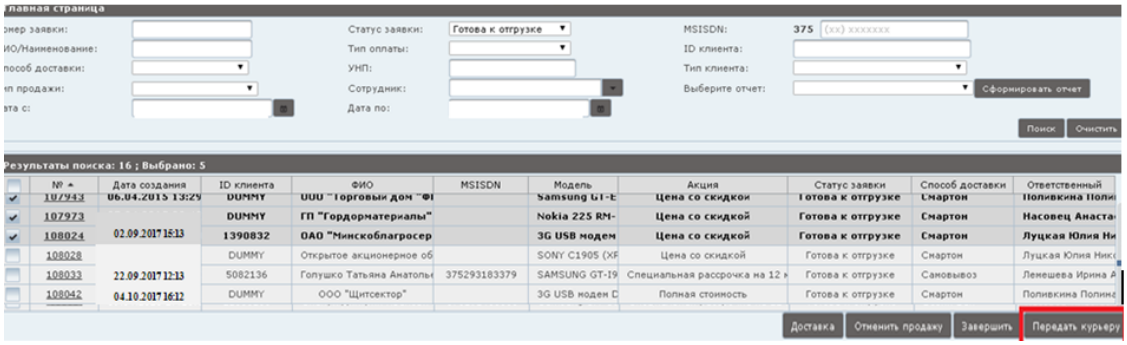


Рисунок 3.11 – Перехід на сторінку «Передачи курьеру»

Після цього, оператор потрапляє на сторінку «Передачи курьеру», на якій можна здійснити фільтр заявок по полю «Превозчик» та «Способ оплати», і передати заявки перевізнику, змінивши їх статус, а також надрукувати необхідні документи. Сторінка «Передачи курьеру» з можливістю друку документів, зображена на рисунку 3.12.

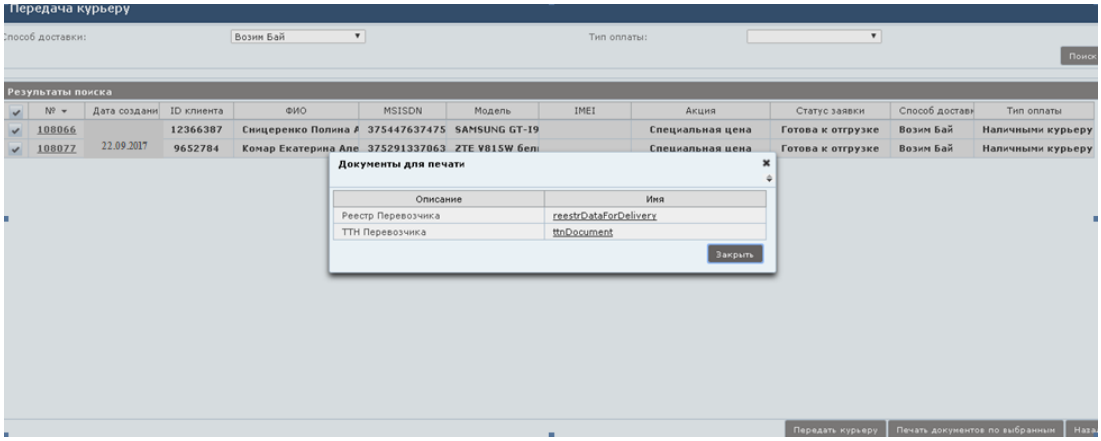


Рисунок 3.12 – Сторінка «Передачи курьеру»

3.5 Тестування роботи системи

Для тестування функціональності сервісу були створені тест-кейси, які покривають основну частину працездібності системи. В якості програмного забезпечення, використовувалась програма TestLink. TestLink – це система управління тестами з веб-інтерфейсом.

Перш за все, потрібно перевірити чи приходять нотифікації оператору коли нова заявка потрапляє в систему. Тест-кейс для перевірки цього функціоналу зображений на рисунку 3.13.

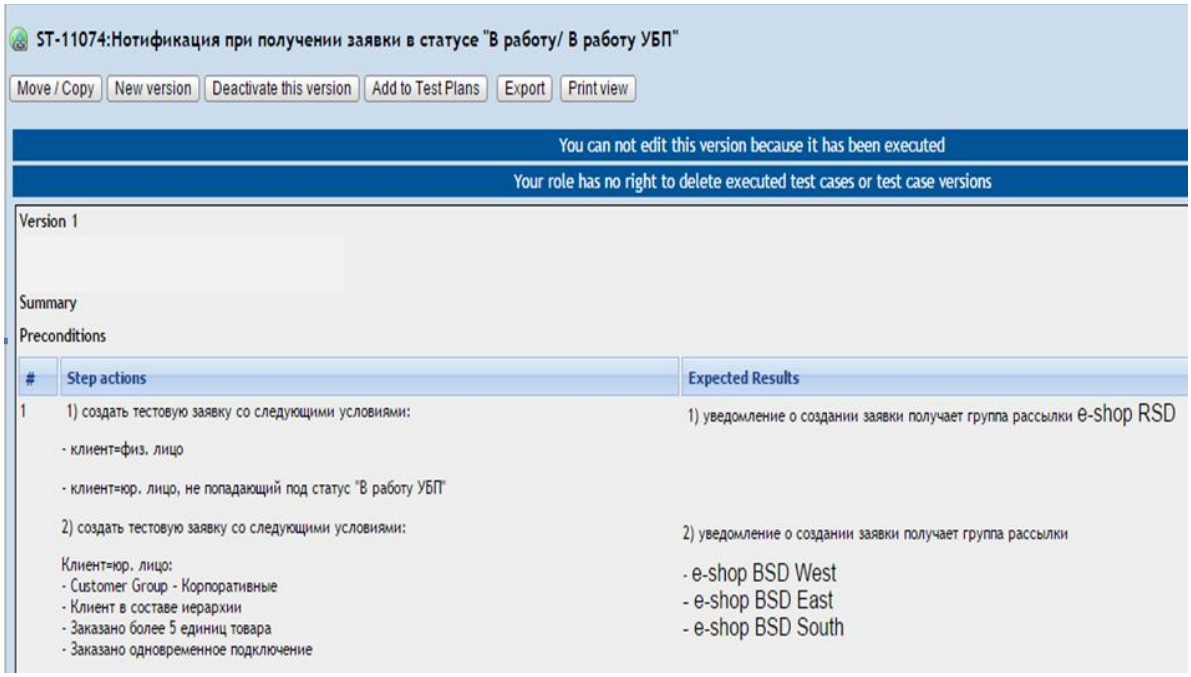


Рисунок 3.13 – Тест-кейс для перевірки роботи нотифікацій

Далі перевіряємо чи коректно працює пошук заявок на головній сторінці (рис.3.14). Для цього необхідно заповнити один із фільтрів пошуку та натиснути кнопку «Поиск». Результатом має бути таблиця із заявками, які відповідають даному критерію.

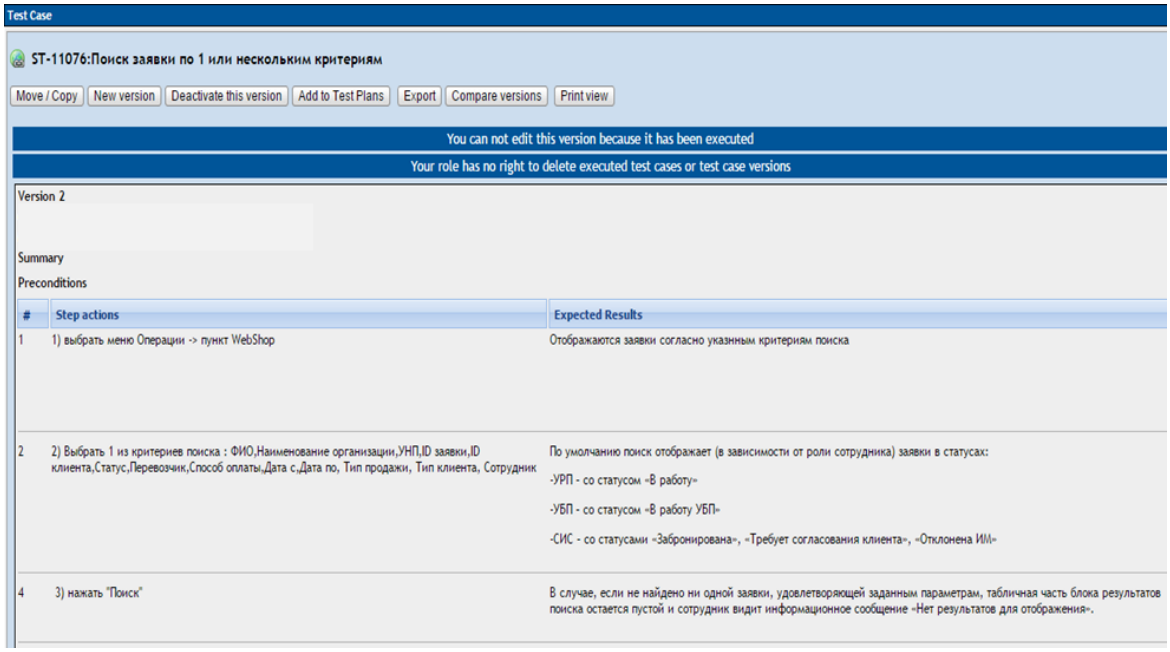


Рисунок 3.14 – Тест-кейс на перевірку пошуку заявок

Для перевірки правильності друку звітів необхідно здійснити пошук заявок, обрати потрібні заявки мітками та натиснути кнопку «Печать документов». При цьому на формі відображається весь доступний список документів для друку. Оператор обирає потрібний йому, натиснувши на лінк цього документу. Приклад цього тест кейсу зображено на рисунку 3.15.

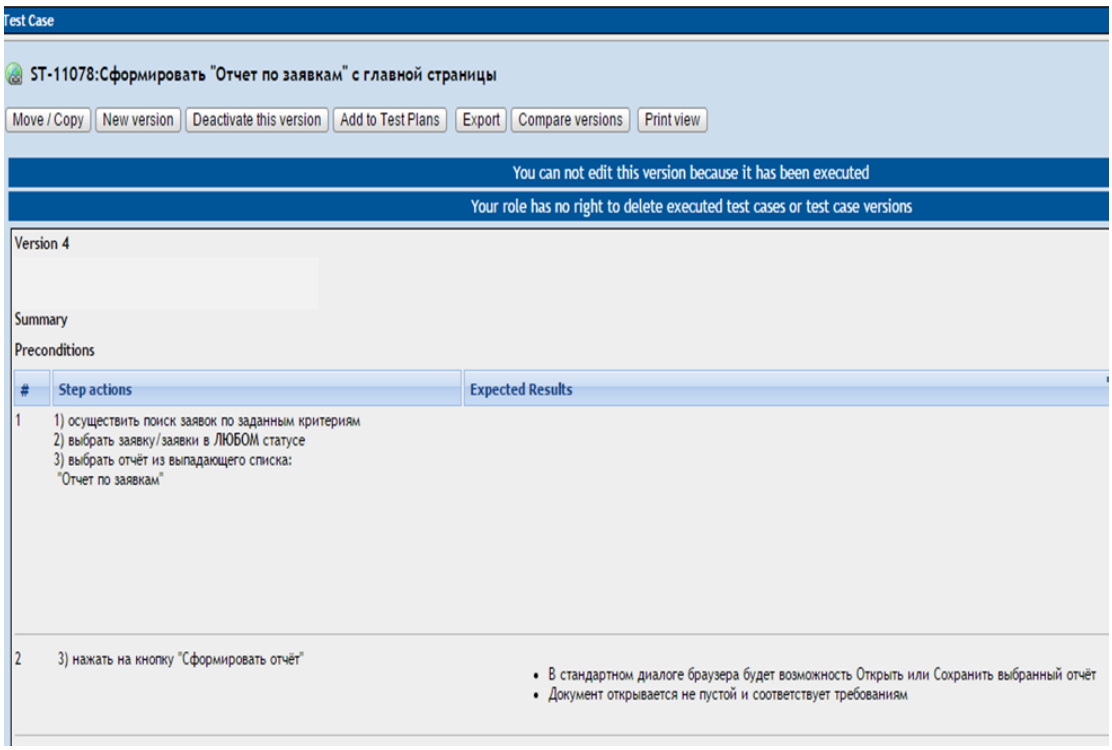


Рисунок 3.15 – Тест-кейс на перевірку коректного формування звіту

Для перевірки працездатності «Планувальника часу» необхідно перейти на сторінку заявки, далі натиснути на лінк «Планировщик времени». На цій сторінці відображений компонент у вигляді календаря. В кожному осередку, який відповідає даті, зображені перевезення кур'єра. Оператору необхідно натиснути на вільний осередок та встановити значення «Дата с», «Дата по», зарезервувавши таким чином послуги кур'єра на даний час. Опис тест-кейсу зображений на рисунку 3.16.

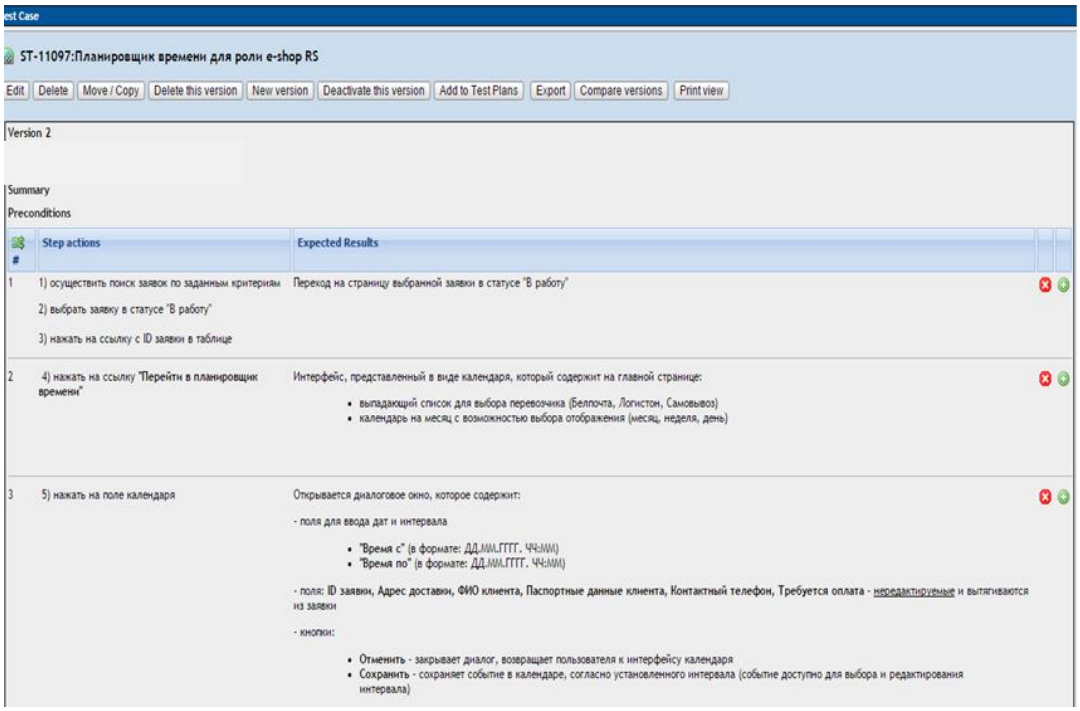


Рисунок 3.16 – Тест-кейс на перевірку працездібності «Планувальника часу»

Для перевірки функціональності сторінки обробки заявки, необхідно перейти на заявку зі статусом «Подтверждена» та натиснути на кнопку «Начать обработку». Після цього повинна з'явитися кнопка для друку документів, а також кнопка для переходу в статус «Готова к отгрузке». Тест-кейс перевірки функціональності зображено на рисунку 3.17.

Для перевірки автоматичного відправлення щомісячних звітів на електронну поштову скриньку, необхідно зайти на тестовий поштовий акаунт (використовується на тестовому оточенні) та знайти вхідний лист за перше число поточного місяця із двома прикріпленими звітами. Звіти формуються із

даних попереднього місяця. Тест кейс для перевірки зображений на рисунку 3.18.

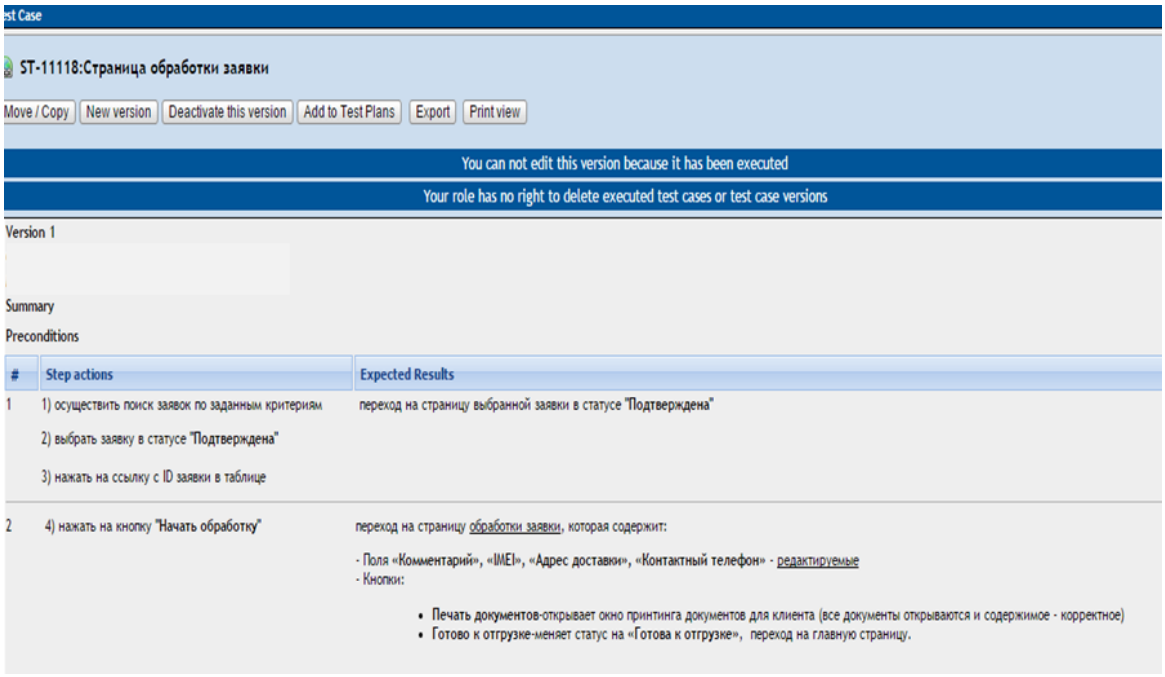


Рисунок 3.17 – Тест-кейс для перевірки функціональності сторінки обробки заявки

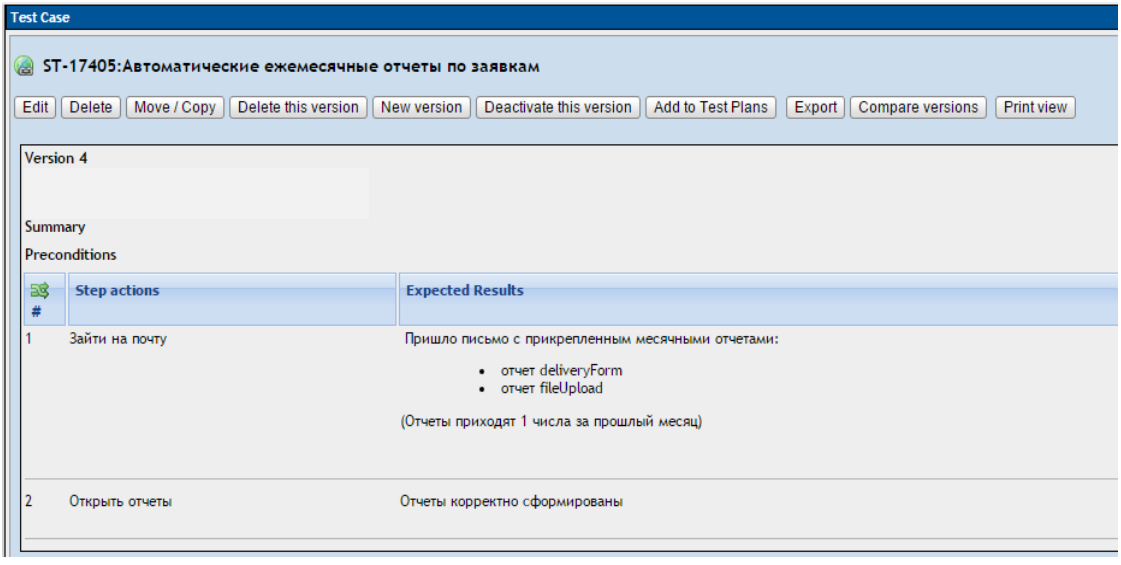


Рисунок 3.18 – Тест-кейс для перевірки автоматичного відправлення щомісячних звітів

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯЇХ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію [9]. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Завданням даної роботи бакалавра було створення сервісу для системи обліку та тарифікації наданих послуг. Так як в процесі проектування використовувалося комп’ютерне обладнання, то аналіз потенційно небезпечних і шкідливих чинників виконується для персонального комп'ютера, на якому буде виконуватися розробка.

4.1 Загальні питання з охорони праці

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. В законі України «Про охорону праці» визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

Основним організаційним напрямом у здійсненні управління в сфері охорони праці є усвідомлення пріоритету безпеки праці і підвищення соціальної відповідальності держави і особистої відповідальності.

4.2 Аналіз стану умов праці

Робота над створенням сервісу для системи обліку та тарифікації наданих послуг проходитиме в приміщенні багатоквартирного будинку. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп’ютером.

4.2.1 Вимоги до приміщень

Геометричні розміри приміщення зазначені в табл. 4.1.

Таблиця 4.1 – Розміри приміщення

Найменування	**Значення**
Довжина, м	5
Ширина, м	5
Висота, м	2.8
Площа, $м^2$	25
Об’єм, $м^3$	70

Згідно з [14] розмір площі для одного робочого місця оператора персонального комп’ютера має бути не менше 6 кв. м, а об’єм — не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

4.2.2 Вимоги до організації місця праці

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця за [12] (табл. 4.2) і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Приміщення знаходиться на другому поверсі трьох поверхової будівлі і

має об’єм 70 м^3, площу – 25 м^2. Обладнано одне місце праці укомплектоване ПК.

Таблиця 4.2 - Характеристики робочого місця

Найменування параметра	Фактичне значення	Нормативне значення
Висота робочої поверхні, мм	770	680 ÷800
Висота простору для ніг, мм	750	не менше 600
Ширина простору для ніг, мм	550	не менше 500
Глибина простору для ніг, мм	700	не менше 650
Висота поверхні сидіння, мм	450	400 ÷500
Ширина сидіння, мм	450	не менше 400
Глибина сидіння, мм	470	не менше 400
Висота поверхні спинки, мм	400	не менше 300
Ширина опорної поверхні спинки, мм	500	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	750	700 ÷800

Температура в приміщенні протягом року коливається у межах 18–24°С, відносна вологість — близько 50%. Швидкість руху повітря не перевищує 0,2 м/с. Шум знаходиться на рівні 50 дБА. Система вентилювання приміщення — природна неорганізована, а опалення — централізоване.

4.2.3 Навантаження та напруженість процесу праці

Під час виконання випускної роботи бакалавра:

за фізичним навантаженням робота відноситься до категорії легкі роботи (Іа), її виконують сидячи з періодичним ходінням. Щодо характеру організування виконання дипломної роботи, то він підпадає під нав’язаний режим, оскільки певні розділи роботи необхідно виконати у встановлені конкретні терміни.

Рекомендовано застосування екранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, а також інші профілактичні заходи наведені в [12].

Роботу за дипломним проектом визнано, таку, що займає 50% часу робочого дня та за восьмигодинної робочої зміни рекомендовано встановити додаткові регламентовані перерви програм тривалістю 15 хв. через кожну годину роботи.

4.3 Виробнича санітарія

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

4.3.1 Аналіз небезпечних та шкідливих факторів при виробництві (експлуатації) виробу

Аналіз небезпечних та шкідливих факторів виконується у табличній формі (табл. 4.3). Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання НПАОП 0.00.-1.28-10 [20], які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП. Основними робочими характеристиками персонального комп'ютера є:

- робоча напруга U=+220В +-5%;
- робочий струм I=2А;
- споживана потужність P=350 Вт.

Робоче місце має відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 N 7 [12].

Таблиця 4.3 – Аналіз небезпечних і шкідливих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
фізичні			
- підвищена температура поверхонь обладнання	експлуатація ЕОМ, принтерів, сканерів чи/або серверного обладнання для роботи	2	[14]
- підвищений рівень іонізуючого випромінення в робочій зоні	-//-	2	[14] [24]
- підвищений рівень електромагнітного випромінення	-//-	2	[24]
- підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини	-//-	4	[25] [14]
- підвищений рівень статичної електрики	-//-	2	[25]
- підвищена напруженість електричного поля	-//-	2	[24]
- підвищена напруженість магнітного поля	-//-	2	[24]
- недостатність природного світла	порушення умов праці (вимог до приміщень)	2	[11]
- недостатнє освітлення робочої зони	порушення гігієнічних параметрів виробничого середовища	3	[11]
- підвищена яскравість світла	порушення умов праці (організації місця праці- налагодження моніторів)	1	[12]
- понижена контрастність	-//-	1	[12]
психофізіологічні:			
- нервово-психічна перевантаження (розумове, перенапруження аналізаторів-зорових)	- пошук інформації для постановки теми; - пошук та аналіз аналогів і літератури; - пошук наявних технологій, моделювання та аналіз алгоритмів; - виконання роботи за темою диплома, тестування; - оформлення роботи	4	[26] [12]
- фізичні (статичне – сидіння)	порушення умов праці (організації місця праці - сидіння користувача) та організації робочого часу - безпервна робота)	2	[26] [12]

4.3.2 Пожежна безпека

Небезпека розвитку пожежі обумовлюється застосуванням розгалужених систем електроживлення ЕОМ, вентиляції і кондиціювання.

Запобігти утворенню горючого середовища (замінити горючі речовини і матеріали на негорючі і важкогорючі) не надається технічно можливим. Тому проектом передбачаються засоби запобігання утворення (або внесення) в горюче середовище джерел запалювання.

Згідно [18] таке приміщення, площею 25 $м^2$, відноситься до категорії "В" (пожежонебезпечної) та для протипожежного захисту в ньому можливо встановлення автоматичної пожежної сигналізації із застосуванням датчиків-сповіщувачів РІД-1 (сповіщувач димовий ізоляційний) в кількості 1 шт., і застосуванням первинних засобів пожежогасіння.

Продуктами згорання, що виділяються на пожежі, є: окис вуглецю; сірчистий газ; окис азоту; синильна кислота; акромін; фосген; хлор і ін. При горінні пластмас, окрім звичних продуктів згорання, виділяються різні продукти термічного розкладання: хлорангідридні кислоти, формальдегіди, хлористий водень, фосген, синильна кислота, аміак, фенол, ацетон, стирол [10].

4.3.3 Електробезпека

На робочому місці виконуються наступні вимоги електробезпеки: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія електромережі для живлення ПК, периферійних пристроїв і устаткування для обслуговування, виконана як окрема групова три- провідна мережа, шляхом

прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників мають спеціальні контакти для підключення нульового захисного провідника. Електромережа штепсельних розеток для живлення персональних ПК укладено по підлозі поруч зі стінами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання. Металеві труби та гнучкі металеві рукави заземлені. Захисне заземлення включає в себе заземлюючих пристроїв і провідник, який з'єднує заземлюючий пристрій з обладнанням, яке заземлюється - заземлюючий провідник.

4.4 Гігієнічні вимоги до параметрів виробничого середовища

4.4.1 Мікроклімат

Мікроклімат робочих приміщень – це клімат внутрішнього середовища цих приміщень, що визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря. Оптимальні значення мікроклімату для робочого місця відповідають [13] (табл. 4.4):

Таблиця 4.4 – Норми мікроклімату робочої зони об’єкту

Період року	**Категорія робіт**	**Температура C^0**	**Відносна вологість %**	**Швидкість руху повітря, м/с**
Холодна	легка-1 а	22 - 24	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

У приміщенні на робочому місці забезпечуються оптимальні значення параметрів мікроклімату. Дане приміщення обладнане системою опалення, кондиціонування повітря. Також має здійснюватися провітрювання

приміщення, в залежності від погодних умов, тривалість повинна бути не менше 10 хв. Найкращий обмін повітря здійснюється при наскрізному провітрюванні.

Рівні позитивних і негативних іонів у повітрі мають відповідати [13].

4.4.2 Освітлення

Світло є природною умовою існування людини. Воно впливає на стан вищих психічних функцій і фізіологічні процеси в організмі. Хороше освітлення діє тонізуюче, створює гарний настрій, покращує протікання основних процесів вищої нервової діяльності.

У приміщенні, де розташовані ЕОМ передбачається природне бічне освітлення, рівень якого відповідає [11]. Джерелом природного освітлення є сонячне світло. Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДБН і для даного приміщення в світлий час доби достатньо природного освітлення.

Розрахунок освітлення.

Для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше -1/8, в побутових – 1/10:

$$S_b = \left(\frac{1}{5} \div \frac{1}{10}\right) \cdot S_n, \qquad (4.1)$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$S_n = a \cdot b = 5 \cdot 5 = 25$ м^2,

$S = 1/8 \cdot 25 = 3{,}125$ м^2.

Приймаємо 2 вікна площею S=1,6 м^2 кожне.

Розрахунок штучного освітлення виробляється по коефіцієнтах

використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні.

Розрахунок кількості світильників n виробляється по формулі (4.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M}, \qquad (4.2)$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 25 м^2;

Z – поправочний коефіцієнт світильника (Z=1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (А.2), отримуємо:

$$n = \frac{300 \cdot 25 \cdot 1{,}1 \cdot 1{,}5}{5400 \cdot 0{,}575 \cdot 2} \approx 2{,}0$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

4.4.3 Шум та вібрація, електромагнітне випромінювання

Рівень шуму, зумовлений як роботою системного блоку, клавіатури, так і друкуванням на принтері, а також зовнішніми чинниками, коливається у межах 50–65 дБА [13].

Віброізоляцію можливо здійснювати за допомогою спеціальної прокладки під системний блок, яка послаблює передачу вібрацій робочого столу. Вібрація на робочому місці в приміщенні, що розглядається, відповідає нормам [13].

4.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Відповідно до санітарно-гігієнічних нормативів та правил експлуатації обладнання передбачено наступні заходи безпеки під час експлуатації персонального комп’ютера та периферійних пристроїв:

- правильне організування місця праці та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;
- експлуатацію сертифікованого обладнання;
- дотримання заходів електробезпеки;
- забезпечення оптимальних параметрів мікроклімату;
- забезпечення раціонального освітлення місця праці (освітленість робочого місця не перевищувала 2/3 нормальної освітленості приміщення);
- облаштовуючи приміщення для роботи з ПК, потрібно передбачити припливно-витяжну вентиляцію або кондиціювання повітря.

Крім того, потрібно дотримуватися правил безпеки під час експлуатації інших електричних приладів та вимоги безпеки при надзвичайних ситуаціях.

Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Згідно з класифікацією приміщень за ступенем небезпеки ураження електричним струмом [19], приміщення в якому проводиться робота відноситься до першого класу (без підвищеної небезпеки). Коефіцієнт використання вертикальних заземлювачів $\eta_в$ в залежності від розміщення заземлювачів та їх кількості знаходиться в межах 0,4…0,99. Взаємну екрануючу дію горизонтального заземлювача (з'єднувальної смуги) враховують за допомогою коефіцієнта використання горизонтального заземлювача η_c.

Послідовність розрахунку.

1) Визначається необхідний опір штучних заземлювачів $R_{шт.з.}$:

$$R_{шт.з.} = \frac{R_д \cdot R_{пр.з.}}{R_{пр.з.} - R_д}, \qquad (4.3)$$

де $R_{пр.з.}$ – опір природних заземлювачів; $R_д$ – допустимий опір заземлення. Якщо природні заземлювачі відсутні, то $R_{шт.з.}=R_д$.

Підставивши числові значення у формулу (А.3), отримуємо:

$$R_{шт.з.} = \frac{4 \cdot 40}{40 - 4} \approx 4 \text{ Ом}$$

2) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом·м. Приблизне значення питомого опору глини приймаємо ρ=40 Ом•м (табличне значення).

3) Розрахунковий питомий опір ґрунту, $\rho_{розр}$, Ом·м, визначається відповідно для вертикальних заземлювачів $\rho_{розр.в}$, і горизонтальних $\rho_{розр.г}$, Ом·м за формулою:

$$\rho_{розр..} = \psi \cdot \rho, \qquad (4.4)$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів I кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів $\rho_{розр.в}$=1,7 і горизонтальних $\rho_{розр.г}$=5,5 Ом·м.

$$\rho_{розр.в} = 1,7 \cdot 40 = 68 \text{ Ом•м}$$

$$\rho_{розр.г} = 5,5 \cdot 40 = 220 \text{ Ом•м}$$

4) Розраховується опір розтікання струму вертикального заземлювача $R_в$, Ом, за формулою (4.5).

$$R_в = \frac{\rho_{розр.в}}{2 \cdot \pi \cdot l_в} \cdot \left(\ln \frac{2 \cdot l_в}{d_{ст}} + \frac{1}{2} \cdot \ln \frac{4 \cdot t + l_в}{4 \cdot t - l_в} \right), \quad (4.5)$$

де $l_в$ – довжина вертикального заземлювача (для труб - 2–3 м; $l_в$=3 м);

$d_{ст}$ – діаметр стержня (для труб - 0,03–0,05 м; $d_{ст}$=0,05 м);

t – відстань від поверхні землі до середини заземлювача, яка визначається за формулою (4.6):

$$t = h_в + \frac{l_в}{2}, \quad (4.6)$$

де $h_в$ – глибина закладання вертикальних заземлювачів (0,8 м); тоді $t = 0,8 + \frac{3}{2} = 2,3$ м

$$R_в = \frac{68}{2 \cdot \pi \cdot 3} \cdot \left(\ln \frac{2 \cdot 3}{0,05} + \frac{1}{2} \cdot \ln \frac{4 \cdot 2,3 + 3}{4 \cdot 2,3 - 3} \right) = 18,5 \text{ Ом}$$

5) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання $\eta_в$:

$$n = \frac{2 \cdot R_в}{R_д} = \frac{2 \cdot 18,5}{4} = 9,25 \quad (4.7)$$

І визначається коефіцієнт використання вертикальних електродів групового заземлювача без врахування впливу з'єднувальної стрічки $\eta_в$ =0,57 (табличне значення).

6) Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання $n_в$, шт:

$$n_в = \frac{2 \cdot R_в}{R_д \cdot \eta_в} = \frac{2 \cdot 18,5}{4 \cdot 0,57} = 16,2 \approx 16 \quad (4.8)$$

7) Визначається довжина з'єднувальної стрічки горизонтального заземлювача $l_с$, м:

$$l_c = 1,05 \cdot L_в \cdot (n_в - 1), \quad (4.9)$$

де $L_в$ – відстань між вертикальними заземлювачами, (прийняти за $L_в$ = 3м);

$n_в$ – необхідна кількість вертикальних заземлювачів.

$$l_c = 1,05 \cdot 3 \cdot (16 - 1) \approx 48 \text{ м}$$

8) Визначається опір розтіканню струму горизонтального заземлювача (з'єднувальної стрічки) $R_г$, Ом:

$$R_г = \frac{\rho_{розр.г}}{2 \cdot \pi \cdot l_c} \cdot \ln \frac{2 \cdot l_c^2}{d_{см} \cdot h_г}, \quad (4.10)$$

де $d_{см}$ – еквівалентний діаметр смуги шириною b, $d_{см}$ =0,95b, b = 0,15 м;

$h_г$ – глибина закладання горизантальних заземлювачів (0,5 м);

l_c - довжина з'єднувальної стрічки горизонтального заземлювача l_c, м

$$R_г = \frac{220}{2 \cdot \pi \cdot 48} \cdot \ln \frac{2 \cdot 48^2}{0,95 \cdot 0,15 \cdot 0,5} = 8,1 \text{ Ом}$$

9) Визначається коефіцієнт використання горизонтального заземлювача $\eta_{с.}$ відповідно до необхідної кількості вертикальних заземлювачів $n_в$.

Коефіцієнт використання з'єднувальної смуги $\eta_с$=0,3 (табличне значення).

10) Розраховується результуючий опір заземлювального електроду з урахуванням з'єднувальної смуги:

$$R_{заг} = \frac{R_в \cdot R_г}{R_в \cdot \eta_с + R_г \cdot n_в \cdot \eta_в} \le R_д. \quad (4.18)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{заг}$ <4 Ом, а саме:

$$R_{заг} = \frac{18,5 \cdot 8,1}{18,5 \cdot 0,3 + 8,1 \cdot 16 \cdot 0,57} = 1,9 \le R_д$$

ВИСНОВКИ

В рамках дипломного проекту було реалізовано сервіс для обробки замовлень на купівлю обладнання від абонентів системи, розробка якого охопила використання технологій, які на сьогоднішній день користуються великим попитом.

Для досягнення поставлених цілей були вирішені наступні завдання:

– вивчені принципи роботи технології Spring Framework;

– досліджені та використані на практиці особливості роботи фрейворк JSF Primefaces;

– проведений порівняльний аналіз ORM Hibernate та MyBatis;

– проаналізований та застосований на практиці підхід проектування DDD;

– вивчені та використані основні принципи підходу проектування Scrum Agile;

– реалізоване формування та завантаження документів за допомогою технології FOP;

– спроектовано та розроблено сервіс;

– проведена інтеграція сервісу в систему VC Billing New;

– реалізовано модульне тестування для всіх частин програми;

– розроблені тест-кейси для перевірки основної функціональності додатку.

У майбутньому розвитку сервісу необхідно реалізувати зв'язок зі сторонніми системами перевізників через FTP сервер для автоматизації процесу доставки товарів.

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи, описано, які заходи потрібно зробити для того, щоб дане приміщення

відповідало необхідним нормам і було комфортним і безпечним для робітника. Приведені рекомендації щодо організації робочого місця, а також інформація щодо пожежної та електробезпеки. Виконано розрахунок захисного заземлення та освітлення. Наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Красов, А.В. Теория информационных процессов и систем [Текст] : учеб. / А. В. Красов, С.Е. Душин СПб.: СПбГЭТУ «ЛЭТИ», 2006. – 126с.

2. Beynon-Davies P,. Information Systems: an introduction to informatics in Organisations [Text]. / P. Beynon-Davies: Palgrave, Basingstoke, 2002 – 788 p.

3. Evans E., Domain-Driven Design: Tackling Complexity in the Heart of Software [Text]. / E. Evans: Addison-Wesley, 2003 – 199 p.

4. Светлов, Н.М. Информационные технологии управления проектами [Текст]: учеб. / Н.М Светлов, Г.Н. Светлова. – М.: ФГОУ ВПО РГАУ–МСХА им. К.А. Тимирязева, 2007. – 207с.

5. Грекул, В.И. Основы информационных технологий. Методические основы управления ИТ-проектами [Текст] : учеб. В.И. Грекул, Н.Л. Коровкина; Издательство: ИНТУИТ, БИНОМ, 2011. – 102 с.

6. Кон, М. Scrum: гибкая разработка ПО [Текст]:учеб / М. Кон. – М.: Вильямс, 2011. — 576с.

7. Книберг, Х. [Текст]: учеб / Х. Книберг —М: Вильямс, 2007. – 140с.

8. Методичні вказівки до виконання і захисту дипломного проекту (роботи) бакалавра спеціальностей 122 "Комп’ютерні науки та інформаційні технології", 123 "Комп’ютерна інженерія", 125 "Кібербезпека" (за напрямами 6.050101 "Комп’ютерні науки", 6.050102 "Комп’ютерна інженерія") для здобувачів вищої освіти денної і заочної форм навчання / Уклад.: Скарга-Бандурова І.С., Барбарук В.М., Кардашук В.С. – Сєродонецьк: СНУ ім. В. Даля, 2018. – 60 с.

9. Методичні вказівки до виконання розділу дипломного проекту (роботи) бакалавра "Охорона праці та безпека в надзвичайних ситуаціях"

(для студентів денної та заочної форм навчання за спеціальностями 122 "Комп’ютерні науки та інформаційні технології", 123 "Комп’ютерна інженерія", 125 "Кібербезпека" (за напрямами 6.050101 "Комп’ютерні науки", 6.050102 "Комп’ютерна інженерія", 6.170101 "Безпека інформаційних і комунікаційних систем")) / Уклад.: Критська Я.О. – Під ред. Скарги-Бандурової І.С. – Сєвєродонецьк: вид-во СНУ ім. В. Даля, 2017. – 71 с.

10. ГОСТ 12.1.044-89 ССБТ. Пожежовибухонебезпека речовин і матеріалів. Номенклатура показників і методи їх визначенняя

11. ДБН В.2.5-28:2015 Природнє і штучне освітлення

12. ДСанПіН 3.3.2.007-98 Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин

13. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку

14. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих

15. НПАОП 0.00-4.12-05 Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці

16. НПАОП 0.00-4.15-98 Про розробку інструкцій з охорони праці

17. НПАОП 0.00-6.03-93 Порядок опрацювання та затвердження власником нормативних актів про охорону праці

18. НАПБ Б.03.002-2007 Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою

19. НПАОП 40.1-1.01-97 Правила безопасной эксплуатации электроустановок

20. НПАОП 40.1-1.32-01 Правила устройства электроустановок. Электрооборудование специальных установок

21. ДСН 3.3.6.039-99 Санітарні норми виробничої загальної та локальної вібрації

22. ДСТУ ГОСТ 12.1.012-90 ССБТ. Вібраційна безпека. Загальні вимоги

23. ДБН В.2.5-67:2013 Опалення, вентиляція та кондиціонування

24. ГОСТ 12.1.006-84 ССБТ. Електромагнітні поля радіочастот. Загальні вимоги безпеки. Допустимі рівні на робочих місцях і вимоги до проведення контролю

25. ГОСТ 12.1.030-81 ССБТ. Електробезпечність. Захисне заземлення. Занулення

26. НПАОП 0.00-1.28-10 Правила охорони праці під час експлуатації електронно- обчислювальних машин

Додаток А

Комп’ютерна презентація

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

Дипломна робота

РОЗРОБЛЕННЯ СЕРВІСУ ДЛЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ОБЛІКУ ТА ТАРИФІКАЦІЇ НАДАНИХ ПОСЛУГ

Здобувач вищої освіти:
студент гр. КІ-14з
Газашвілі Д.В.

Керівник:
доц. Барбарук В.М.

Сєвєродонецьк 2018

Рисунок А.1 – Слайд №1

Постановка задачи

Целью дипломной работы является разработка сервиса для биллинговой системы VC Billing New.

Система VC Billing New работает с абонентами мобильной связи.

Сервис предназначен для обработки заказов на покупку оборудования от абонентов системы.

Оператор сервиса имеет возможность контролировать ЖЦ заявки и формировать необходимые статистические отчеты. Так же оператор формирует перевозки курьера после согласования времени доставки с клиентом.

Рисунок А.2 – Слайд №2

Для достижения поставленной цели необходимо:

1. Изучить следующие подходы и технологии создания ПО:

- принципы работы технологии Spring Framework;
- особенности работы JSF Primefaces;
- ORM Hibernate и MyBatis;
- шаблон проектирования DDD;
- подход к управлению проектами Scrum Agile.

2. На основе изученного разработать систему для обработки заказов на покупку оборудования от абонентов системы.

3. Реализовать модульное тестирование (Unit tests) для всех частей программы.

4. Провести интеграцию сервиса в систему VC Billing New.

Рисунок А.3 – Слайд №3

Актуальность разработки

Разработка сервиса была вызвана необходимостью автоматизации обработки заказов в системе VC Billing New и генерации отчетов.

Данный сервис является web-приложением. Web-приложение – это прикладное программное обеспечение, логика которого распределена между серверами клиентом, а обмен информацией происходит по сети.

Актуальность исследований в области вопросов построения web-приложений обусловлена тем, что данный вид программного обеспечения:

- перспективен, как инструмент электронной коммерции;
- предоставляет широкие возможности социального взаимодействия.

Web-приложение должно соответствовать таким требованиям:

1. выполняться независимо от операционной системы клиента (при этом налагается требование кроссбраузерности);
2. представлять собой распределенную информационную систему, поддерживая большое число одновременных обращений;
3. обеспечивать высокую скорость обработки информации;
4. гарантировать надежность вычислений и т.д.

Рисунок А.4 – Слайд №4

Актуальность технологий

Современные web-приложения можно охарактеризовать как сложные программные комплексы. Основными требованиями в разработке являются:

- скорость;
- эффективное использование ресурсов.

Данные требования обеспечиваются благодаря современными технологиями разработки, которые позволяют разработчику сфокусироваться на основной логике приложения (такие языки и технологии как, Java, ORM, Oracle PL/SQL и разработанные на их основе фреймворки).

Не маловажным пунктом в создании ПО является выбранная методология разработки. На сегодняшний день широко используется совместное применение методологии Domain Driven Design с подходом управления проектами Scrum Agile.

Рисунок А.5 – Слайд №5

Подходы к проектированию и технологии, используемые в работе

Domain Driven Design

Domain-driven design (DDD) является доменно-ориентированной методологией проектирования. Идеология DDD основана на предположении, что существенные изменения в предметной области происходят значительно реже, чем изменения требований к ПО. Таким образом, архитектура системы, основанная на модели предметной области, будет более стабильной. Так была разработана пирамида, которую очень удобно использовать при анализе нового функционала или при начальной разработке системы. На рисунке изображена пирамида DDD, которая состоит из следующих блоков:

1. BOM - бизнес-обьекты системы;
2. DTO - модель представления таблиц базы данных;
3. Translator - класс, который осуществляет трансляцию BOM в DTO и наоборот;
4. Domain Service - реализация бизнес - методов (оперирует BOM);
5. Connector - реализация методов работы с базой данных;
6. UI - реализация графического интерфейса.

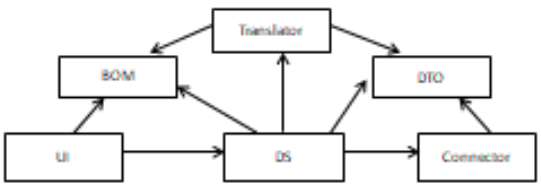


Рисунок 1 – Пирамида DDD

Рисунок А.6 – Слайд №6

Подходы к проектированию и технологии, используемые в работе

Преимущества подхода Domain-driven design

При подходе DDD используется многослойная архитектура.

Общие принципы проектирования с использованием многослойной архитектуры:

- четкая определенность. Разделение функциональности между слоями очень четкое.

Верхние слои, такие как слой представления, посылают команды нижним слоям, таким как бизнес-слой и слой данных, и могут реагировать на события, возникающие в этих слоях, обеспечивая возможность передачи данных между слоями вверх и вниз;

- возможность повторного использования;
- отсутствие зависимостей между нижними и верхними слоями обеспечивает потенциальную возможность их повторного использования в других сценариях.

При разработке системы была получена диаграмма бизнес-объектов (рис.2).

Рисунок 2 – Диаграмма BOM-объектов

Рисунок А.7 – Слайд №7

Подходы к проектированию и технологии, используемые в работе

Подходы управления проектами при создании ПО

- Управление проектами - это область деятельности, в ходе которой определяются и достигаются четкие цели проекта при балансировании между объемом работ, ресурсами (такими как деньги, труд материалы, энергия, пространство и др.), качеством и рисками.
- Методология Agile была разработана для управления ИТ-проектами в области разработки программного обеспечения и базируется на принципах, противоположных классическому Waterfall-подходу. Сравнение методологий показано на рисунке 3.

Основные принципы методологии Agile:

- итеративный цикл разработки составляет 1-6 недель;
- фокус на продукте, а не проектной документации;
- фокус на коммуникациях в команде;
- полное техническое задание не пишется на весь проект, а только на будущий спринт (релиз);
- гибкий процесс внесения изменений в проект;
- команда разработчиков должна располагаться в одной комнате вместе с владельцем продукта.

Рисунок 3 – Сравнение методологий

Рисунок А.8 – Слайд №8

Подходы к проектированию и технологии, используемые в работе

Использование подхода управления проектами Scrum Agile

- Agile Scrum – это набор принципов, на которых строится процесс разработки, он позволяет в жестко фиксированные и небольшие по времени итерации, называемые спринт (sprints), предоставлять конечному пользователю работающее ПО с новыми возможностями, для которых определен наибольший приоритет.
- При написании дипломной работы принципы Agile строго выполнялись.

Рисунок А.9 – Слайд №9

Подходы к проектированию и технологии, используемые в работе

Особенности языка программирования Java

Объектно-ориентированное программирование - это метод программирования, в центре внимания которого находятся данные (то есть объекты) и средства доступа к ним.

Язык Java в первую очередь предназначен для создания программ, которые надежно работают на любых платформах и под любой нагрузкой.

В Java пришлось отказаться от:

- перегрузки операторов (но перегрузка методов в Java осталась);
- множественного наследования;
- указателей;
- автоматического расширенного приведения типов.

Разработка приложений на Java приводит к получению программного обеспечения, которое:

- является переносным на различные архитектуры, операционные системы и графические интерфейсы;
- безопасно;
- высокопроизводительно.

Благодаря Java работа по разработке программного обеспечения значительно упрощается, все старания направлены на достижение конечной цели: вовремя получить передовой продукт.

10

Рисунок А.10 – Слайд №10

Особенности использования Spring Framework

Spring Framework - это универсальный фреймворк с открытым исходным кодом для Java-платформы. Spring Framework, обеспечивает полную поддержку совместной работы частей Java приложения. Он обеспечивает решение многих задач, с которыми сталкиваются Java-разработчики, которые хотят создать информационную систему, основанную на платформе Java.

Spring упрощает работу с JDBC, ORM frameworks, Web services.

К основным преимуществам Spring относятся:

- простота;
- удобство тестирования;
- использование Spring положительно сказывается на дизайне приложения и простоте кода.

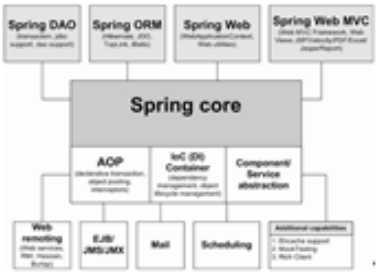


Рисунок 4 – Архитектура Spring

Рисунок А.11 – Слайд №11

Характеристики Object-relational mapping

- Object-relational mapping (ORM) - это технология программирования, которая позволяет преобразовывать несовместимые типы моделей в ООП, в частности, между хранилищем данных и объектами программирования (напр. фреймверки Hibernate, MyBatis).
- Использование ORM в проекте лишает разработчика необходимости работы с SQL и написания большого количества кода, часто однообразного и подверженного ошибкам.

Рисунок 5 – Схема работы ORM

Рисунок А.12 – Слайд №12

Подходы к проектированию и технологии, используемые в работе

Обзор шаблона проектирования Test-driven development

- При написании дипломной работы возникла необходимость покрытия функционала тестами для проверки работоспособности отдельных модулей. Для этого использовался шаблон программирования Test-driven development (TDD).
- Разработка через тестирование - это техника разработки программного обеспечения, основанная на повторении очень коротких циклов разработки: сначала пишется тест, покрывающий желаемое изменение, затем пишется код, который позволит пройти тест, и под конец проводится рефакторинг нового кода к соответствующим стандартам.

Рисунок 6– Схема работы TDD

Рисунок А.13 – Слайд №13

Разработка сервиса WebShop

Разработка модели БД

При написании дипломного проекта была спроектирована схема базы данных для сервиса, и сьинтегрирована с уже существующей базой данных проекта VC Billing New. Использовалась СУБД Oracle 10g.

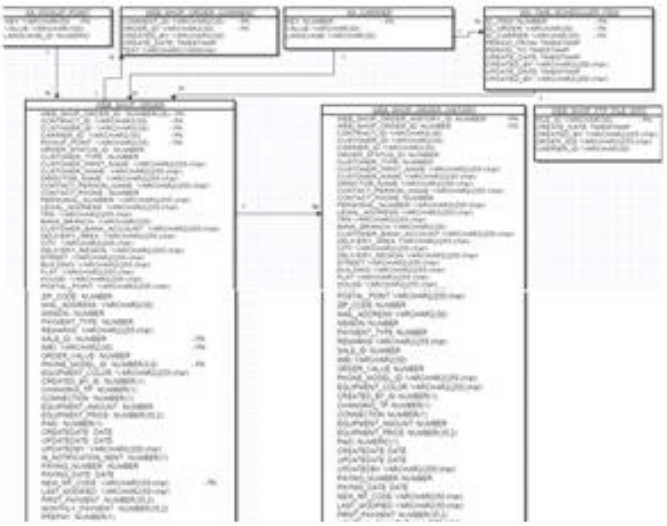


Рисунок 7 – Локальная схема БД сервиса

Рисунок А.14 – Слайд №14

Разработка сервиса WebShop

Схема БД проекта VC Billing New

На рисунке изображена часть схемы БД проекта VC Billing New.

- Таблица PHONE_MODEL - содержит информацию о товаре, который продается в интернет магазине
- Таблица CCONTACT_ALL - содержит информацию об абонентах данной биллинговой системы, которые являются потенциальными клиентами Интернет магазина.
- Таблицы SALE, SALE_TYPE, SALE_PROTOTYPE - содержат информацию о типах продажи товаров (акции, рассрочки).

Рисунок 8 – Часть схемы БД проекта VC Billing New

Рисунок А.15 – Слайд №15

Разработка сервиса WebShop

Дизайн сервиса WebShop

- Диаграмма переходов между страницами модуля изображена на рисунке 9.

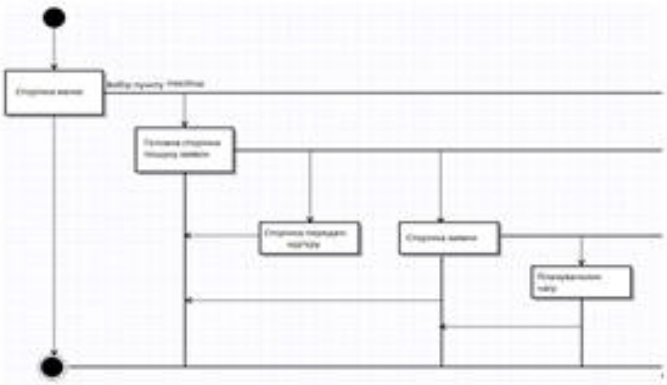


Рисунок 9 – Диаграмма переходов между страницами сервиса

Рисунок А.16 – Слайд №16

Иллюстрация работы сервиса

«Главная страница»

Сервис является частью биллинговой системы VC Billing New, поэтому для начала работы необходимо прежде всего, пройти процедуру авторизации. При успешной аутентификации пользователь выбирает в меню «Интернет магазин» и попадает на страницу поиска заказов.

Рисунок 10 – Страница поиска заявок

При поиске заказов можно использовать фильтр по необходимым полям:
- № заказа;- название клиента;- способ доставки;- тип продажи;- статус заказа;- тип оплаты;- MSISDN абонента
- тип клиента;
- сотрудник, создал заказ.

Также есть кнопка «Сформировать отчет», которая становится активной после выбора необходимых для отчета заказов. Приме

17

Рисунок А.17 – Слайд №17

Иллюстрация работы сервиса

Страница отображения заявки

- При нажатии на номер заказа – оператор попадает на страницу заявки. На рисунке изображена заявка в статусе «Готовая к отгрузке». На странице заявки отражена информация о заказе, а также есть возможность перевода заявки в другой статус или печать необходимых документов.

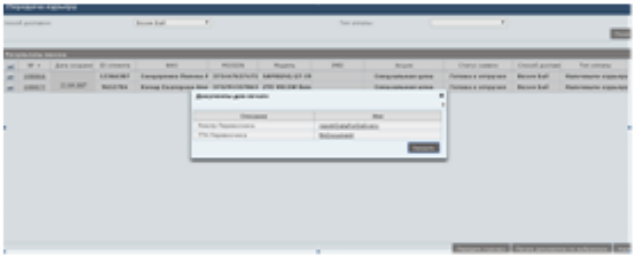
Рисунок А.18 – Слайд №18

Иллюстрация работы сервиса

Страница «Передачи курьеру»

- У заказов, которые находятся в статусе «Готова к отгрузке», уже уточнены все детали с клиентом, и для успешной доставки остается только выбрать тип перевозчика, и перевести заявку в статус «Переданна курьеру». Для этого, с главной страницы поиска заявок выбираем фильтр по полю «Статус», отмечаем нужные заявки и нажимаем на кнопку «Передать курьеру».

- После этого, оператор попадает на страницу «Передачи курьеру», на которой можно осуществить фильтр заявок по полю «Превозчик» и «Способ оплаты», и передать заявки перевозчику, изменив их статус, а также напечатать необходимые документы.



19

Рисунок А.19 – Слайд №19

Иллюстрация работы сервиса

Страница «Планировщик времени»

- Со страницы заявки можно перейти в планировщик времени для перевозчиков. Для этого необходимо нажать на ссылку «Перейти в планировщик времени». Функционал выполнен с помощью компонента Primefaces p: schedule. Интерфейс функционала изображен на рисунке 11.

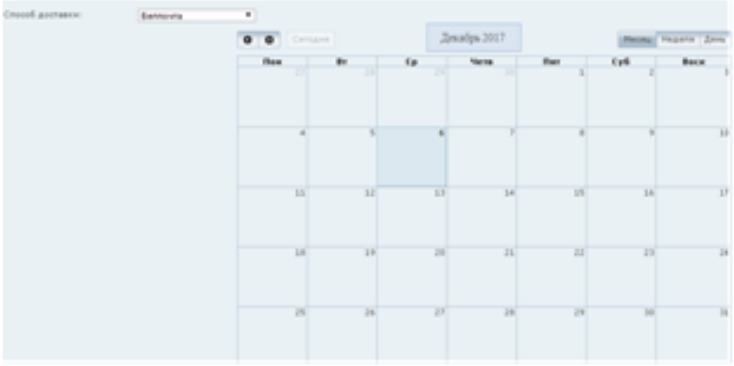


Рисунок 11 – Планировщик времени

Рисунок А.20 – Слайд №20

Тестирование работы сервиса WebShop

- Для тестирования функциональности сервиса WebShop были созданы тест-кейсы, которые покрывают основную часть работоспособности системы. В качестве программного обеспечения, использовалась программа TestLink. TestLink - это система управления тестами с веб-интерфейсом.
- Например, прежде всего, нужно проверить приходят ли уведомления оператору когда новая заявка попадает в систему.

Рисунок А.21 – Слайд №21

Охрана труда

В результате проведенной работы был сделан анализ условий труда, вредных и опасных факторов, с которыми сталкивается работник.

Были определены параметры и определенные характеристики помещения для работы над предложенным проектом, описано, какие меры нужно предпринять для того, чтобы данное помещение соответствовало необходимым нормам и было комфортным и безопасным для работающего.

Приведены рекомендации по организации рабочего места, а также важная информацию о пожарной и электробезопасности.

Были приведены размеры помещения и приведены значения температуры, влажности и подвижности воздуха, необходимое количество и мощность ламп и другие параметры, значение которых влияет на условия труда работника, а также - приведены инструкции по охране труда, технике безопасности при работе на компьютере.

Рисунок А.22 – Слайд №22

Выводы

В рамках дипломного проекта был разработан сервис для биллинговой системы VC Billing New.

Для достижения поставленных целей были решены следующие задачи:

- изучены принципы работы технологии Spring Framework;
- исследованы и использованы на практике особенности работы фреймворк JSF Primefaces;
- проведен сравнительный анализ ORM Hibernate и MyBatis;
- проанализирован и применен на практике подход проектирования DDD;
- изучены и использованы основные принципы подхода проектирования Scrum Agile;
- реализовано формирование и загрузка документов с помощью технологии FOP;
- спроектирован и разработан сервис;
- проведена интеграция сервиса в систему VC Billing New;
- реализовано модульное тестирование (Unit tests) для всех частей программы;
- разработаны тест-кейсы для проверки основной функциональности приложения;
- проанализировано состояние охраны труда.

В качестве дальнейшего развития сервиса планируется реализовать связь со сторонними системами перевозчиков через FTP сервер для автоматизации процесса доставки товаров.

23

Рисунок А.23 – Слайд №23