МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра Інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д.т.н., проф.,
_____Ю.П. Кондратенко
«____»_____202_ року

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

СИСТЕМА CRM ДЛЯ КОНТРОЛЮ ЗА РОБОТОЮ ПЕРСОНАЛУ СТО ЗА РАХУНОК ОПТИМІЗАЦІЇ ПРОЦЕСІВ ВЗАЄМОДІЇ З КЛІЄНТАМИ

Спеціальність 122 – «Комп'ютерні науки»

122-МКР-601.21610213

Виконав: студент 6 курсу, групи 601
_____С.І. Лістов_____
(підпис, ініціали та прізвище)
«___» _____2022р

Керівник к.ф-м.н., доцент
_____І.В. Кулаковська
«__» лютий 2022 р

Миколаїв – 2022

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ 3
ВСТУП 5
1 АНАЛІЗ СУЧАСНОГО СТАНУ ОБРАНОЇ ЗАДАЧІ 8
1.1 Огляд та аналіз аналогів автоматизованої інформаційної системи керування СТО 9
1.2 CRM-системи для контролю роботи СТО 17
1.3 Постановка задачі 20
Висновки до розділу 1 21
2 ПІДХОДИ ТА ТЕХНОЛОГІЇ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ 22
2.1 Опис підходів та технологій 23
2.2 Огляд та вибір платформ й підходів 30
2.3 Вибір мови програмування 31
Висновки до розділу 2 38
3 ПРОЕКТУВАННЯ СИСТЕМИ ТА ДИЗАЙНУ 39
3.1 Функціональна/процесна модель систем 40
3.2 Проектування інформаційного забезпечення 44
Висновки до розділу 3 64
4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА РОЗРОБКА ДОКУМЕНТАЦІЇ 65
4.1 Вибір технологій розробки програмного забезпечення 65
4.2 Вибір середовища розробки 78
4.3 Створення інтерфейсу застосунку 80
4.4 Інструкція користувача 85
Висновки до розділу 4 87
ВИСНОВКИ 102
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 104
ДОДАТОК А 108
ДОДАТОК Б 110

ПЕРЕЛІК СКОРОЧЕНЬ

ДСТУ	державний стандарт України
ІС	інформаційна система
МКР	магістерська кваліфікаційна робота
ПЗ	програмне забезпечення
CSS	Cascading Style Sheets
ECTS	European Community Course Credit Transfer System
API	Application Programming Interface
HTML	HyperText Markup Language
IDE	Integrated Drive Electronics
MVC	Model View Controller
ORM	Object-Relational Mapping
SDK	Software Development Kit
UI	User Interface

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему:

«СИСТЕМА CRM ДЛЯ КОНТРОЛЮ ЗА РОБОТОЮ ПЕРСОНАЛУ СТО ЗА РАХУНОК ОПТИМІЗАЦІЇ ПРОЦЕСІВ ВЗАЄМОДІЇ З КЛІЄНТАМИ»

Спеціальність 122 – «Комп'ютерні науки»

122 – МКР – 601.1610213

Студент _____ Лістов С.І.
«__» _____ 20___ р.

Консультант _____ Кулаковська І.В.
к.ф-м.н., доцент
«__» _____ 20___ р.

Миколаїв – 2022

ВСТУП

Однією з основних завдань забезпечення конкурентоспроможності підприємства, працюючою у сфері надання послуг населенню, є просуваннясучасної клієнто-орієнтованої стратегії.

Найбільш яскравим прикладом є сфера технічного обслуговування та ремонт автотранспортних засобів.

У процесі оформлення замовлення ремонт транспортного засобу необхідно заповнити велику кількість різноманітних документів, на перевірку яких потрібно багато часу.

Крім якісного надання послуг, такі клієнти сподіваються отримати від компанії знижки на послуги, у тому числі щодо акцій, про проведення яких менеджмент компанії часто їх не повідомляє заздалегідь. Подібні факти можуть викликати невдоволення клієнтів, що, у свою чергу, призводить до їх відпливу у конкуруючі фірми [1].

Для дослідження підвищення конкурентоспроможності прийнято рішення запровадити автоматизовану інформаційну систему, розроблену на основі сучасних Web технологій, яка містить елементи Customer Relationship Management (CRM) – систем.

CRM системи можуть допомогти збільшити продажі, їх продуктивність продажів та точність прогнозу продажів на 55%. Завдяки CRM, що допомагає легко визначити, на якому етапі вирви знаходяться клієнти, маркетологи можуть розставити пріоритети всередині взаємодії з клієнтами. Неактивні клієнти залучаються до циклу продажів у відповідний для клієнта та компанії момент.

Кожен етап життєвого циклу клієнта можна відслідковувати та вживати заходів, забезпечуючи при цьому стратегічне узгодження в рамках усього бізнесу.

Роль CRM-систем можна коротко охарактеризувати такою фразою: це ефективне використання всіх каналів комунікацій з клієнтом для забезпечення його лояльності до компанії.

Наявність солідної бази лояльних клієнтів є основним і чи не єдиним фактором стійкості та процвітання підприємств, працюючих у сфері надання послуг населенню[2].

Таким чином, **актуальність теми** магістерської кваліфікаційної роботи обумовлена необхідністю просування ідеї клієнто-орієнтованої стратегії з розробкою та подальшим використанням CRM-системи.

Об'єктом дослідження випускної дипломної роботи є бізнес-процес оформлення замовлень на ремонт транспортних засобів у CRM-системі.

Предмет дослідження – автоматизація бізнес-процесу управління замовленнями клієнтів на ремонт транспортних засобів за рахунок використання CRM-системи.

Метою роботи є розробка автоматизованої інформаційної системи управління замовленнями клієнтів з елементами CRM-системи.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- аналіз існуючих технологій розробки CRM-систем;
- дослідження відомих підходів та технологій реалізації CRM-систем;
- вибрати методології та технології проектування інформаційної системи;
- розробити логічну модель системи;
- розробити та проаналізувати моделі бізнес-процесу.
- створення діючого прототипу за стосунку з доступним і простим інтерфейсом.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ОБРАНОЇ ЗАДАЧІ

У світі впровадження інноваційних систем має велике значення як для загального зростання конкурентоспроможності підприємств, а й для формування ефективних клієнтських відносин, що забезпечують, у свою чергу, прибутковість компанії.

Сьогодні споживач став фокусом усіх зусиль виробників, а їх задоволеність відносинами з постачальником товарів та послуг – ключовим фактором успіху компанії. Боротьба за клієнта стає дедалі більше запеклої. Водночас деякі керівники досі вважають, що місце інноваційних технологій у боротьбі за клієнта – у розряді діяльності допоміжних служб. Однак, як показує досвід, ті керівники, які розглядають передові технології як важливий інструмент управління бізнес, швидше приходять до успіху. Для компаній, які бажають бути успішними, настав момент, коли не можна далі відкладати приведення в порядок власної клієнтської бази. При цьому не має значення, чи працює компанія на корпоративному чи роздрібному ринку. Будь-якому керівнику в поточної ситуації необхідно приймати рішення, які потребують гарного знання клієнтів[3].

Основна роль у розвитку компаній в даний час відводиться менеджери з продажу. Але просто набрати менеджерів з продажу недостатньо, навіть якщо кожен з них навчений – необхідні довгострокові відносини між клієнтами і компанією принесуть далеко не всі.

Рішення цієї проблеми вбачається у введенні в менеджмент компанії нові інноваційні продукти, такі як системи CRM. Така система – це, бізнес-стратегія залучення та управління клієнтами, спрямована на оптимізацію їх вартості у довгостроковій перспективі. CRM передбачає наявність в організації філософії та культури, орієнтованої на клієнта, спрямовані на ефективність роботи у сфері маркетингу, збуту та післяпродажне обслуговування. CRM-додатки дозволяють ефективно управляти

відносинами з клієнтами за умови, що підприємство має правильні цілі, стратегію та культуру. CRM – це підхід до управління модель, якої ставить клієнта в центр бізнес-процесів і практик компанії[4].

Виникає необхідність провести аналіз предметної галузі та виявити основні недоліки існуючих бізнес-процесів управління обслуговування клієнтів. За рахунок чого визначити шляхи удосконалення бізнес-процессів за рахунок впровадження автоматизованої інформаційної системи управління клієнтами із CRM-системи.

1.1 Огляд та аналіз аналогів автоматизованої інформаційної системи керування СТО

За своїми функціональними особливостями CRM-системи керування та контролю роботи станції технічного обслуговування відноситься до програмного забезпечення операційних служб сервісних центрів, або SaaS (Software As A Service).

SaaS — це, одна з форм хмарних обчислень, модель обслуговування, за якої підписникам надається готове прикладне програмне забезпечення, яке повністю обслуговує провайдер. Постачальник у цій моделі самостійно керує програмою, надаючи замовникам доступ до функцій з клієнтських пристроїв, як правило через мобільний застосунок або веб-версію.

Основна перевага моделі SaaS для споживача послуги полягає у відсутності витрат, пов'язаних із встановленням, оновленням та підтримкою працездатності обладнання та працюючого на ньому програмного забезпечення.

Розглянемо відомі аналоги інформаційних систем для контролю за роботою персоналу СТО за рахунок оптимізації процесів взаємодії з клієнтами.

Перший програмний продукт, який ми розглянемо — це, iDirector Авто — онлайн CRM для автосервісів та автосалонів.

Під час аналізу системи iDirector Авто були виявлені наступні можливості:

Замовлення-наряд. Веде замовлення, виставляє знижки на роботи та запчастини, калькуляція вартості, зміна нормогодин, роздруківка замовлень, актів, рахунків тощо), управління робочими місцями (див. рис. 1.1).

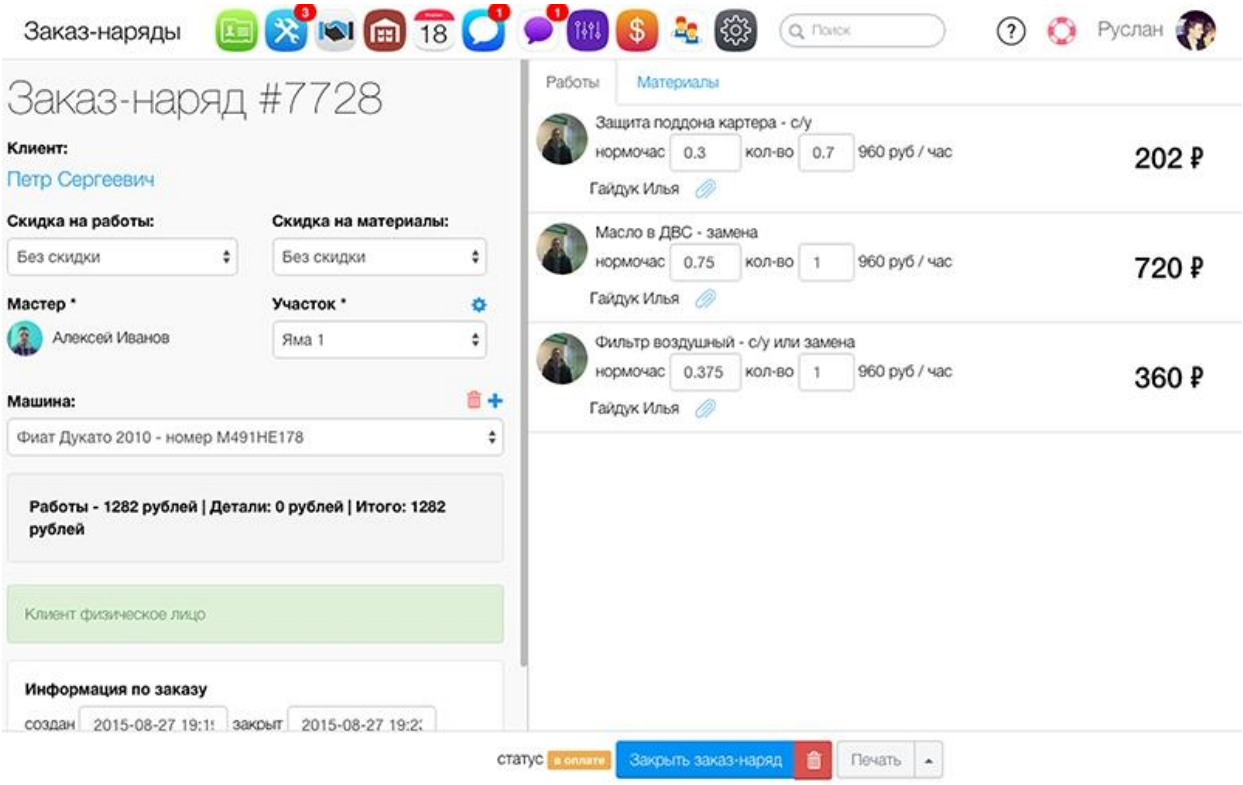


Рис. 1.1. Користувальницький інтерфейс замовлення-наряду iDirector

Склад. Ведення складського обліку, будь-яка кількість складів, постачальників. Автоматичний зв'язок із бухгалтерією. Широкий асортимент звітів за складом;

Календар. Можливість легко та швидко записувати клієнтів на конкретний час, відображення складу зміни по днях, планування робітника графіка, нотатки;

Інтеграція із сайтом. Можливість легко і швидко вбудувати готову форму запису клієнта прямо на ваш сайт, обробка вхідних заявок з сайту підприємства через CRM iDirector; проста інтеграція.

Клієнт має можливість завантажити необхідні документи (рахунки-фактури, акти, наряди на виконання робіт), отримувати доступ до історії ремонтів та іншого необхідної інформації для роботи.

Через особистий кабінет клієнт також зможе отримувати поради від своїх майстрів і планувати свої візити до свого обслуговування заздалегідь. Якщо підключити додаткові функції, то клієнт зможе спостерігати за процесом ремонту або переглядати детальні дані діагностика свого автомобіля в режимі реального часу.

Наступною системою для аналізу було обрано програмний продукт «1С:Підприємство 8. Автосервіс». для автоматизації оперативного та управлінського обліку дрібних автопідприємства, основним видом діяльності яких є забезпечення послуги з ремонту та технічного обслуговування автомобілів (див. рис. 1.2).

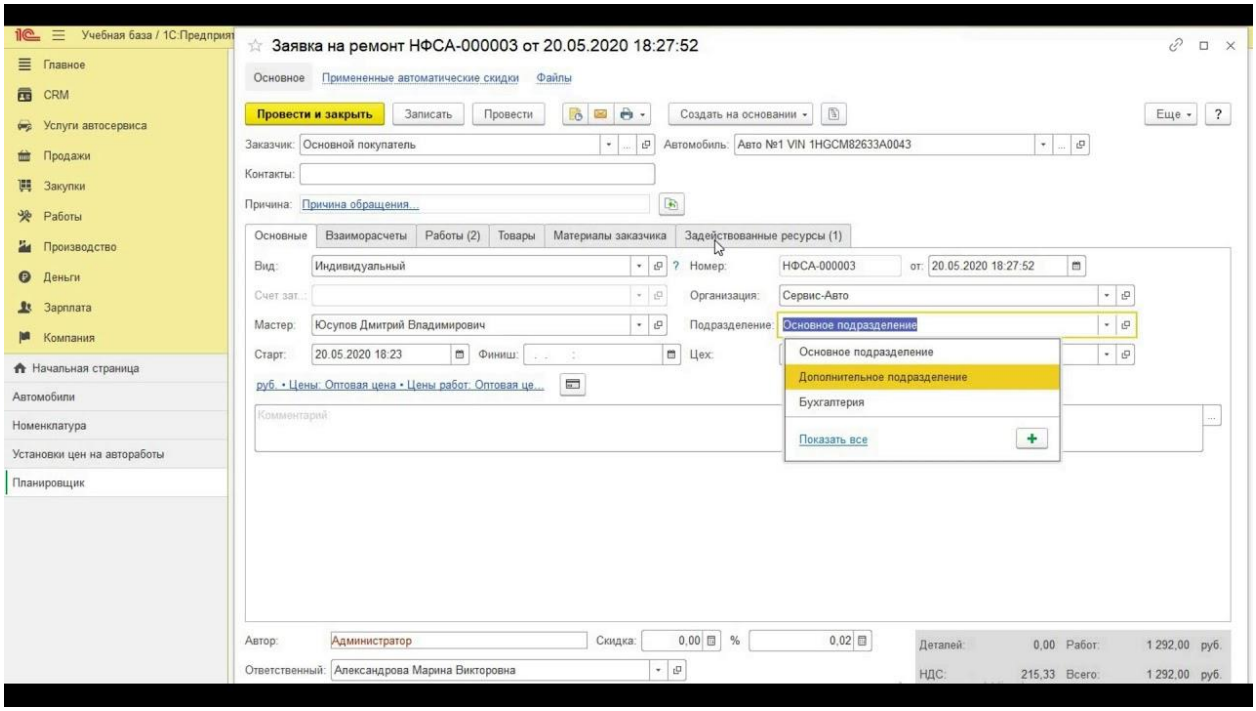


Рис. 1.2. Користувальницький інтерфейс замовлення-наряду iDirector

Основним функціоналом обраної для аналізу CRM-системи програмного продукту представлений наступними можливостями:

− підтримання клієнтської бази;

− реєстрація та зберігання контактної інформації замовника;

− оформлення та зберігання списку контактних осіб контрагентів та їх Контактна інформація;

− фіксація всіх контактів з клієнтами: вхідні та вихідні дзвінки, листи, зустрічі тощо;

− попередній запис на ремонт.

Продукт розроблено на основі типової конфігурації «Менеджмент невеликих фірм» програмної системи «1С:Підприємство 8» при збереженні всі основні особливості та механізмів цього типового рішення[5].

В якості ще одного сервісу було прийнято рішення розглянути можливості та запропонований функціонал системи, розробленою компанією AutoSoft.

Програма для автосервісів АвтоПідприємство позиціонується своїми розробниками як потужний інструмент, який значно підвищити ефективність роботи автосервісу, привести в порядок документообіг, складський облік, підвищення швидкості роботи працівників і знизити ймовірність своїх помилок, підвищити лояльність клієнтів (див. рис. 1.3).

Рис. 1.3. Користувальницький інтерфейс AutoSoft АвтоПідприємство

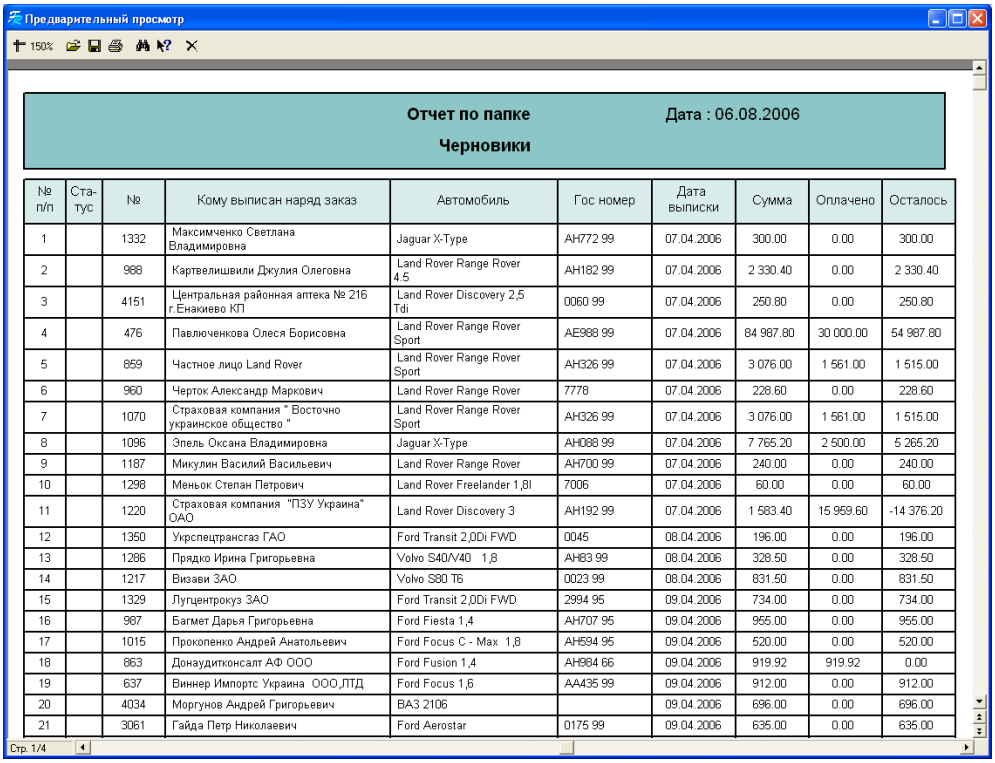
Слід відзначити, що оформлення всіх документів у програмі АвтоПідприємство виконується швидко та зручно [6]. Використання довідників дозволяє заощадити час користувача або власника підприємства, усуваючи необхідність двічі вводити повторювану інформацію. У деяких операціях довідники викликаються незалежно, запобігаючи введення неправильної інформації. Наприклад, при оформленні квитанції в потрібний момент викликається довідник «Структура складу», і сервіс пропонує вказати, де на складі буде розміщено товар. Інакше про це можна просто забути.

Програма містить довідник по нормам трудомісткості - в поставку програми входить близько 2 000 000 нормативів часу, а також можна створити власну базу даних за стандартами подібної моделі.

Не менш важливою особливістю даного сервісу є заповнення наряд-заказа з використанням європейських норм щодо конкретного автомобіля, а

також власних нормативних баз та створення й подальше ведення власних нормативних баз

Установка кратності, коефіцієнт складності, кваліфікація роботи, вартість нормо-часу, кожного виконавця, оподаткування, знижки як для роботи в окремості, так і для всіх клієнтів (див. рис 1.4).



№ п/п	Ста-тус	№	Кому выписан наряд заказ	Автомобиль	Гос номер	Дата выписки	Сумма	Оплачено	Осталось
1		1332	Максимченко Светлана Владимировна	Jaguar X-Type	АН772 99	07.04.2006	300.00	0.00	300.00
2		988	Картвелишвили Джулия Олеговна	Land Rover Range Rover 4.5	АН182 99	07.04.2006	2 330.40	0.00	2 330.40
3		4151	Центральная районная аптека № 216 г.Енакиево КП	Land Rover Discovery 2,5 Tdi	0060 99	07.04.2006	250.80	0.00	250.80
4		476	Павлюченкова Олеся Борисовна	Land Rover Range Rover Sport	АЕ988 99	07.04.2006	84 987.80	30 000.00	54 987.80
5		859	Частное лицо Land Rover	Land Rover Range Rover Sport	АН326 99	07.04.2006	3 076.00	1 561.00	1 515.00
6		960	Черток Александр Маркович	Land Rover Range Rover	7778	07.04.2006	228.60	0.00	228.60
7		1070	Страховая компания " Восточно украинское общество "	Land Rover Range Rover Sport	АН326 99	07.04.2006	3 076.00	1 561.00	1 515.00
8		1096	Эпель Оксана Владимировна	Jaguar X-Type	АН088 99	07.04.2006	7 765.20	2 500.00	5 265.20
9		1187	Микулин Василий Васильевич	Land Rover Range Rover	АН700 99	07.04.2006	240.00	0.00	240.00
10		1298	Меньок Степан Петрович	Land Rover Freelander 1,8I	7006	07.04.2006	60.00	0.00	60.00
11		1220	Страховая компания "ПЗУ Украина" ОАО	Land Rover Discovery 3	АН192 99	07.04.2006	1 583.40	15 959.60	-14 376.20
12		1350	Укрспецтрансгаз ГАО	Ford Transit 2,0Di FWD	0045	08.04.2006	196.00	0.00	196.00
13		1286	Прядко Ирина Григорьевна	Volvo S40/V40 1,8	АН83 99	08.04.2006	328.50	0.00	328.50
14		1217	Визави ЗАО	Volvo S80 T6	0023 99	08.04.2006	831.50	0.00	831.50
15		1329	Лугцентрокуз ЗАО	Ford Transit 2,0Di FWD	2994 95	09.04.2006	734.00	0.00	734.00
16		987	Багмет Дарья Григорьевна	Ford Fiesta 1,4	АН707 95	09.04.2006	955.00	0.00	955.00
17		1015	Прокопенко Андрей Анатольевич	Ford Focus C - Max 1,8	АН594 95	09.04.2006	520.00	0.00	520.00
18		863	Донаудитконсалт АФ ООО	Ford Fusion 1,4	АН984 66	09.04.2006	919.92	919.92	0.00
19		637	Виннер Импортс Украина ООО,ЛТД	Ford Focus 1,6	АА435 99	09.04.2006	912.00	0.00	912.00
20		4034	Моргунов Андрей Григорьевич	ВАЗ 2106		09.04.2006	696.00	0.00	696.00
21		3061	Гайда Петр Николаевич	Ford Aerostar	0175 99	09.04.2006	635.00	0.00	635.00

Стр. 1/4

Рис. 1.4. Реалізація системи документообігу в AutoSoft

На рисунку зображено попередній перегляд збережених документів.

Розробники CPM-системи «АвтоПідприємство» надають наступний функціонал свого програмного продукту:

– здійснення обліку та автоматизації документообігу кількох підприємств будь-якої форми власності;

– ведення всього документообігу підприємства;

– вбудовані довідники за нормами часу на ремонт автомобілів - вітчизняних та імпортних;

– ведення складу на всіх виробничих рівнях;

– прогноз витрати на майбутній період;

− підтримка системи замовлень, закупівель, зберігання та продажу запасних частин;

− повний облік операцій ремонту автотранспорту, зберігання історії ремонтів по всіх автомобілях, заведених у програму;

− облік наданих послуг (зарплата, прибуток, трудовитрати);

− формування переліку робіт – калькуляції вартості робіт;

− складання складських та товарних звітів, аналіз продажу та касових операцій, моніторинг фінансів, інвентаризація та ін;

− експорт інформації до зовнішніх програм (ІНФО-Бухгалтер, 1С, БЕСТ та ін.);

− можливість роботи програми в локальній мережі та на термінальному сервер;

− настроювання та розмежування прав користувачів програми з можливістю контролю з боку керівництва за діями працівників;

− можливість надсилання смс-повідомлень клієнтам прямо з інтерфейсу програми - модуль SMS-інтегратор та ін. [7].

У цьому варіанті CRM-системи застосовано платформу зберігання інформації - клієнт-сервер на основі FireBird SQL Server.

Далі пропонується огляд WEB-застосунок від компанії RemOnline. В ході аналізу було виявлено, що ця CRM-система користується на абиякою популярністю серед власників СТО підприємств.

За відгуками користувачам можна сказати, що система відповідає всім сьогоднішнім критерієм, які є до даного сегменту програмного забезпечення. Мова йде про зручність використання, зрозумілого та приємного інтерфейсу користувача а також є дуже привабливою з точки зору економії часу.

Під час ознайомлення з програмним продуктом було виявлено наступний функціонал системи[8]. Для наглядної зручності усі основні можливості системи розбито на функціональні блоки.

Просте управління замовленнями. Ведення обліку клієнтів та їх замовлень, відстеження історії виконаних робіт та використаних матеріалів. Налаштування ланцюжків статусів для різних видів замовлень, щоб прискорити обслуговування та контролювати терміни. А також ведення попередній записів на дні та тижні вперед, щоб рівномірно розподілити навантаження між співробітниками та ресурсами.

Контроль працівників. Надана можливість контролю всіх дії працівників у програмі на одній сторінці журналу подій (див. рис. 1.5). Налаштування індивідуального доступу до програми для співробітників, щоб вони бачили ті дані, які потрібні їм для роботи та можливість спостереження за ходом робіт працівників.

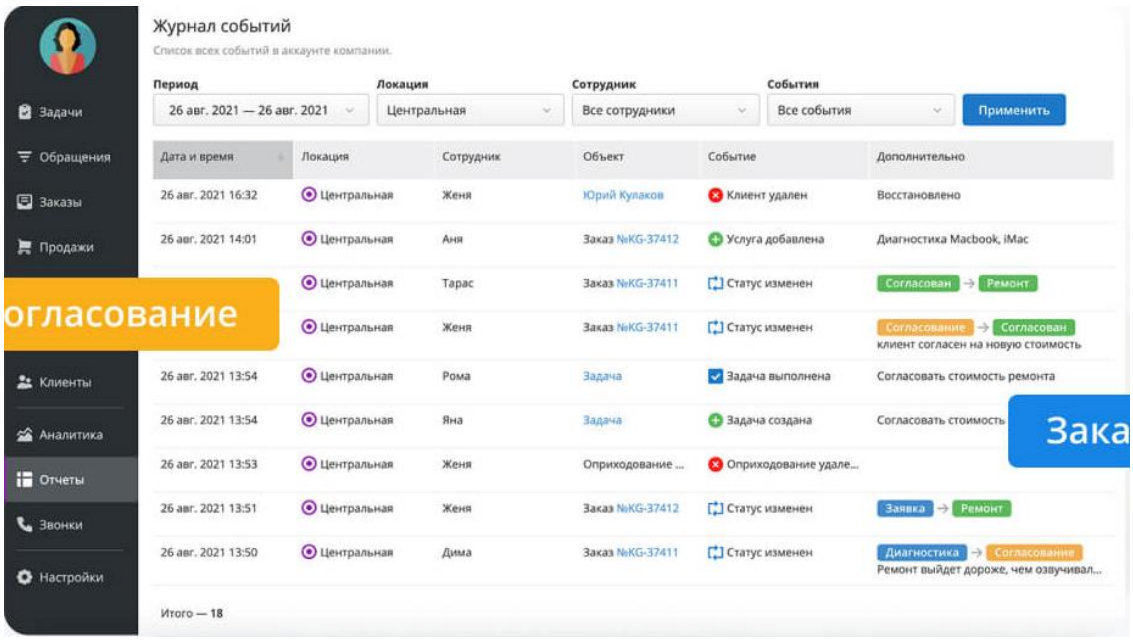


Рис. 1.5. Реалізація журналу подій в RemOnline

Зручний складський облік. Можливість виявляти популярні та рентабельні товари, щоб не заморожувати бюджет на складі. Визначення недостачі та пересортиці одним із 4-х доступних у програмі способів інвентаризації.

Облік фінансів. Додає до програми податкові ставки, що застосовуються до компанії. Ведення взаєморозрахунків з клієнтами та

постачальниками у програмі, щоб контролювати заборгованості та контроль витрат та доходів компанії за статтями для оптимізації бюджету.

Звітність та аналітика. Можливість генерації більше 20 видів звітів про стан складу, замовлень та фінансів, щоб швидко приймати правильні управлінські рішення. Опція аналітичних звітів для відстеження показників бізнесу у зручному форматі графіків, що зображено на рисунку 1.6.

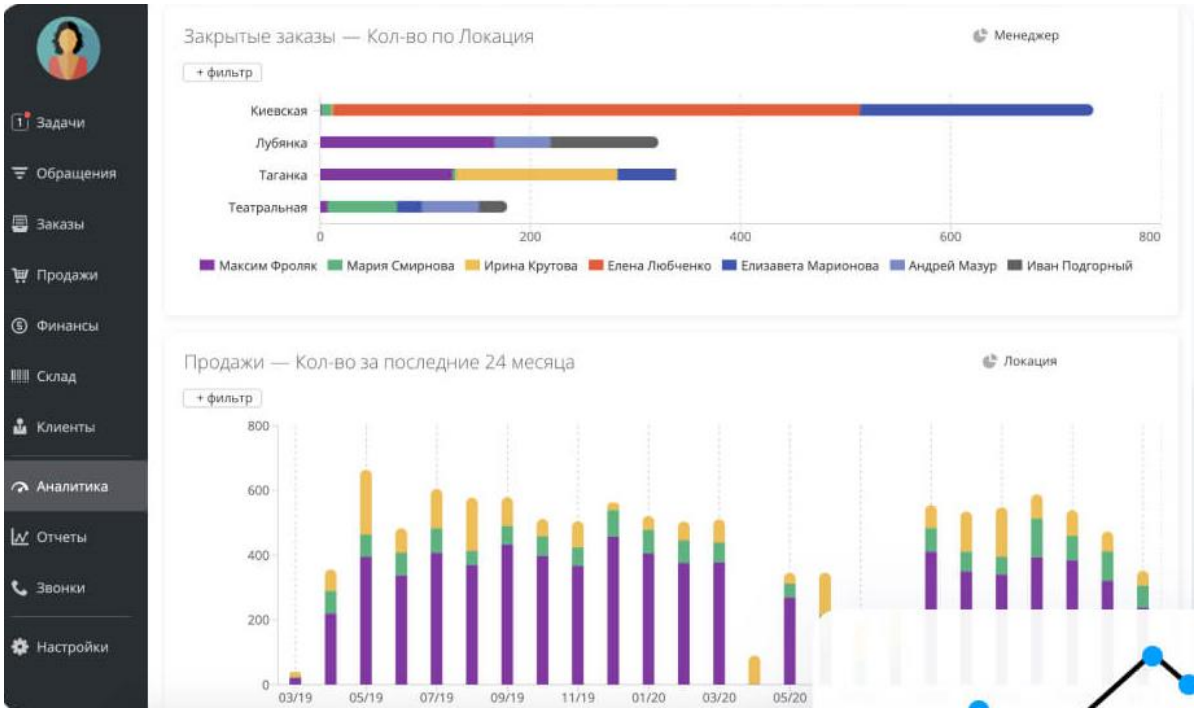


Рис. 1.6. Реалізація аналітичної звітності за допомогою графіків

Таким чином, було виконано аналіз аналогічних систем та виявлено їх позитивні сторони. Під час аналізу було зроблено певні висновки та зауваження до вже існуючих систем, що допоможуть в подальшій розробці власного програмного продукту.

В наступному підрозділі буде проведена порівняльна характеристика усіх виявлених систем.

1.2 CRM-системи для контролю роботи СТО

У цьому підрозділі магістерської дипломної роботи виконано аналітичне порівняння усіх виявлених аналогів для CRM-системи, що буде виконана в подальшому.

Для власної системи було визначено ряд основних критеріїв, які маюьб бути реалізовані. Під час аналізу, увагу було звернено на наявність наступних факторів web-застосунку — простота інтеграція з сайтом компанії, реалізація за допомогою сучасних web-технологій, низькі витрати на реалізацію та впровадження, онлайн реєстрація нових клієнтів, наявність особистого кабінету клієнта, заповнення бланку замовлення в режимі онлайн , вікно чату та обмін із клієнтом повідомлення за допомогою електронної пошти та SMS-повідомлень.

Усі ці критерії були визначені, як необхідний мінімум в реалізованій системі. Кожен з них, має таких чи інший вплив на потенціальний успіх у ведені бізнесу, як вже було зазначено, необхідно бути завжди орієнтованим на клієнта щоб досягти певних успіхів.

Для візуальної зручності наявність необхідного критерію буде відмічена знаком «+» та знаком «-» у випадку його відсутності. Звичайно, аналіз вже існуючих систем проводиться для того, щоб визначити свого роду орієнтир під час розробки програмного продукту. Тому, логіка порівняння полягає в тому, щоб визначити аналог який найбільше відповідає задуманій CRM-системі.

Саме за допомогою такого аналітичного процесу буде значно легше поставити необхідні задачі, що мають бути реалізовані у подальшому. Методи порівняльного підходу призводять до того, що можна переглянути вже існуючі погляди на ті чи інші приклади реалізації. При порівнянні можуть бути відкриті особливості, які специфічні для конкретного критерію чи явища, проте раніше були відомі дослідникам [9]. Таким чином, порівняння, що були виконані відображені в таблиці 1.1.

Таблиця 1.1

Порівняльна характеристика iDirector Авто, 1С Підприємство 8. Автосервіс , АвтоПідприємство, RemOnline

Критерій	iDirector Авто	1С Підприємство 8. Автосервіс	АвтоПідприємство	RemOnline
Простота інтеграція з сайтом компанії	Відсутня	Відсутня	Відсутня	Можливість є
Реалізована за допомогою сучасних Web технологій	Можливість є	Відсутня	Можливість є	Можливість є
Низькі витрати на реалізацію та впровадження	Відсутня	Відсутня	Відсутня	- Відсутня
Онлайн реєстрація нових клієнтів	Можливість є	Відсутня	Можливість є	Можливість є
Наявність особистого кабінету клієнта	Можливість є	Відсутня	Можливість є	Можливість є
Заповнення бланку замовлення в режимі онлайн	Можливість є	Можливість є	Можливість є	Можливість є
Вікно чату	Відсутня	Відсутня	Відсутня	Можливість є
Обмін із клієнтом повідомлення	Можливість є	Можливість є	Можливість є	Можливість є
Підсумок	**5**	**2**	**5**	**7**

Таким чином, існуючі аналоги не відповідають усімвимогам щодо функціональності та вартості.

1.3 Постановка задачі

Беручи до уваги недоліки вже існуючих аналогів та використання CRM-систем, в найближчі роки цей спосіб керування станцією технічного обслуговування очікує стрімке зростання. Якщо власник бізнесу хоче перебувати у добрих стосунках з клієнтами, або як ще кажуть клієнтоорієнтованим, потрібно вже зараз включати до роботи автоматизовану інформаційну системи яка надасть йому для цього всі існуючі можливості та шляхи вдосконалення стану речей на сьогоднішній день.

Аналіз ринку існуючих CRM-систем для контролю за роботою персоналу СТО за рахунок оптимізації процесів взаємодії з клієнтами виявив наступні фактори:

- ринок більше зосереджений на представлені послуг, а не товарів;
- в усіх існуючих сервісах відсутня принаймні одна деталь механізму, щоб усвідомлювати користувачів максимально конкретно.

Було зроблено висновок про доцільність створення власної системи CRy, що буде поєднувати в собі функціональність існуючих веб-сервісів та десктопних за стосунків для ведення клієнтської бази та та вже існуючих мобільних застосунків для більш швидкої взаємодії користувачів. Створюваний програмний застосунок має поєднувати в собі простоту, функціональність та зручність використання.

Пропонується створення програмного застосунку, що буде поєднувати в собі простоту і функціональність, буде корисним як працівникам автосервісу, так і власнику.

Створюваний продукт має реалізовувати наступні функціональності:

- клієнт-серверна архітектура;
- веб-застосунок;
- Telegram бот для більш зручного використання;

Висновки до розділу 1

Було проведено аналіз у ході аналізу предметної галузі було виявлено основні недоліки існуючого бізнес-процесу управління замовленнями клієнтів та контролю персоналу СТО. Окрім цього, були запропоновані недоліки, які потребують окремої уваги.

Запропоновано удосконалити бізнес-процес за рахунок впровадження CRM-системи, що значно підвищить об’єми продажів.

Аналіз існуючих аналогів показав, що вони не відповідають усім вимогам щодо функціональності та вартості, том ухвалено рішення про створення власної CRM-системи керування замовленнями клієнтів та подальшої комунікації.

Для маркетингових та інформаційних цілей, а також покращення показників бізнесу доцільно використовувати всі технології які допоможуть власнику слідкувати за якістю роботи сервісу, так як сучасний розвиток інформаційних технологій призводить до зміни споживчих знань і очікувань від ринку, особливо в підприємницькій сфері та автосервісу, що, в свою чергу, вимагає підвищення рівня сервісу.

Були запропоновані функціональні вимоги до програмного продукту, що задовольнить вимоги користувачів.

2 ПІДХОДИ ТА ТЕХНОЛОГІЇ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ

В ході розвитку сфери розробки CRM-систем та програмного забезпечення фахівці почали створювати та використовувати в процеси розробки програм шаблони, алгоритми та методики для виконання певних завдань. В ході багатьох десятиліть розвитку сфери розробки програмного забезпечення багато таких ідей виявились не досить доречними та працездатними й наразі рідко використовуються. Однак, певна частина таких ідей показала свою працездатність й закріпилась у колах розробників й наразі використовується доволі часто. Тому, на мою думку, слід придивитися до таких ідей та підходів та почати орієнтуватися на них, адже сьогодні вони є досить популярними та перевіреними роками.

Також перед тим, як почати реалізацію web-застосунку CRM-системи слід здійснити проектування всієї архітектури розроблювальної системи, бази даних, обрати платформу та визначитись з мовою програмування. Треба описати можливі технології та обрати необхідні архітектуру базуючись на найбільш вдалих та технологіях, що повністю задовольняють та дозволяють реалізувати необхідний функціонал для вирішення поставленої мети.

Слід зазначити, що при досить вдалому й доречному виборі необхідних технологій та грамотної архітектури ми можемо оминути та вирішити велику кількість проблем й забезпечити наступні можливості:

- забезпечення продуктивності й безперебійної роботи системи;
- зручне та швидке додавання нових функцій до існуючої системи;
- легке масштабування та розширення на нові платформи;
- поділ повноважень користувачів інформаційної системи;
- просте та легке тестування створюваної системи;
- розширення можливостей інформаційної системи;
- можливість повторного використання вже написаного коду;

2.1 Опис підходів та технологій

Постає першочергова задача необхідності вибору шляху розробки програмного продукту та сам підхід створення CRM-системи для контролю за роботою працівників та оптимізації бізнес-процесів. Нам відомо, що на сьогоднішній день, найбільш популярними платформами для інтелектуальних інформаційних системи є персональний комп'ютер та смартфон. Проте, при розробці й розвитку сучасних застосуків велику роль відіграє кросс-платформеність та гібридність. Цей підхід, як відомо, необхідний для залучення більшої кількості користувачів програмним продуктом. Насамперед, слід відзначити, що світ продажів працює 24/7 а сам менеджер з продажів або навіть власник автосервісу має бути на зв'язку завжди, щоб залишатися конкурентоспроможним[10].

Із статистики, що можна вільно віднайти в мережі Інтернет ми маємо показники, що свідчать про те що, половину всього часу, яку людина проводить в мережі Інтернет припадає на використання смарфтону. Також, не менш важливі чинники смарфону – зручність та миттєвий доступ до нього. Тому було прийнято рішення зосередитися на розробці web-застосунку із дотриманням усіх вимог до адаптивності ресурсу. Проте, щоб рішення було більш обґрунтованим необхідно провести дослідження. Результати цього дослідження будуть для того, щоб бути максимально впевненим в прийнятті такого рішення. Слід порівняти всі відомі способи створення CRM-системи з нуля, їх переваги та недоліки. Дослідити усі потенційні платформи та підходів до реалізації запланованої CRM-системи для контролю за роботою автосервісу

Першочергово для вибору підходів та технологій, що будуть використовуватись під час створення web-застосунку для контролю роботи СТО, необхідно детально їх описати [11].

Постає необхідність, обрати підхід до створення застосунку. Нам відомо, що існують такі підходи: створення мобільних застосунків, створення десктопних застосунків, створення веб–застосунків..

Необхідно брати до уваги той факт, що в основному такі засоби програмного забезпечення, як CRM-системи були розроблені під цільову аудиторію, що являє собою здебільшого офісних працівників.

Для усіх працівників відділу продажів, компанія намагається надати робочі ресурси такі як – ноутбук, десктоп в рідших випадках смартфони та планшети. Тобто, іншими словами, виділити працівнику достатнє технлогчне оснащення для продуктивної роботи [12]. В сьогоднішніх реаліях, такі дії керівництва компанії вважаються як необхідні, оскільки велика кількість менеджерів працює віддалено.

Беручи це до уваги, оптимальним рішенням являється створення такої інформаційної системи, яка буде зручною у використанні саме на ноутбуках та персональних комп’ютерах. Звичайно, як вже зазначалося раніше, що в сфері продажів необхідно бути завжди в тонусі і тому постає окрема завдання з реалізації адаптивності створеної системи, але про це пізніше. Також, можна з впевненістю затверджувати, що CRM-система несе в собі великі об’єми інформації. Для її комфортного використання просто необхідно користуватися пристроєм з широкою діагоналлю екрану, щоб мати перед собою всю актуальну інформацію.

Наразі, з огляду на попередній аналіз аналогічних систем – ми можемо зробити висновки, що кінцевий програмний продукт можна розробити як з повного нуля, так і користуючись вже готовими технологічними рішеннями як базовою точкою у дорожній карті проекту [13]. Але чи це все, що в нас є? Відповідь на це питання й планується розкрити в даному розділі магістерської дипломної роботи.

Якщо, вже було визначено, під які саме пристрої ланується розробка кінцевого програмного продукту, то тепер слід визначити та розібрати – який підхід підійде краще?

Нам відомо, що на ноутбуках та персональних комп’ютерах існує дві основні «ніші» в розрізі розробки. Мова йде про створення web-застосунку або повноцінного за стосунку для персонального комп’ютера. Відповідь на це питання заслуговує на більш розгорнуту відповідь, а також на виявлення сильних та слабких сторін кожного. Необхідно визначитись, ще на самому початку оскільки це буде впливати не тільки на весь процес реалізації ідеї, а й на майбутній користувацький досвід.

Далі пропонується аналіз, що був проведений для вирішення поставленої задачі, який буде представлений в таблиці 2.1.

Для зручності порівняння та встановлення кінцевого рішення критерії які будуть розглядатися в якості основних визначено наступні – інсталяція та оновлення продукту, кінцевий реліз, надійність, доступність, кросплатформеність, функціональність та безпечність використання системи. А подальший аналіз буде представлений як виділення основних тезисів з приводу розробки web-застосунку спочатку й за стосунку для персонального комп’ютера після[14].

Інсталяція та оновлення продукту – це, перше на що було звернемо уваги. Що стосується web-застосунку. Він не вимагає установки, як такової, всі оновлення відбуваються на сервері, доставляються користувачам відразу - досить просто перезавантажити сторінку або вийти, а потім знову зайти в обліковий запис. Але іноді для його роботи необхідно встановити додаткові бібліотеки або використовувати захищені мережеві протоколи.

Десктопний застосунок потрібно встановлювати на комп'ютері або мобільному пристрої, оновлювати щоразу, коли виходить нова версія. Незважаючи на те, що найчастіше процес автоматизований — це все одно

займає час користувачів і ресурси пристроїв. Додатково доведеться відстежувати версії на кожному комп'ютері, смартфоні та планшеті.

Далі розглянуто процес публікації або як кажуть розгортання за стосунку. В першому випадку за стосунку він публікується на локальному або хмарному сервері, там відбувається процес оновлення. У цьому сервер потрібен у разі, навіть якщо рішення дуже просте. Адже крім фронтенду, з яким користувачі працюватимуть через браузер, потрібно десь розміщувати бекенд.

В процесі розробки нативного продукту для персонального комп’ютера доведеться встановлювати вручну на кожному пристрої. У компанії, де багато робочих місць, це може забрати досить багато часу. Плюс у тому, що не обов'язково вибирати сервер або шукати ресурси для публікації, якщо не йдеться про клієнт-серверне рішення.

Питання надійності в web-застосунку залежить не тільки від того, наскільки грамотно воно розроблено і яким характеристикам відповідає пристрій, але також від швидкості інтернет-з'єднання, працездатності віддаленого сервера.

Десктопне працює автономно, тому головне - якість коду та стабільність обладнання, на якому цей код виконується. Але якщо зв'язок із сервером необхідний, то виникають ті ж проблеми, що у «конкурента».

Доступність. Веб-застосунок доступний з будь-якої точки світу, з будь-якого пристрою, а файли користувача завжди будуть під рукою. Але якщо є інтернет-з'єднання або реалізована можливість роботи офлайн і завантаження-вивантаження даних. Дуже важливий фактор для кінцевого прийняття рішення. Раніше було зазначено, про необхідність саме такого варіанту.

З ноутбуку, наприклад, буде доступне завжди — але лише з пристрою, на якому його встановлено. Щоб працювати з різних пристроїв, його

доведеться встановити на кожному окремо, а також вигадати, де зберігати файли, щоб завжди мати до них доступ[15].

Кросплатформеність. Перший варіант однаково добре буде працювати на будь-якому пристрої, чи то стаціонарний комп'ютер, ноутбук, планшет чи смартфон — адже він практично не залежить від «заліза» чи операційної системи. Головне - браузер. Як правило, для більшості веб-клієнтів підходять Google Chrome, Mozilla Firefox, Safari від Apple або Windows-браузер (Microsoft Edge / Internet Explorer).

Другий варіант цілковито залежить від операційної системи, процесора, відеокарти, інших параметрів. Доводиться враховувати нюанси кожного середовища (у тому числі при «вилові» помилок), писати код з урахуванням можливих варіантів, наймати окремих розробників або навіть цілі команди для версій під різні ОС.

Функціональність, швидкодія. web-застосунок повністю залежить від браузера та технологій його роботи. Тому є ряд обмежень, наприклад, доступ до апаратного забезпечення пристрою. Це й деякі інші обмеження оминути неможливо. Але цілу низку завдань можна вирішити. Редактори документів, зображення, аудіо, відео, 3D графіки; системи управління проектами; сховища файлів; no-code конструктори – успішно працюють у браузерах. Інструменти швидкої інтеграції сервісів, а також інтерфейсні бібліотеки ще більше розширюють можливості.

Десктопне дозволяє реалізувати буквально будь-які функції - у цьому воно однозначно перевершує web. Принаймні повноцінного онлайн аналога Photoshop або Sony Vegas ще ніхто не розробив. Системні утиліти - безумовно сфера десктопної розробки. Як і програми, які повинні довго працювати у фоновому режимі, наприклад, чати або торрент-клієнти, через браузер з ними просто незручно буде працювати. Також таке ПЗ найчастіше використовується для специфічних проектів, з нестандартними інтерфейсами

або функціями. Тому web розробка поки не становить небезпеки для desktop програмістів - ці технології розвиватимуться паралельно, просто під різні завдання.

Щодо швидкості роботи все не так однозначно, як може здатися. Незважаючи на те, що браузерний клієнт постійно обмінюється даними з сервером, швидкодія багато в чому залежатиме від того, наскільки грамотно він спроектований, «чистоти» коду, можливостей обладнання, стабільності каналу зв'язку. Різниця в швидкодії, яка є очевидною при тестуванні, часто непомітна для користувачів.

Останній за списком, проте не по важливості критерій – безпека. web-застосунок, розроблений з використанням сучасних протоколів та засобів захисту, здатний повноцінно забезпечувати збереження даних. Однак на деякі моменти розробники не можуть вплинути: браузер, хмарний сервер, канал зв'язку можуть підвищити рівень безпеки за рахунок додаткових засобів перевірки, але також знизити його за рахунок своїх вразливостей. Безперечний плюс для користувачів: таке програмне забезпечення простіше контролювати. Обмеження середовища знижують ймовірність, що воно приховано отримає доступ до файлів або запустить будь-який процес.

На комп’ютерах та ноутбуках налаштовується гнучкіше, а значить - теоретично при його розробці можна передбачити всі потенційні вразливості. Насправді — навряд. Втім, зробити його повністю безпечним все ж таки можна. Але тільки якщо пристрій, на якому він встановлений, не підключатиметься нікуди, навіть до захищеної локальної мережі. Інакше ризик все одно буде.

Однозначно сказати, що безпечніше складно (якщо взагалі можливо). На це впливають багато факторів, насамперед людський. Адже саме у захисті від людського фактора, у різних його проявах, полягає сенс усіх заходів безпеки.

Але очевидно, що довіра до десктопного програмного забезпечення вище. Деякі організації принципово не погоджуються працювати в браузерах, багато користувачів все ще ставляться до них насторожено. Проте ситуація змінюється – з розвитком технологій зростає лояльність людей до них.

Можливості браузерної розробки є величезними, її потенціал розкритий далеко не повністю. Технології розвиваються, ринок рішень зростає, пропонуючи все нові програми - за інших рівних користувачі будуть вибирати web просто тому, що це зручніше. Якщо говорити про рішення для корпоративних клієнтів, то тут браузерні програми незамінні. Вони гнучкі, універсальні, вимагають попередньої підготовки середовища, дозволяють заощадити фінанси компанії, апаратні ресурси, час співробітників.

Таблиця 2.1

Порівняльна характеристика розробки застосунку для WEB та персонального комп'ютера

	Web-застосунок	**ПК застосунок**
Встановлення	+	-
Розгортання	+	-
Надійність	+	+
Доступність	+	-
Кросплатформеність	+	-
Функціональність	+	+
Безпека	-	+
Підсумок	**6**	**3**

В ході проведеного аналізу було встановлено та виявлено усі переваги та недоліки двох популярних підходів та технологій у розробці програмного забезпечення для таких пристроїв як ноутбуки та персональні комп'ютери. Після детального опису кожного з визначених критеріїв можна впевнено запевняти, що майбутня реалізація CRM-системи для контролю за роботою станції технічного обслуговування буде виконана у вигляді web-застосунку.

Оскільки було виявлено лише один недолік, нажаль їм постало питання безпеки. Проте, відомо, що ми стоїмо на порозі входу в еру WEB 3.0 та будемо розраховувати на покращення та знаходження відповідей на питання безпечності аналогічних систем.

Також існує й інша думка. Деякі розробники вважають, що перспективи далеко не безхмарні. Занадто недосконалі технології роботи браузерів. Тому користувачі браузерних рішень повертатимуться назад до десктопних.

Web-застосунки вже зараз підходять для вирішення багатьох завдань - як бізнесу, так і звичайних користувачів.

2.2 Огляд та вибір платформ й підходів

Система CRM для контролю за роботою персоналу СТО за рахунок оптимізації процесів взаємодії з клієнтами, вимоги до якої були представлені у першому розділі магістерської дипломної роботи, може бути реалізована на таких платформах:

— персональний комп’ютер;

— смартфон.

Слід розглянути кожну з них більш детально. Детальний огляд має надати розуміння про кількість активних користувачів із обраної цільової аудиторії, а саме менеджери та адміністратори, програмного продукту.

Отже, щоб створити такий програмний продукт для персональних комп’ютерів можна виділити декілька підходів:

— створення окремого застосунку для персональних комп’ютерів;

— створення веб-сайту.

Створення окремого застосунку для персональних комп’ютерів є доволі популярним підходом в плані реалізації програмного продукту. Проте, на меті стоїть зацікавити користувача, а саме справити на нього враження зручним для використання інтерфейсом, багатим функціоналом та

продуктивністю застосунку. Отже вибір падає на створення веб-застосунку, адже с технічної точки зору та користувацького досвіду, у підході розробки веб-застосунку набагато більше переваг.

Нативні десктопні застосунки являються дещо складнішими з точки зору обслуговування і розгортання за рахунок того, що створений продукт необхідно встановлювати на комп'ютери в ручному режимі. Проте, великим однією з переваг є продуктивність такого програмного продукту, що може бути особливо важливо, якщо стоїть необхідність у додатку до функцій, що вимагають глибокої інтеграції обладнання.

В CRM для контролю за роботою персоналу СТО найважливішим аспектом є продуктивність. Саме за рахунок продуктивності надається можливість опрацьовувати велику кількість інформації дуже швидко. В свою чергу, це на змусить користувача довго чекати.

Слід зазначити, що існують області в яких саме розробка веб-застсоунку є панацеєю та цілком виправдана, наприклад ті ж самі інформаційні системи або соціальні мережі. Якщо ж проект не являється розважальним та вимагає позитивної зворотної відповіді від користувача – така розробка залишається єдиним варіантом.

Було прийнято рішення, що постає необхідність розробити систему CPM у веб-просторі. Оскільки у подальшому завжди залишаються можливості до розширення та подальшого розширення функціоналу програмного продукту.

Таким чином, в якості підходу для вирішення задачі було обрано підхід створення веб-застосунку CRM для контролю за роботою персоналу СТО.

2.3 Вибір мови програмування

Вибір мови програмування — це, найбільш відповідальний момент під час попередньої розробки проекту на бумазі. Опираючись на ті чи інші

показники, функціонал, вимоги та переваги кожної можна робити певні прогнози з приводу працездатності реалізованої системи.

В наш час існує великий вибір мов програмування для написання бекенду web-застосунку. В цьому підрозділі магістерської дипломної роботи необхідно визначитись із способом реалізації та довести, що саме обрані інструменти являються найбільш раціональним способом для вирішення задачі[16].

Нам відомо, що найбільш популярними мовами програмування для розробки бекенду являються Java, PHP, .NET, Ruby, Python, JavaScript та GO.

Далі необхідно визначитись із можливостями, що надає кожна з них. Звісно, озброївшись цією інформацією можна бути зробити вирішальний вибір щодо інструменту реалізації запланованої CRM-системи.

Тепер, постає необхідність виділити ці елементи як найшвидше, з застереженням на те, що від цього не постраждає якість кінцевого продукту. Першою мовою програмування розглянуто Java.

Java - одна з найпопулярніших мов програмування. Вона настільки універсальна, наскільки це можна собі уявити. Окрім вирішення бекенд задач, вона має вплив й на інші сегменти у світі розробки програмного забезпечення. Java дуже універсальна мова програмування та використовується вже понад 20 років.

Універсальність забезпечується віртуальною машиною Java (Java Virtual Machine, JVM). Багато мов під час компіляції програма переводиться в код, який може працювати по-різному на різних пристроях або платформах. У Java цієї проблеми немає. JVM грає роль проміжного рівня - з програми Java вона робить код, який може виконуватися на будь-якому комп'ютері незалежно від того, де код був скомпільований.

Java має велику спільноту, і в Інтернеті можна знайти відповіді практично на будь-які питання про мову її синтаксис та ознайомитися із усіма підводними каменями.

Що можна робити на Java? Технології Java можуть використовуватися для багатьох завдань, серед яких:

— розробка мобільних програм під Android;

— розробка веб-сайтів;

— розробка API для роботи з базами даних;

— цифрове оброблення зображень;

— створення настільних додатків;

— програмування мережевих завдань.

Мова програмування Java пропонує чудові можливості для зниження ризиків безпеки. Крім того, JVM перевіряють байт-коди Java, щоб виявити та знизити ризик зараження вірусами. Так само дана модель безпеки Java разом із тестуванням повторно використовуваних кодів допомагають розробникам уникнути загроз безпеки[17].

Найбільш відомі компанії які використовують Java для досягнення своїх комерційних цілей: FitBit, Evernote, Uber, LinkedIn, eBay, Pinterest, Groupon, Airbnb, Hubspot.

Наступна за списком йде мова програмування PHP. Її використовують близько 78% всіх сайтів. Мова з'явилася у 1995 році, коли було не так багато можливостей для створення динамічних веб-сторінок. PHP динамічно типізована, і той самий фрагмент коду може поводитися по-різному залежно від контексту, що робить програми на PHP складними для масштабування і іноді повільними. За цим критерієм дана опція уже програє попередній, однак необхідно розібратися в тому, які ж все таки є переваги у використанні PHP.

PHP - відмінна мова для початківців. І це вже являє собою неабиякий плюс. Пропоную ознайомитись із рядом причин, що складають таку картину в цілому:

- він прощає помилки: ви можете запустити програму, і вона виконуватиметься, доки досягне ділянки з проблемним кодом;
- у мови велика спільнота, а для новачків є багато навчальних матеріалів. Мова постійно оновлюється, тому переконайтеся, що вивчаєте останню версію;
- встановити та налаштувати PHP досить легко в порівнянні, наприклад, з Ruby on Rails. Ви можете завантажити MAMP (для Mac) або WAMP (для Windows) і все буде готове до роботи через 5 хвилин [18].

Тепер звернемо увагу на те, які ж можливості надає PHP і що взагалі, які програмні продути можна реалізувати використовуючи її:

- збирати дані форм;
- створювати динамічний контент на сторінках;
- відправляти та отримувати файли cookies;
- писати скрипти у командному рядку;
- виконувати сценарії за сервера;
- розробляти десктопні застосунки.

Ця інформації представлена розробниками мови програмування PHP із офіційного сайту.

У сфері створення динамічних веб-сайтів та веб-додатків PHP займає значну частку ринку. Майже всі популярні CMS для веб-розробки написані на PHP.

Компанії які використовують PHP для розробки своїх програмних продуктів: Hootsuite, Mint, Lyft, Facebook, DocuSign, Buffer, Viber.

.NET (VB, C#) Фреймворк з відкритим вихідним кодом ASP.NET від Microsoft використовується для створення веб-сайтів за допомогою мов Visual Basic (VB), C#, F# та інших.

Далі будемо розглядати .NET в розрізі VB та C#. Так як вони являються найбільш популярними в цьому сегменті застосування.

.NET працює на основі архітектурного шаблону MVC (Model-View-Controller). Контролер приймає запити користувача та взаємодіє з моделлю для обробки даних. Потім результат передається до представлення та відображається у вигляді інтерфейсу веб-сторінки.

Викладений у відкритий доступ у 2016 році, .NET може інтегруватися з iOS, Linux та Android через .NET Core. Він дуже стабільний і надійний, що робить його найпопулярнішим вибором для бізнесу. Оскільки .NET є продуктом Microsoft, у нього досить хороша підтримка.

C# - високорівнева мова програмування. Це означає, що розробники можуть написати на ньому програми, незалежні від архітектури процесора конкретного комп'ютера[19].

C# популярний серед розробників, тому що він має деякі переваги C++, але на ньому простіше писати код і уникати при цьому грубих помилок.

Visual Basic - це нащадок BASIC, який успадкував його стиль і поєднує елементи ООП. Це проста мова для початківців: вона широко поширена і має нескладний синтаксис. VB часто застосовують для прототипування.

Недоліком програмування на VB є великий обсяг пам'яті, необхідний установки і запуску інструментів розробки.

Сфера застосування .NET:

—створювати десктопних застосунків;

—створювати мобільних застосунків;

—створювати веб-додатки та ігри;

—працювати з великими даними;

Серед відомих технологічних компаній, які використовують .NET, окрім Microsoft, можна виділити Docplanner та StackOverflow.

Ruby on Rails – це веб-фреймворк на мові програмування Ruby. Ruby on Rails має набір готових інструментів, що дозволяють швидко виконувати базові завдання. Як вже було зазначено раніше, швидкість виконання коду являє собою один із найважливіших критеріїв для реалізації CRM-системи.

Ruby досить лаконічний і не вимагає багато коду для бекенда, що дозволяє розробникам швидко створювати та запускати прототипи Популярність Ruby зросла на початку 2000 років, але з того часу помітно зменшилася.

Ruby - мова з відкритим вихідним кодом, а значить її можна модифікувати та доповнювати, що є як і плюсом так і мінусом.

Ruby дозволяє:

— автоматизувати завдання, що повторюються;

— створювати веб-програми;

— писати мобільні програми та ігри;

— створювати прототипи.

Завдяки короткому коду та доступності сторонніх бібліотек Ruby є дуже продуктивною бекенд-технологією. Розробникам зазвичай потрібно менше часу для вивчення документації Rubi. Це дозволяє бекенд-технологіям використовувати вже існуючі проекти.

Python став дуже популярною мовою програмування. Він використовується як для веб-розробки, так і для створення десктопних програмних продуктів. У мережі можна знайти величезну кількість навчальних сайтів, навчальних посібників та відеоматеріалів з Python, що робить його доступною мовою для новачків.

Крім того, синтаксис Python є простим і легким для розуміння в порівнянні з іншими мовами. Python підтримує об'єктно-орієнтоване,

функціональне та аспектно-орієнтоване програмування, а також динамічно типізована мова з відкритим вихідним кодом.

Деякі з найбільш популярних застосувань Python:

— крос платформні shell-скрипти;

— швидка автоматизація;

— веб розробка:

— Data Science, Machine Learning.

Python використовує відкритий вихідний безкоштовний код. Саме тому розробники та компанії можуть використовувати для своїх проектів велику кількість безкоштовних бібліотек та інші джерела.

JavaScript — гнучка мова, яку можна використовувати як для фронтенду, так і для бекенда. Це хороша мова для початківців, оскільки в неї мало налаштувань, і можна почати писати код у браузері.

Гнучкість JavaScript іноді обходиться дорого: скрипти працюють повільно, їх складно підтримувати і масштабувати, як і в більшості динамічно типованих мов.

JavaScript має велику спільноту, і для її вивчення в мережі є багато корисних матеріалів.

Значно розширює можливості JS програмної платформи Node.js. З її допомогою код, написаний на JS, можна запускати без браузера на бекенді. А наявність величезної кількості готових рішень у пакетній екосистемі, дозволяє розробнику не витрачати час створення більшості типових рішень.

Також була проведена аналітика з приводу популярності тої чи іншої мови програмування. Результати дослідження відображені на рисунку 2.1

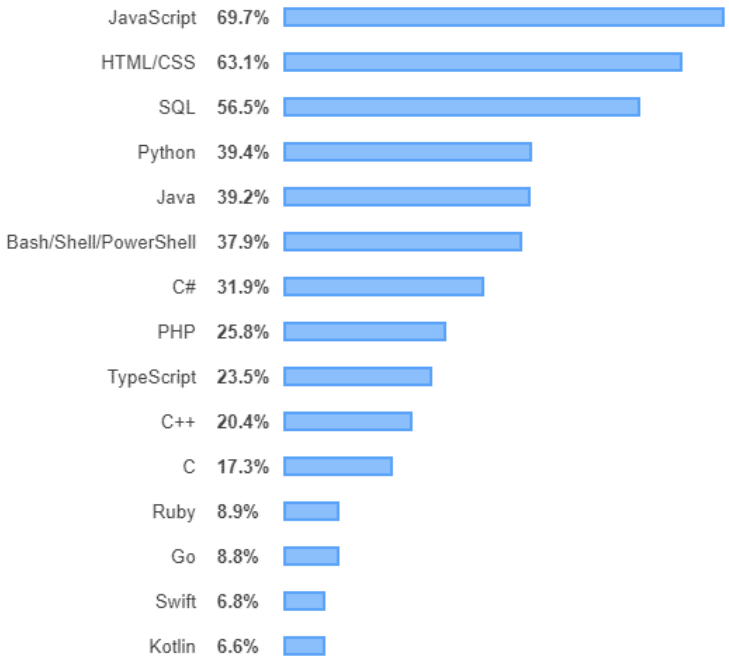


Рис. 2.1. Популярність мов програмування для вирішення бекенд задач

Після аналізу і збору даних було обрано мову програмування Python. Адже, усі переваги, що в неї є перед своїми «опонентами» підходять якнайкраще для реалізації поставленої раніше задачі.

Висновки до розділу 2

Даний розділ має на меті детальний опис зазначених підходів та технологій для вирішення задач з подальшим обґрунтуванням причин вибору того чи іншого підходу або технологій для вирішення поставленої задачі, щодо реалізації CRM-системи. В якості підходу до створення програмного продкутку було прийнято рішення обрати підхід до реалізації web-застосунку. Обрана та, що задовольняє більше вимог а саме мова програмування Python.

Було визначено ряд причин, що вказують на переваги створення інформаційної CRM-системи. Саме ці переваги мають вирішальне значення у баченні того, як необхідно розробляти програмний продукт і який має бути кінцевий результат опираючись на ці критерії.

3 ПРОЕКТУВАННЯ СИСТЕМИ ТА ДИЗАЙНУ

На сьогоднішній день можна спостерігати деякі тенденції розвитку інформаційних систем, а саме використання методів автоматизації підтримки рішень, підвищення зростання масштабованості так складності самих інформаційних систем, так і програмних та апаратних ресурсів. Враховуючи специфічні вимоги оточення з’являється необхідність у розробці методів та засобів підтримки моделювання та працездатності інформаційної системи.

Зараз широко використовується структурний підхід до проектування інформаційних систем, особливо на початкових етапах планування інформаційної системи, розробниками активно використовується об’єктні методи проектування систем. Такі методи орієнтовані на системи, що функціонують у порівняно стаціонарному середовищі і не завжди пристосовані до динамічних середовищ. Ці системи орієнтовані скоріше на повторне проектування інформаційних систем, що вимагається у випадку постійних змін значних додаткових часових та грошових ресурсів.

Взагалі, проектування є одним із найважливіших етапів, якщо не самим, всього життєвого циклу будь якого програмного продукту. Для вдалого запуску проекту, він має бути дуже детально описаним та вдало спланованим. Однією з основних вимог для вдалого проектування інформаційної системи є те, що всі необхідні функціональності повинні бути розписані, а інформаційні моделі – побудовані. Також необхідно звернути увагу на те, що потреби користувачів можуть змінюватись, навіть під час розробки програмного забезпечення, а це в свою чергу і так ускладнює процес розробки. На етапі проектування головним завданням розробника є перетворення власних бажань, або навіть бажань замовника при комерційній розробці продукту, у відповідні рішення, що будуть забезпечувати здійснення та реалізацію усіх побажань та замислів у вигляді відповідної налагодженої та працюючої системи, застосунку.

Основним принципом проектування інформаційної системи є принцип декомпозиції кінцевого продукту на менші частини, підсистеми та компоненти. Такий принцип допомагає розробнику, яким способом необхідно розроблювати програмне забезпечення.

Таким чином, декомпозиція кінцевого продукту розробки веб-застосунку CRM-системи для контролю за роботою станції технічного обслуговування, буде проведено шляхом створення функціональної та процесної моделі систем. Після чого необхідно буде створити діаграми прецедентів для відображення використання елементів моделі.

3.1 Функціональна/процесна модель систем

Перед початком проектування та створення функціональної моделі було прийнято рішення розробити графічне відображення списку головних функцій. Таке відображення повинно містити в собі конкретні блоки , саме на ці блоки буде розбито функціональну модель[20]. Графічне відображення функцій системи відображено на рис. 3.1.

Рис. 3.1. Графічне відображення дерева функцій системи

Перелік функціональних блоків систем зображених на рисунку:

1. Вхід до системи
2. Реєстрація нового користувача
3. Редагування власних параметрів
4. Додавання нових продуктів
5. Видалення продуктів
6. Редагування поживної цінності продуктів

Після того як було зроблено виділення функціональних блоків системи, була створена функціональна модель

Функціональна модель надає можливість розробнику для виконання досконалого аналізу роботи всієї системи взагалі та домотає звернути увагу на можливості вдосконалення роботи. Нотація IDEF0 найчастіше використовується для її представлення[21].

Така методологія частіше за все використовується для опису функціональної моделі, або інакше кажучи – опису всіх процесів верхнього рівня. Нотація IDEF0 цілком дозволяє описати всі можливості розробленого продукту або навіть всю діяльність компанії. Такий метод базується цілком на графічному відображені функціонального змісту різноманітних систем, включаючи ідентифікацію елементів суб’єкта та навіть інформацію про його компоненти та функції Поліпшення сприйняття наглядних моделей для широкого спектру користувачів та розробників досягається за рахунок зручного у використанні властивостей даного графічного представлення. Даний метод надає та дозволяє використовувати засоби для відображення не тільки входів і виходів блоку, що розглядається та аналізується, а й управління і задіяних механізмів. Використовуючи нотацію можна відображати інструменти та ресурси, за допомогою яких реалізується бізнес-процес.

У даній методології за допомогою входу зображають різноманітні об’єкти – інформаційні та матеріальні потоки, що задіяні в бізнес-процесі. За допомогою управління показуються об’єкти – матеріальні та інформаційні потоки, що в подальшому будуть перетворені у процеси, що потрібні для його виконання. Загальна картина полягає в відображені прямокутника та стрілок, які входять та виходять із нього. За своєю суттю прямокутник являє собою певний бізнес-процес, а стрілки – вхід, управління, вихід та задіяні механізми.

Нотація IDEF0 також має певні переваги та недоліки. До переваг цієї методології можна віднести те, що вона показує взаємодію всіх бізнес-процесів в загальному вигляді, без зайвих подробиць. До мінусів можна віднести те, що вона не надає можливості перегляду алгоритму виконання бізнес-процесів [22]. Тобто потребує певної підготовки для розробки та прочитання нотації. Плюси та мінуси даної нотації слід розуміти основним споживачам даної методології, яким необхідно бачити й розуміти взаємозв’язок процесів.

В якості діаграми для опису меж системи було створено функціональну модель, представлену у вигляді нотації IDEF0 з елементами входів та виходів, «керування» та «механізмів». Приклад IDEF0 діаграми та її декомпозицію можно побачити на рис. 3.2 та 3.3.

Рис. 3.2. Функціональна модель, представлена у вигляді нотації IDEF0

Процес декомпозиції – це, такий процес під час якого проводиться розбиття однієї великої задачі або компонента на ряд менших компонентів або завдань, які необхідно вирішити.

Для сформованої функціональної моделі слід виконати процедуру декомпозиції складових функціональних блоків [23]. Таким чином усі функціональні блоки будуть представлені за допомогою нотації IDEF0

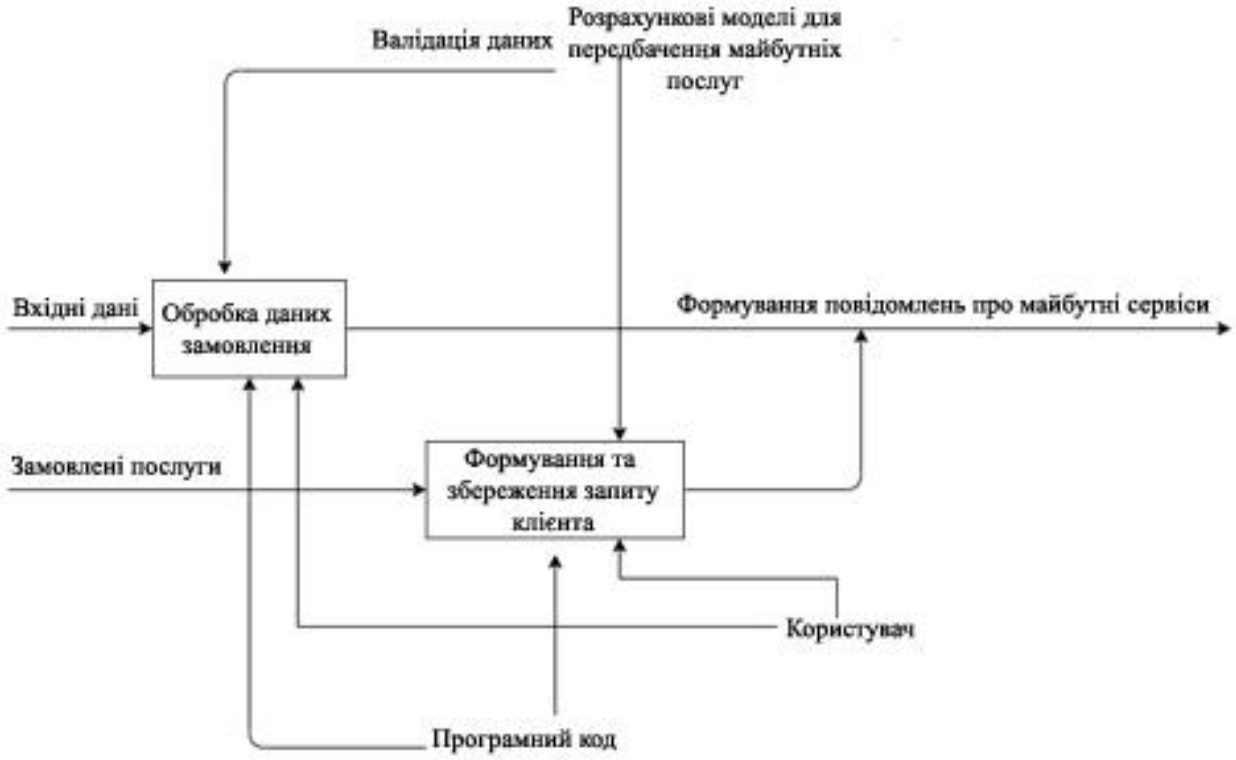


Рис. 3.3. Декомпозиція функціональної моделі за допомогою нотацій IDEF0

Після виконання загальної процедури декомпозиції функціональної моделі на блоки постала необхідність провести декомпозицію блоків для відображення відповідних дій процесів, що відбуваються у системі. Декомпозиція блоків зображена на рис. 3.4 та 3.5.

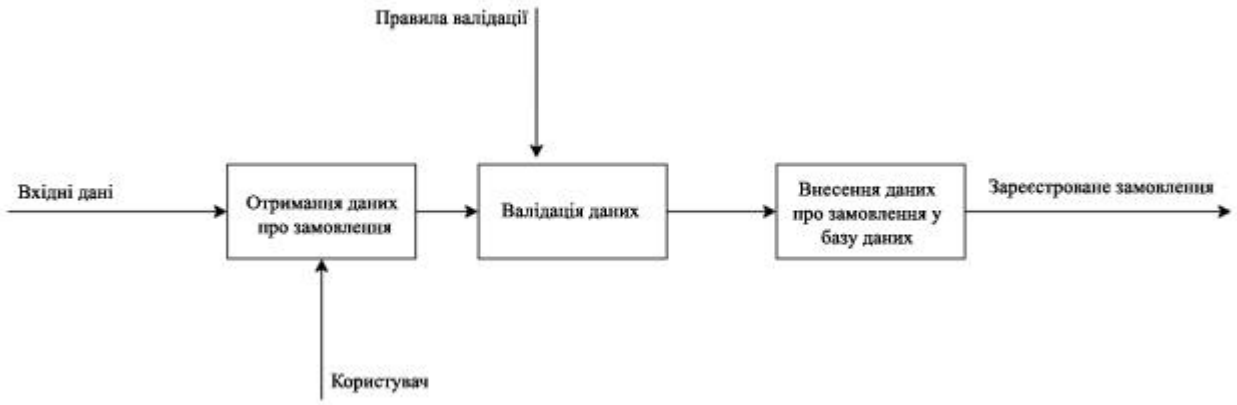


Рис. 3.4. Декомпозиція блоку «Обробка даних замовлення»

Таким чином була виконана декомпозиція блоку «Отримання даних про замовлення».

Рис. 3.4. Декомпозиція блоку «Розрахунок вартості послуги»

Після виконання процесу декомпозиції та проектування функціональної моделі CRM-систем для контролю за роботою персоналу СТО за рахунок оптимізації процесів взаємодії з клієнтами стане етап проектування інформаційного забезпечення.

3.2 Проектування інформаційного забезпечення

Наступним кроком після проектування і опису функціональної моделі та декомпозиції функціональних блоків постала необхідність в реалізації діаграми прецедентів[24]. Тобто постала необхідність в проведені її опису. Також необхідно розробити логічну схему для даних та роботи з ними й в

цілому створити дизайн загальної системи та продумати всі можливості імплементації та підключення додаткових сервісів.

3.2.1 Діаграма прецедентів. Опис прецедентів. Логічна схема даних

Діаграма прецедентів є базовим концептуальним поданням системи в процесі її розробки та проектування. Дана діаграма зазвичай використовується в Unified Modeling Language (UML).

Поняття прецедент – це, певна можливість вже змодельованої системи, завдяки якій кінцевий користувач може отримати цілком конкретний, вимірний і потрібний йому результат в ході виконання певного процесу, що відбувається в системі.

За своєю суттю прецедент відповідає окремій частині або сервісу системи й визначає його один з можливих варіантів її використання, описуючий типовий варіант використання цієї системи та користувача. Такі варіанти використання зазвичай застосовуються для документації специфічних зовнішніх вимог до системи[25].

Діаграма прецедентів – графічний опис, що складається з множини елементів: акторів(користувачів системи), прецедентів (іншими словами можливостями й варіантами використання), всіх відношень між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами[26].

Основним призначенням діаграми прецедентів є опис фунціоналу та поведінки, що дозволяє замовнику, кінцевому користувачу та розробнику сумісно обговорювати систему, що проектується або вже існує.

При модулюванні системи за допомогою діаграми прецедентів слід зосередитися на:

- чіткому відділені системи від її оточення ;
- визначені акторів, їх взаємодію з системою й очікувану функціональність системи;

– визначити дефініції в предметній сфері, що відносяться к детальному опису функціональної системи, тобто прецедентів.

Робота над діаграмою може розпочатись з текстового опису, які були отримані від замовника. При цьому не функціональні вимоги при складані моделі прецедентів не ураховуються.

В якості прикладу, було окремо наведено діаграми прецедентів проектованої системи мобільного застосунку для планування здорового харчування. Така діаграма зображена на рис. 3.6 – 3.9.

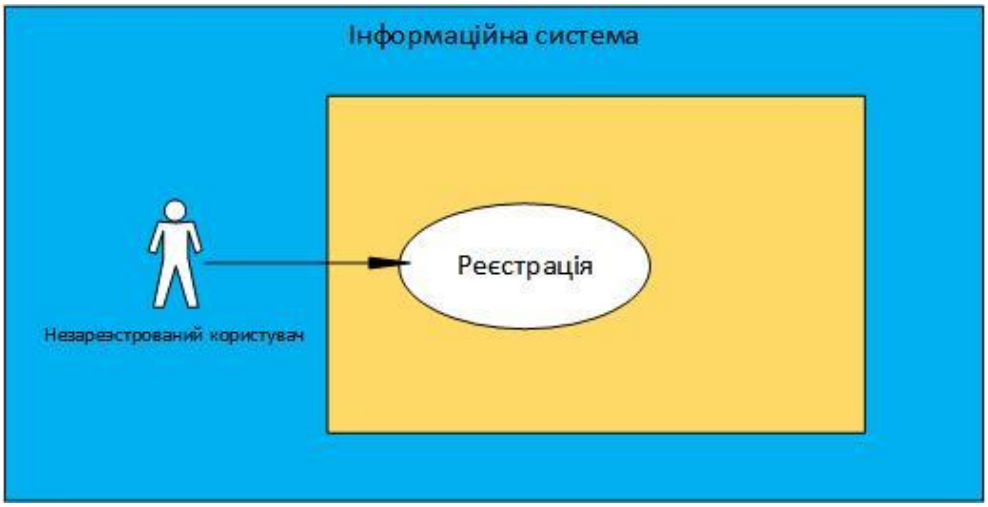


Рис. 3.6. Діаграма прецедентів для незареєстрованого користувача

Рис. 3.7. Діаграма прецедентів для клієнта

Рис. 3.8. Діаграма прецедентів для менеджера

Рис. 3.9. Діаграма прецедентів для адміністратора

Варто зауважити, що під час процесу моделювання й проектування потенційної поведінки системи виникає необхідність не тільки представити процес змін її станів, але й детально представити та охарактеризувати процес й особливості реалізації виконуваних системою функцій та різного роду операцій[27].

Для цього в UML існують діаграми діяльності, що активно використовуються. Діаграма діяльності – одна з багатьох моделей опису

поведінки взаємодіючих груп об’єктів. Як правило, кожна окрема діаграма взаємодії описує поведінку тільки в межах одного варіанту використання.

На таких діаграмах прийнято відображати екземпляри об’єктів та повідомлення, якими ці об’єкти спілкуються між собою в рамках одного варіанту використання. Також в них використовується позначення станів та переходів. Кожен стан на діаграмі діяльності можна віднести до виконання певної елементарної інформації, а перехід в наступний стан для виконання відбувається лише за умови завершення попередньої операції[28].

В якості прикладу, нижче наведено більш детальний опис головних прецедентів в формі таблиць в формі табл. 3.1 - 3.11, та діаграми діяльності головних прецедентів на рис. 3.6 – 3.9.

Таблиця 3.1

Прецедент «Перегляд інформації про статус замовлення»

Характеристика	Опис
Короткий опис	Варіант використання дозволяє користувачу переглянути інформацію про статус його замовлення
Акторы	Активний суб’єкт «Зареєстрований користувач»
Передумови	Зареєстрований в системі профіль користувача
Основний потік подій	Перегляд інформації про стан власного замовлення в системі
Альтернативний потік подій	Немає
Послідуюче	Після перегляду інформації про статус його замовлення користувач переходить до подальшого використання сервісу

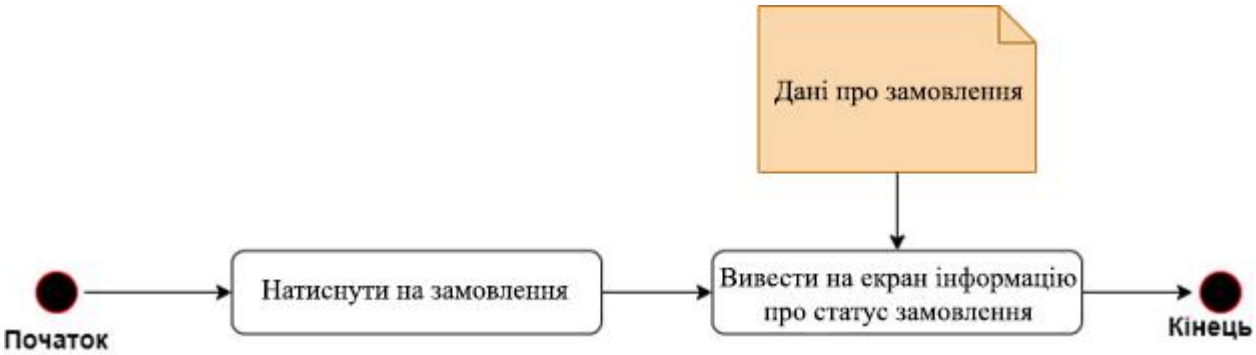


Рис. 3.10. Діаграма діяльності для прецеденту «Перегляд інформації про статус замовлення»

Таблиця 3.2

Прецедент «Перегляд інформації на вкладці сервіси»

Характеристика	Опис
Короткий опис	Варіант використання дозволяє користувачу переглянути інформацію, що знаходиться на вкладці сервіси
Акторы	Активний суб'єкт «Зареєстрований користувач»
Передумови	Зареєстрований в системі користувач
Основний потік подій	Перегляд інформації, що знаходиться на вкладці сервіси
Альтернативний потік подій	Немає

Характеристика	Опис
Послідуюче	Після перегляду інформації зображеної у вкладці сервіси користувач переходить до подальшого використання сервісу

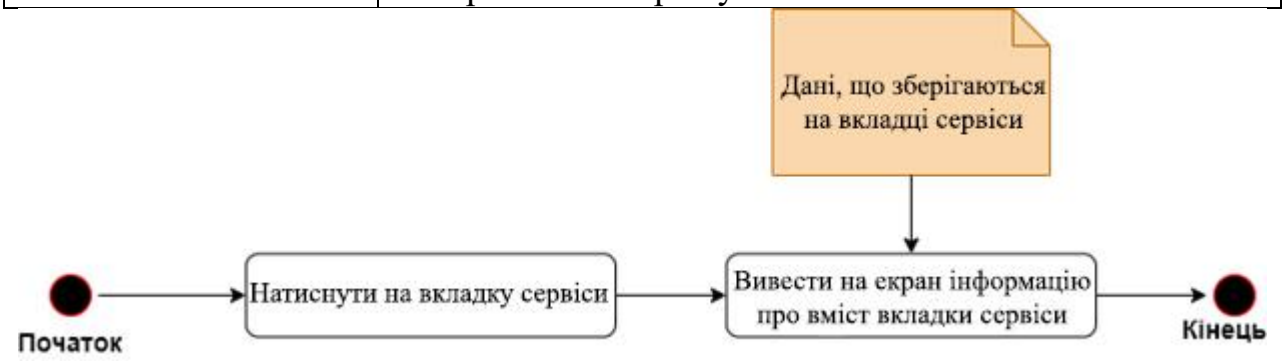


Рис. 3.11. Діаграма діяльності для прецеденту «Перегляд інформації із вкладки сервіси»

Таблиця 3.3

«Прецедент перегляд інформації на вкладці акції»

Характеристика	Опис
Короткий опис	Варіант використання дозволяє користувачу переглянути інформацію, що знаходиться на вкладці акції
Акторы	Активний суб'єкт «Зареєстрований користувач»
Передумови	Зареєстрований в системі користувач
Основний потік подій	Перегляд інформації, що знаходиться на вкладці акції. На вкладці акції знаходить інформація акційною ціною та інформація про всі майбутні акції
Альтернативний потік подій	Немає
Послідуюче	Після перегляду інформації зображеної у вкладці акції користувач переходить до подальшого використання сервісу

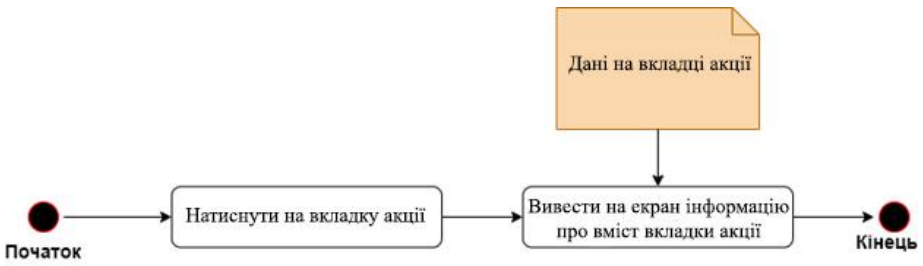


Рис. 3.12. Діаграма діяльності для прецеденту «Перегляд інформації на вкладці акції»

Таблиця 3.4

Прецедент «Перегляд інформації на домашній вкладці»

Характеристика	Опис
Короткий опис	Варіант використання дозволяє користувачу переглянути інформацію, що знаходиться на вкладці профіль
Акторы	Активний суб'єкт «Зареєстрований користувач»
Передумови	Зареєстрований в системі користувач
Основний потік подій	Перегляд інформації, що знаходиться на домашній вкладці
Альтернативний потік подій	Немає
Послідуюче	Після перегляду інформації зображеної у вкладці профіль користувач переходить до подальшого використання сервісу

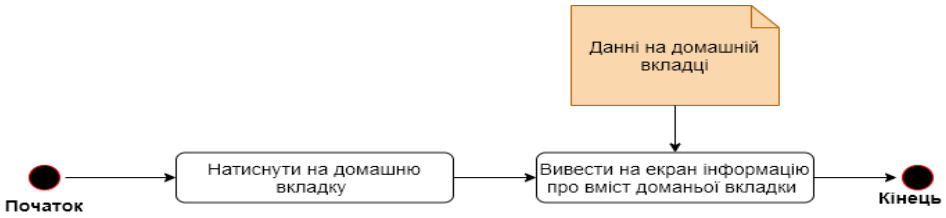


Рис. 3.13. Діаграма діяльності для прецеденту «Перегляд інформації на домашній вкладці»

Таблиця 3.5

Прецедент «Перегляд інформації на вкладці профіль»

Характеристика	Опис
Короткий опис	Варіант використання дозволяє користувачу переглянути інформацію, що знаходиться на вкладці профіль
Акторы	Активний суб'єкт «Зареєстрований користувач»

Передумови	Зареєстрований в системі користувач
Основний потік подій	Перегляд інформації, що знаходиться на домашній вкладці
Альтернативний потік подій	Немає
Послідуюче	Після перегляду інформації зображеної у домашній вкладці користувач переходить до подальшого використання сервісу

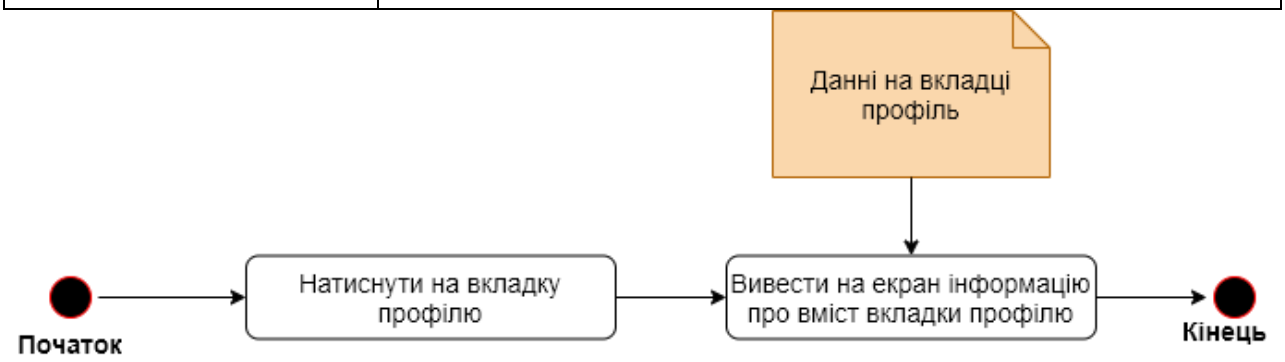


Рис. 3.14. Діаграма діяльності для прецеденту «Перегляд інформації на вкладці профіль»

Таблиця 3.6

Прецедент «Додавання нового замовлення»

Характеристика	Опис
Короткий опис	Варіант використання дозволяє користувачу додати нове замовлення до бази даних
Характеристика	Опис
Акторы	Активний суб’єкт «Зареєстрований користувач»
Передумови	Зареєстрований в системі користувач
Основний потік подій	Додавання нового замовлення
Альтернативний потік подій	Немає

Характеристика	Опис
Послідуюче	Після оформлення замовлення користувач переходить до подальшого використання сервісу

Рис. 3.15. Діаграма діяльності для прецеденту «Додавання нового продукту»

Таблиця 3.7

Прецедент «Редагування замовлення»

Характеристика	Опис
Короткий опис	Варіант використання дозволяє користувачу редагувати інформацію, про замовлення
Акторы	Активний суб'єкт менеджер
Передумови	Зареєстрований в системі користувач
Основний потік подій	Перегляд та редагування інформації про замовлення
Альтернативний потік подій	Немає
Послідуюче	Після редагування інформація зберігається

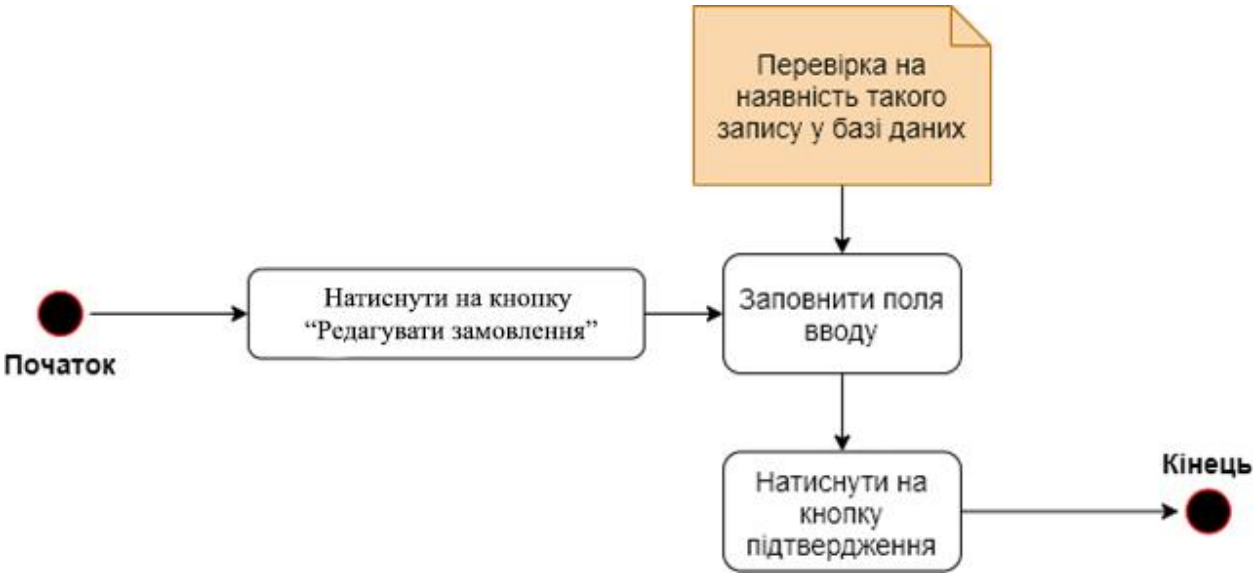


Рис. 3.16. Діаграма діяльності для прецеденту «Редагування замовлення»

Таблиця 3.8

Прецедент «Видалення замовлення»

Характеристика	Опис
Короткий опис	Варіант використання дозволяє користувачу видаляти замовлення
Акторы	Адміністратор
Передумови	Зареєстрований в системі користувач
Основний потік подій	Видалення замовлення із бази даних
Альтернативний потік подій	Немає
Послідуюче	Інформацію про замовлення буде видалено

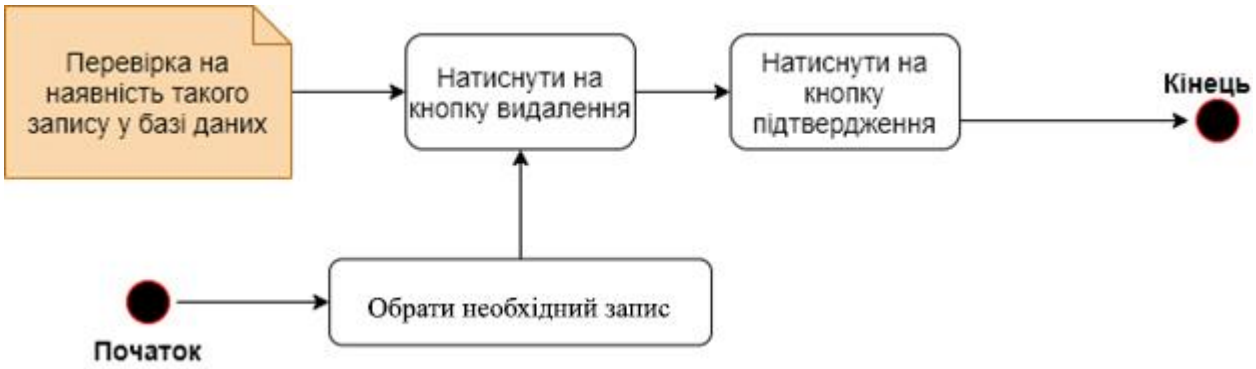


Рис. 3.17. Діаграма діяльності для прецеденту «Видалення замовлення»

Таблиця 3.9

Прецедент «Редагування ролей користувача»

Характеристика	Опис
Короткий опис	Варіант використання дозволяє адміністратору редагувати ролі користувачів
Акторы	Адміністратор
Характеристика	Опис
Передумови	Зареєстрований в системі користувач
Основний потік подій	Редагування ролі користувача у системі
Альтернативний потік подій	Немає
Послідуюче	Відредаговану інформацію про роль користувача

Рис. 3.18. Діаграма діяльності для прецеденту «Редагування ролі користувача в системі»

Таблиця 3.10

Прецедент «Формування звітності»

Характеристика	Опис
Короткий опис	Варіант використання дозволяє адміністратору отримати звітність
Акторы	Активний суб’єкт Адміністратор
Передумови	Зареєстрований в системі користувач
Характеристика	Опис
Основний потік	Формування звітності у форматі pdf
Альтернативний потік подій	Немає
Послідуюче	Буде сформовано звіт

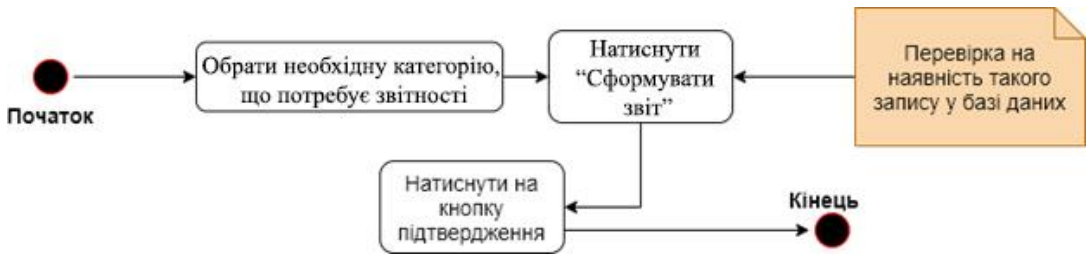


Рис. 3.19. Діаграма діяльності для прецеденту «Створення звітності»

Таблиця 3.11

Прецедент «Формування повідомлень про акції»

Характеристика	Опис
Короткий опис	Варіант використання дозволяє менеджеру сформувати повідомлення про акції
Акторы	Менеджер
Передумови	Зареєстрований в системі користувач
Основний потік подій	Додавання власної категорії користувача до списку
Альтернативний потік подій	Немає
Послідуюче	Буде надіслано повідомлення про акцію

Рис. 3.20. Діаграма діяльності для прецеденту «Формування повідомлення про акції»

3.2.2 Логічна схема даних

Для виниклої потреби в зручному представлені елементів розроблюваної системи постала необхідність розробки діаграми класів [29].

Діаграма класів – це вид статичних структурованих діаграм, які мають описувати структуру системи за допомогою демонстрації класів цієї системи, атрибутів, методів та зв’язків між об’єктами. Зазвичай діаграму класів застосовують при прямому проектуванні, тобто під час процесу розробки нової системи, і при зворотному проектуванні – опис існуючих і частин, що

використовуються. Інформація з таких діаграм класів безпосередньо перетворюється в вихідний код програми. Варто зазначити, що в більшості існуючих інструментів UML-проектування можлива генерація коду для певної мови програмування. Діаграма класів є основним засобом об’єктно-орієнтованого моделювання. Її використовують для загального концептуального моделювання систематизації програми, а також для детального моделювання для подальшого перетворення моделі в код програми. Також діаграми класів можуть бути використані для моделювання даних. Класи в діаграмі представляють як основні елементи. За допомогою діаграми класів створюється внутрішня структура системи, також описується спадкування й взаємне положення класів один до одного. Тут описується логічне представлення системи.

Виходячи з цього, діаграма класів – це кінцевий результат проектування і точка початку для роботи процесу роботи всієї програмної реалізації спроектованої інформаційної системи.

Зазвичай на діаграмі класів демонструються асоціації та узагальнення. Кожна асоціації несе в собі інформацію про зв'язок між об’єктами в середині системи. Найбільш часто використовують бінарні асоціації, що пов’язують між собою два класи. Асоціація може мати назву, яка повинна відображати суть зв’язку, що відображається. Окрім назви, асоціація може мати таку характеристику, як множинність. Вона відображає кількість об’єктів кожного класу, які можуть приймати участь в асоціації.

В свою чергу узагальнення на діаграмах класів використовується, щоб показати зв'язок між класом-батьком й класом-нащадком. Воно вводиться на діаграму, коли виникає розповідність того чи іншого класу, а також в тих випадках, коли в системі виявляється декілька класів, що мають схожу поведінку.

Після того як всі можливі прецеденти були описані та діаграми класів побудовані почалась побудова логічної схеми даних системи. Логічна схема або логічна модель даних – це модель даних конкретної предметної області, що виражено незалежно від конкретного продукту керування базами даних або технології, за допомогою якої реалізується зберігання даних. Логічну модель даних простіше всього представити у вигляді Entry Relationship Diagram (ERD), що входить до мови UML.

Під час проектування CRM-систем для контролю за роботою персоналу СТО за рахунок оптимізації процесів взаємодії з клієнтами було побудовано логічну схему. Графічне відображення логічної схеми даних зображена на рис. 3.21.

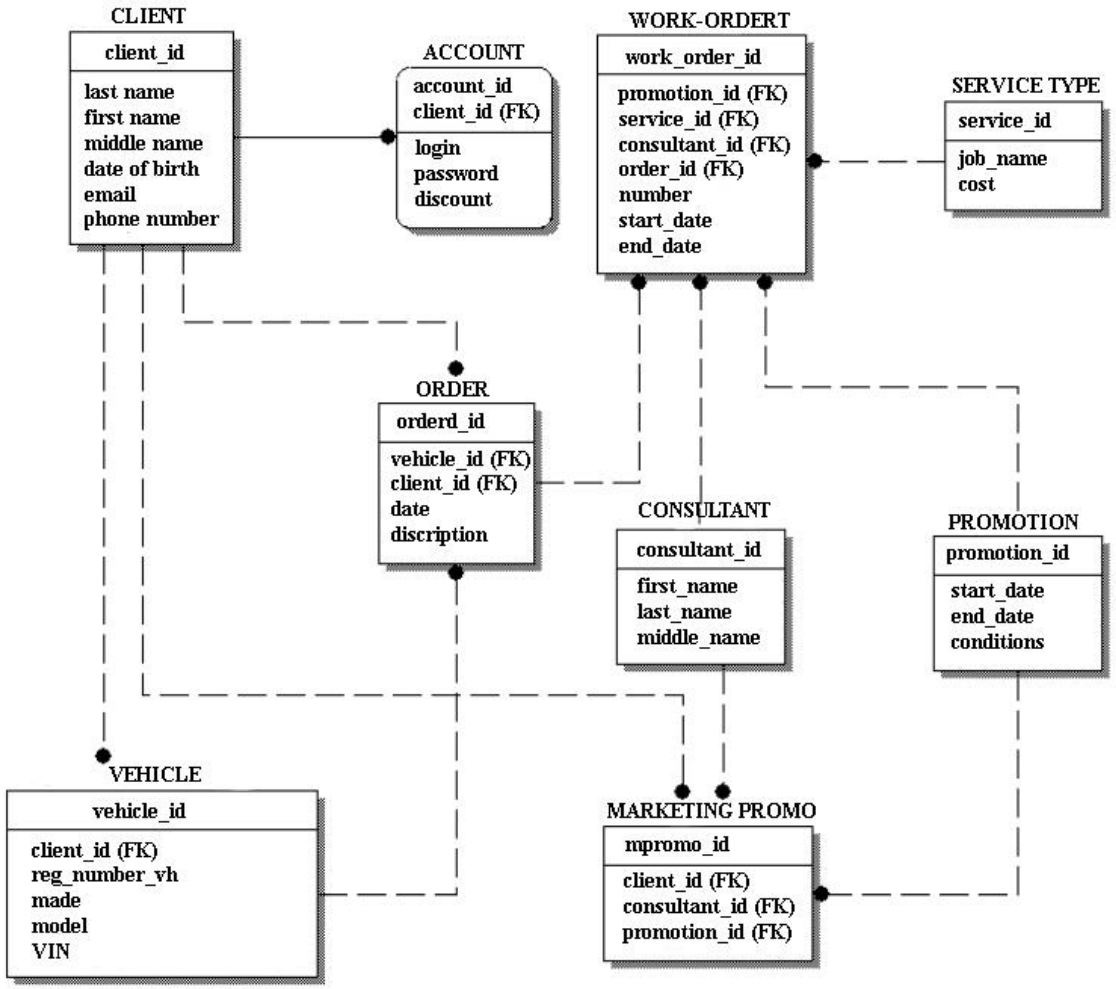


Рис. 3.21. Логічна схема даних CPM-системи

Одразу після чого постала необхідність розробити фізичну схему даних, що буде використовуватися у якості схеми для системи керування базами даних, яку буде обрано пізніше[30]. Результат розробки фізичної схеми даних зображено на рисунку 3.22

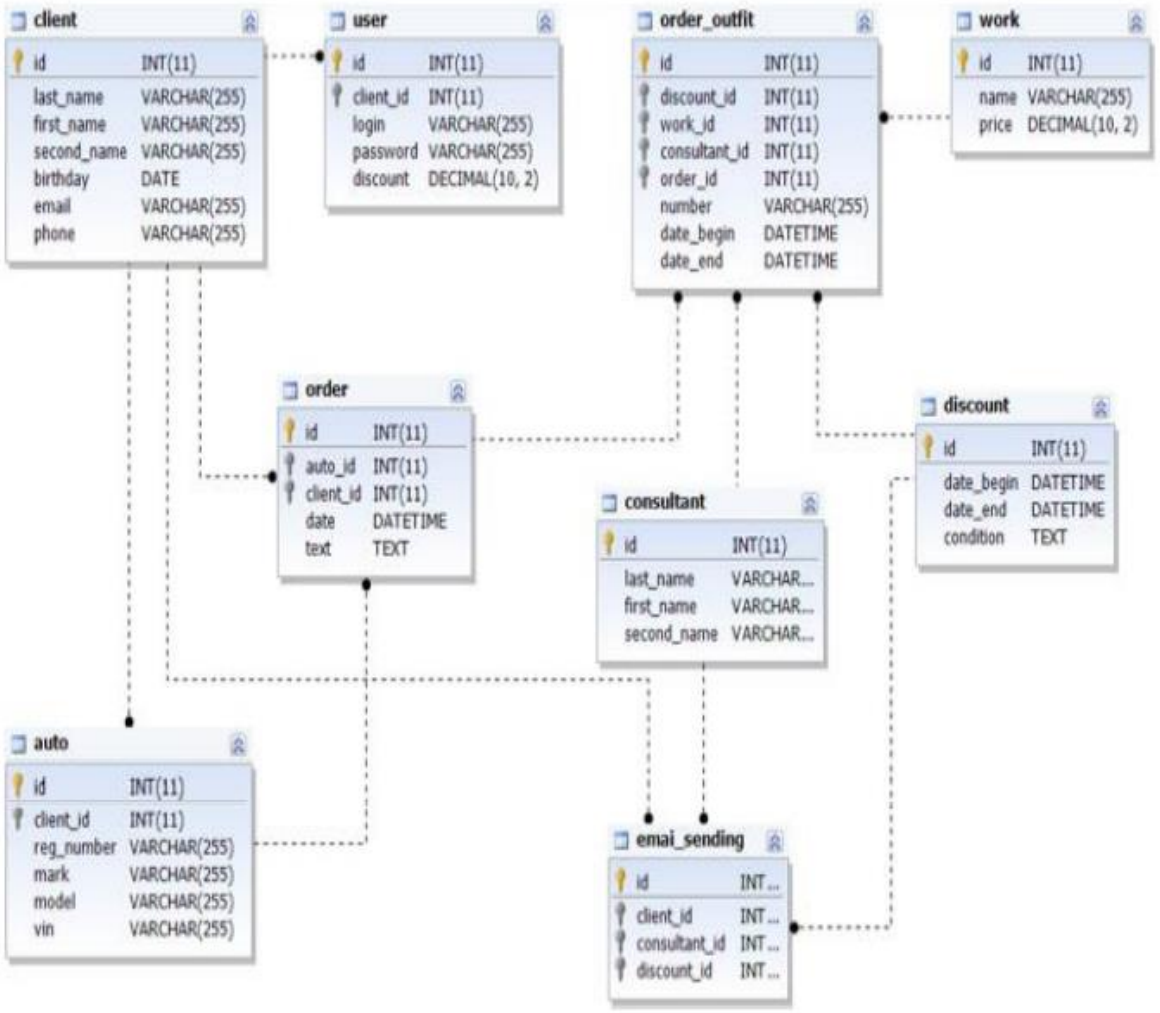


Рис. 3.22. Фізична схема даних CPM-системи для контролю за роботою персоналу СТО

Тепер постала необхідність описати атрибути головних сутностей: користувач, їжа, категорії та ціль. Опис цих сутностей представлений у вигляді табл.. 3.12 – 3.18.

Таблиця 3.12

Сутність «Клієнт»

№	Назва поля	Тип даних	Первинний ключ	Унікальний ключ
1	id	INTEGER(11)	+	+
2	first_name	VARCHAR(225)	-	-
3	last_name	VARCHAR(225)	-	-
4	middle_name	VARCHAR(225)	-	-
5	birthday	DATE	-	-
6	email	VARCHAR(20)	-	-
7	phone	VARCHAR(225)	-	-

Таблиця 3.13

Сутність «Користувач»

№	Назва поля	Тип даних	Первинний ключ	Унікальний ключ
1	id	INTEGER(11)	+	+
2	client_id	INTEGER(11)	-	+
3	login	VARCHAR(225)	-	-
4	password	VARCHAR(225)	-	-
5	discount	DECIMAL (10,2)	-	-

Таблиця 3.14

Сутність «Менеджер»

№	Назва поля	Тип даних	Первинний ключ	Унікальний ключ
1	id	INTEGER(11)	+	+

№	Назва поля	Тип даних	Первинний ключ	Унікальний ключ
2	first_name	VARCHAR(225)	-	+
3	last_name	VARCHAR(225)	-	-
4	middle_name	VARCHAR(225)	-	-

Таблиця 3.15

Сутність «Транспортний засіб»

№	Назва поля	Тип даних	Первинний ключ	Унікальний ключ
1	id	INTEGER(11)	+	+
2	client_id	INTEGER(11)	-	+
3	reg_number	VARCHAR(225)	-	-
4	mark	VARCHAR(225)	-	-
5	model	VARCHAR(225)	-	-
6	VIN	VARCHAR(20)	-	-

Таблиця 3.16

Сутність «Email кампанія»

№	Назва поля	Тип даних	Первинний ключ	Унікальний ключ
1	id	INTEGER(11)	+	+
2	client_id	INTEGER(11)	-	+
3	consultant_id	INTEGER(11)	-	+
4	discount_id	INTEGER(11)	-	+

Таблиця 3.16

Сутність «Email кампанія»

№	Назва поля	Тип даних	Первинний ключ	Унікальний ключ
1	id	INTEGER(11)	+	+
2	client_id	INTEGER(11)	-	+
3	consultant_id	INTEGER(11)	-	+
4	discount_id	INTEGER(11)	-	+

Таблиця 3.17

Сутність «Замовлення»

№	Назва поля	Тип даних	Первинний ключ	Унікальний ключ
1	id	INTEGER(11)	+	+
2	auto_id	INTEGER(11)	-	+
3	client_id	INTEGER(11)	-	+
4	date	DATE	-	-
5	text	TEXT	-	-

Таблиця 3.18

Сутність «Наряд-замовлення»

№	Назва поля	Тип даних	Первинний ключ	Унікальний ключ
1	id	INTEGER(11)	+	+
2	discount_id	INTEGER(11)	-	+
3	work_id	INTEGER(11)	-	+
4	consultant_id	INTEGER(11)	-	+
5	order_id	DATE	-	-

№	Назва поля	Тип даних	Первинний ключ	Унікальний ключ
6	number	VARCHAR(225)	-	-
7	date_begin	DATETIME	-	-
№	Назва поля	Тип даних	Первинний ключ	Унікальний ключ
8	date_end	DATETIME	-	-

Слід зазначити, що створена логічна схема даних може бути легко реалізована й імплементована використовуючи реляційну СКБД, так як вона представлена у вигляді таблиць-сутностей та має зовнішні зв’язки.

Висновки до розділу 3

В цьому розділі дипломної роботи головним наміром було проектування системи та логічної схеми даних. Під час опису даного розділу було спроектовано та створено функціональну модель інформаційної системи, представлену у вигляді нотації IDEF0, а також безпосередньо визначено перелік головних функціональних блоків системи на основі її функціональності, які в свою чергу були визначені в якості голових вимог, побудовано дерево функцій системи, проведено декомпозицію функціональної моделі на функціональні блоки за допомогою нотації IDEF0.

На основі висновків після проведеної декомпозиції функціональної моделі було визначено множину прецедентів, актора, зв’язків між ними, побудовано діаграму прецедентів й діаграму діяльності. Використовуючи таблиці, для кожного прецеденту, було виконано детальний опис та надано характеристику, в якій зазначено назву прецеденту, актора, умови, додаткових факторів, необхідних передумов, вихідних умов, нормальний напрямок виконання прецеденту, а також альтернативні напрямки виконання та можливі виключення в роботі системи.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА РОЗРОБКА ДОКУМЕНТАЦІЇ

4.1 Вибір технологій розробки програмного забезпечення

Після детального аналізу існуючих підходів та технологій для вирішення поставленої задачі було обрано такі технології та засоби програмної реалізації, які допоможуть швидко, зручно та легко почати й підтримувати процес розробки програмної реалізації та довести розробку CRM-системи для контролю за роботою персоналу СТО за рахунок оптимізації процесів взаємодії з клієнтами.

4.1.1 Фреймворк Django

Django - це програмний каркас з багатими можливостями, що підходить для розробки складних сайтів та веб-застосунків, написаний мовою програмування Python.

Django — фреймворк для веб-застосунків мовою Python. Один з основних принципів фреймворку - DRY (don't repeat yourself). Веб-системи на Django будуються з одного або кількох програм, які рекомендується робити відчужуваними та підключається. Це одна з помітних архітектурних відмінностей цього фреймворку від інших (наприклад, Ruby on Rails). Також, на відміну від багатьох інших фреймворків, обробники URL Django конфігуруються явно (за допомогою регулярних виразів), а не автоматично задаються зі структури контролерів[31].

Django проектувався для роботи під управлінням Apache (з модулем mod_python) та з використанням PostgreSQL як база даних. В даний час, окрім PostgreSQL, Django може працювати з іншими СКБД: Firebird, SQL Anywhere, SQLite, Microsoft SQL Server, MySQL (MariaDB), DB2, та Oracle. Для роботи з базою даних Django використовує власний ORM, у якому модель даних описується класами Python, і генерується схема бази даних.

Архітектура Django «Модель-Уявлення-Контролер» (MVC). Контролер класичної моделі MVC приблизно відповідає рівню, який Django називається Представлення (View), а презентаційна логіка Представлення реалізується в Django рівнем Шаблонів (Templates). Через це рівневу архітектуру Django часто називають "Модель-Шаблон-Представлення" (MTV).

Спочатку розробка Django велася для забезпечення зручнішої роботи з ресурсами новин, що досить сильно позначилося на архітектурі: фреймворк надає ряд засобів, які допомагають у швидкій розробці веб-сайтів інформаційного характеру. Наприклад, розробнику не потрібно створювати контролери та сторінки для адміністративної частини сайту, у Django є вбудований додаток для керування вмістом, який можна включити в будь-який сайт, зроблений на Django, і який може керувати відразу кількома сайтами на одному сервері. Адміністративна програма дозволяє створювати, змінювати та видаляти будь-які об'єкти наповнення сайту, протоколюючи всі вчинені дії, та надає інтерфейс для управління користувачами та групами (з пооб'єктним призначенням прав) [32].

Деякі компоненти фреймворку між собою пов'язані слабко, тому їх можна просто замінювати на аналогічні. Але з деякими (наприклад, ORM) це зробити не дуже просто. Крім можливостей, вбудованих у ядро фреймворку, існують пакети, що розширюють його можливості.

На базі Django розроблено досить багато готових рішень, що розповсюджуються під вільною ліцензією, серед яких системи управління інтернет-магазинами, універсальні системи управління змістом, а також більш вузькоспрямовані проекти.

Django створили розробники видання Lawrence-Journal World. Цій газеті знадобився сайт, щоб публікувати новини в Інтернеті. Програмісти Едріан Головатий та Саймон Віллісон створили веб-додаток і зробили його публічним.

Навколо Django швидко сформувалася активна спільнота. Фреймворк став швидко розвиватися зусиллями волонтерів. Значну роль у успіху Django відіграли кілька відомих сайтів, які використали цей фреймворк. До них входять Pinterest, Dropbox, Spotify, сайт The Washington Post. В даний час співтовариство Django включає понад 11 тис. розробників із 166 країн світу.

У Django реалізований принцип DRY (don't repeat yourself). Завдяки цьому скорочується час створення веб-сайтів. Тобто при використанні Django вам не потрібно кілька разів переписувати один і той самий код. Фреймворк дозволяє створювати сайт із компонентів[33].

Фреймворк Django написаний мовою програмування Python, тому його структура відповідає особливостям мови. Творці продали в Django патерн MVC, і він застосовується в поточній версії фреймворку.

Архітектура MVC дозволяє розробнику працювати з візуальним поданням та бізнес-логікою програми окремо. До речі, під час роботи з Django фахівці частіше використовують термін MVT — Model-View-Template або модель-вистава-шаблон. Компоненти MVT можна використовувати незалежно один від одного.

Подання (view) вирішує три завдання: приймає HTTP-запити, реалізує бізнес-логіку, визначену методами та властивостями, надсилає HTTP-відповідь у відповідь на запити. Тобто уявлення отримує дані від моделі та надає шаблонам (templates) доступ до цих даних або попередньо обробляє дані та потім надає до них доступ шаблонів.

У Django реалізований потужний двигун шаблонів та власна мова розмітки. Шаблони є файлами з HTML-кодом, за допомогою якого відображаються дані. Вміст файлів може бути статичним або динамічним. Шаблони не містять бізнес-логіки. Тому вони лише відображають дані.

Досвідчені розробники радять сприймати Django як систему. Це означає, що фреймворк зазвичай використовується з великою кількістю сторонніх програм. Їх можна обирати залежно від потреб конкретного проекту.

Щоб краще зрозуміти цей принцип, представте конструктор Lego. У ньому є багато типових блоків. У Django також є типові блоки. Наприклад, блок авторизації або блок підписки на розсилку застосовується практично у кожному проекті. Створені за допомогою фреймворку веб-програми складаються з таких незалежних блоків.

Адміністративна панель Django автоматично генерується під час створення програми. Це позбавляє розробника необхідності створювати адмінку вручну.

За допомогою сторонніх програм дефолтну консоль управління Django можна вдосконалити та адаптувати під потреби свого проекту. Крім того, фреймворк дозволяє настроювати інтерфейс дефолтної адміністративної панелі.

Функціональність Django розширюється за допомогою плагінів. Це програмні модулі, що дозволяють швидко додати на сайт потрібну функцію. В офіційному каталозі є сотні плагінів, які дозволяють легко реалізувати на сайті sitemap.xml, керувати доступами, підключити платіжну систему Stripe і таке інше. При необхідності ви можете вимкнути або замінювати плагіни, щоб адаптувати програму до поточних потреб проекту.

Опираючись на цю інформацію було обрано фреймворк Django завдяки таким характеристикам:

- Поділ бізнес-логіки та візуальної частини на рівні архітектури.
- SEO-дружність.
- Розширюваність.
- Розвинена інфраструктура: велика кількість бібліотек та плагінів.

4.1.2 Бібліотека TelegramBotAPI

На даний момент є два основні інструменти API, за допомогою яких можна використовувати сервіси Telegram — Telegram Bot API і Telegram API. Перший служить для розробки чат-ботів, другий дозволяє робити повністю кастомні телеграм-клієнти. Розробникам також доступна відкрита бібліотека TDLib (Telegram Database Library), за допомогою якої можна створювати свою версію месенджера з унікальними опціями (наприклад, Telegram X, побудований саме на TDLib). Telegram Bot API є надбудовою над Telegram API, тому користуватися Bot API можна без знань про механізм використовуваного протоколу MTProto.

Для його роботи задіяний проміжний сервер з інтерфейсом HTTPS, який шифрує трафік і забезпечує зв'язок з Telegram API. Bot API дозволяє легко створювати програми, які використовують інтерфейс Telegram для виконання коду на локальному сервері. Користувачі можуть взаємодіяти з ботами, надсилаючи їм повідомлення, команди та вбудовані запити.

Принцип роботи будь-якого бота полягає в тому, що він перманентно надсилає запити на сервер і регулярно отримує оновлення. Отримувати їх можна двома способами. По-перше, можна використовувати вебхуки, коли сервер здійснює зворотний дзвінок на вказаний URL. По-друге, можна просто «закидати» запитами Telegram, отримуючи постійні відповіді.

Для будь-якого бізнесу важливо встановити канал спілкування із клієнтами. Але зробити це не так просто. Психологія людей така, що вони не хочуть захаращувати пам'ять свого смартфона новим фірмовим додатком із того місця, яке вони відвідали. Зовсім інша справа – чат-бот. Ненав'язливий та доброзичливий. А крім того, з усіма необхідними фішками: рекламними акціями, знижками та швидким замовленням. І повідомлення в одному єдиному зручному месенджері.

Щоб розпочати створення власного бота, необхідно отримати токен для авторизації та підключення через API. Робиться це за допомогою службового бота.

BotFather запитатиме вас ім'я, яке обов'язково має закінчуватися на bot, наприклад, cto_crm_bot. Далі для бота буде згенеровано унікальний токен.

Якщо Telegram використовувати у комерційних цілях, чат-бот можна озброїти засобами прийому платежів. Варто звернути увагу, що сам Telegram не займається проведенням транзакцій, він лише дозволяє підключити послуги довгого списку провайдерів.

Серед них такі платіжні системи, як Stripe, YooMoney, Ощадбанк, PayMaster, PSB, Tranzzo, Payme, CLICK, LiqPay, Portmone, Paymega, ECOMMPAY та ін. Зрозуміло, щоб використовувати ці платіжні системи, потрібно бути юридичною особою.

Telegram зберігає всі дані, тобто всі чати, а також ботів в хмарі. Таким чином, зовнішня резервна копія даних Telegram не є абсолютно необхідною, і всі особисті налаштування доступні користувачам, які ввійшли в систему, на різних платформах у будь-який час і скрізь. Однак конфіденційні дані та команди, до яких бот потім звертається ззовні, можуть бути доступні за межами хмари, напр. на локальних серверах компанії у власних базах даних.

Після отримання токена, переходимо в середовище розробки. Слід зазначити, що процес розробки відбувається на мові програмування Python у середовищі PyCharm.

Необхідно встановити бібліотеку у віртуальне середовище після чого ми отримаємо доступ до у всіх методів та класів бібліотеки. Процес розробки не включає в себе чогось надто важкого і по безкоштовним матеріалам з мережі Інтернет можна легко зорієнтуватися.

Оскільки бот напряму функціонує з фреймворком Django, панель адміністрування також буде знаходитись на віртуальному сервері. Установка

Webhook полягає в передачі боту спеціального URL адреси на який надходитиме POST запит кожного разу, коли хтось почне посилати повідомлення боту. Саме цей варіант ми і використовуватимемо для взаємодії між ботом та його користувачем. Для того, щоб задати URL-адресу, необхідно використовувати API метод setWebhook. Відзначу, що URL повинен починатися з https, тобто мати захищене SSL з'єднання з сертифікатом. Telegram дозволяє використовувати самопідписаний сертифікат, правда для цього потрібно в способі setWebhook передавати також громадський ключ у PEM форматі (ASCII base64). Або можна отримати валідний безкоштовний SSL сертифікат від Let's Encrypt.

Отже, повернемося до бібліотеки python для роботи з Telegram - telepot. На даний момент останньої її версій є 6.7. Встановлюємо її у віртуальне оточення python virtualenv.

CommandReceiveView чекає POST запит на себе, парсить його та відповідає виходячи із заданої команди. Повноцінне Django програму можна знайти за цим посиланням. Варто зазначити в коді використання ще одного API виклику – sendMessage. Цей метод надсилає повідомлення заданому користувачеві, використовуючи при цьому chat_id і текст повідомлення. Chat_id – це унікальний ідентифікатор чату між користувачем та ботом (його ідентифікатор є у відповіді на запит getUpdates). Telegram боти мають одне обмеження, вони не можуть надсилати повідомлення користувачам, які попередньо не ініціювали спілкування з ним. Очевидно це зроблено, щоб уникнути масового створення спам-ботів.

Мета розробки бота для системи CRM доволі проста. Необхідно реалізувати увесь важливий функціонал системи у компактному вигляді всередині месенджеру.

План розробки полягав у тому, що треба відштовхуватися від роботи повноцінної системи. Іншими словами, функціонал бота також починається з авторизації, що зображено на рисунку 4.1.

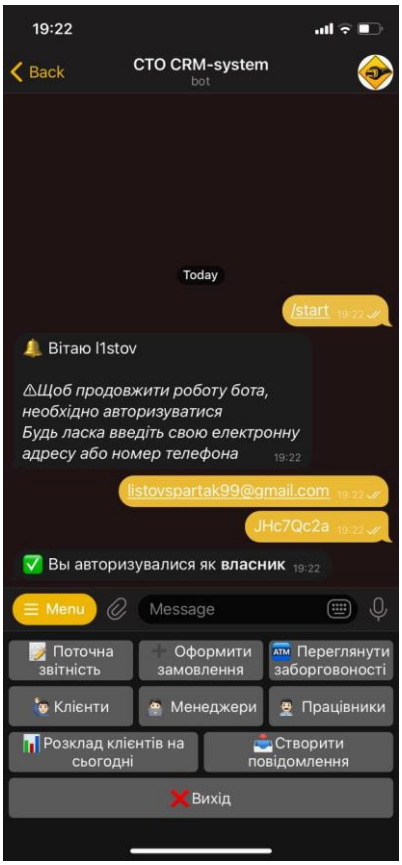


Рис. 4.1. Авторизація у Telegram боті

Оскільки на стороні Django було виділено декілька ролей з різним функціоналом, то й постає задача розробити відповідний функціонал бота для кожної з них. На рисунку відображена панель керування для адміністратора ресурсу.

Було прийнято рішення брати та користуватися усма можливими функціями, що надає до роботи TelegramBotAPI. Для меню клієнтів, менеджерів та працівників була реалізована можливість телефонного дзвінка. Перегляд менеджерів та працівників відображає тільки тих, що так би мовити «онлайн». На рисунку 4.2 відображена можливість перегляду працюючих сьогодні та можливість телефонного дзвінку.

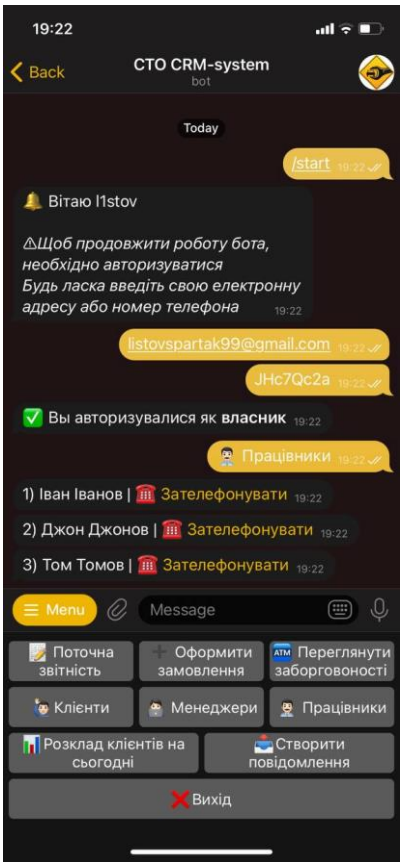


Рис. 4.2. Перегляд працівників та можливість телефонного дзвінку

Для адміністратора сервісу було розроблено можливість перегляду усіх видів звітностей, що він може переглянути у свому власному кабінеті на стороні веб-браузера.

Також він має можливість перегляду заборгованостей, оформлення замовлень тих чи інших розхідних матеріалів та присутня можливість створювати повідомлення. Ці повідомлення будуть приходити активним користувачам бота напряму через нього. Надалі планується переробка такого функціоналу, оскільки є ймовірність, що не ввсі працівники СТО будуть використовувати Telegram.

На рисунку 4.3 відображена можливість перегляду записів клієнтів на сьогоднішнє число. Після обробки запиту на сервер, результат повертається до адміністратора у вигляді pdf-файлу, який буде зручно друкувати та редагувати у майбутньому.

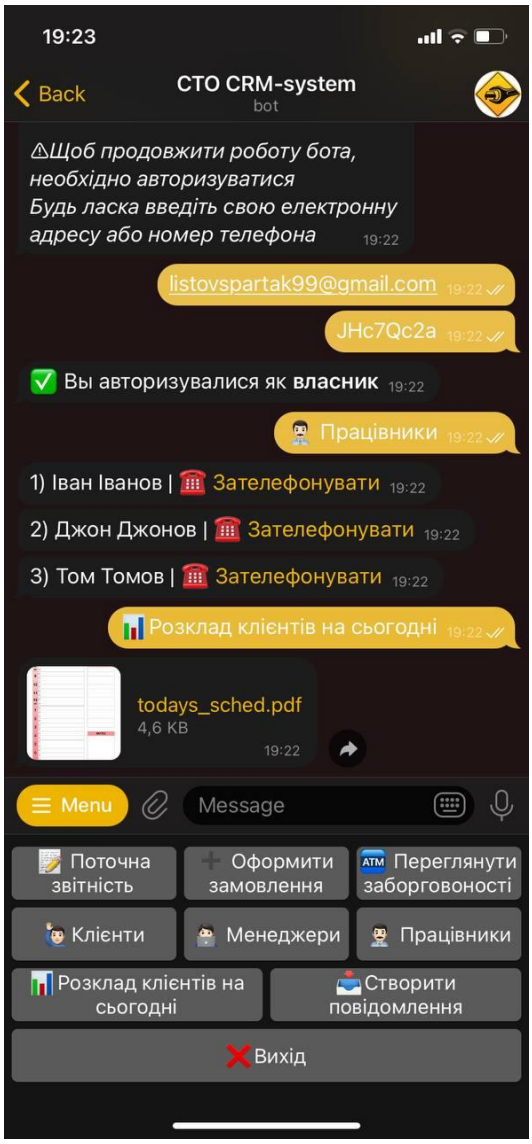


Рис. 4.3. Перегляд записів клієнтів на сьогодні

Після чого була створена можливість перегляду поточної звітності, зараз вона налаштована таким чином, щоб повертати pdf-файл з щотижневим звітом. Реалізація зображена на рисунку 4.4.

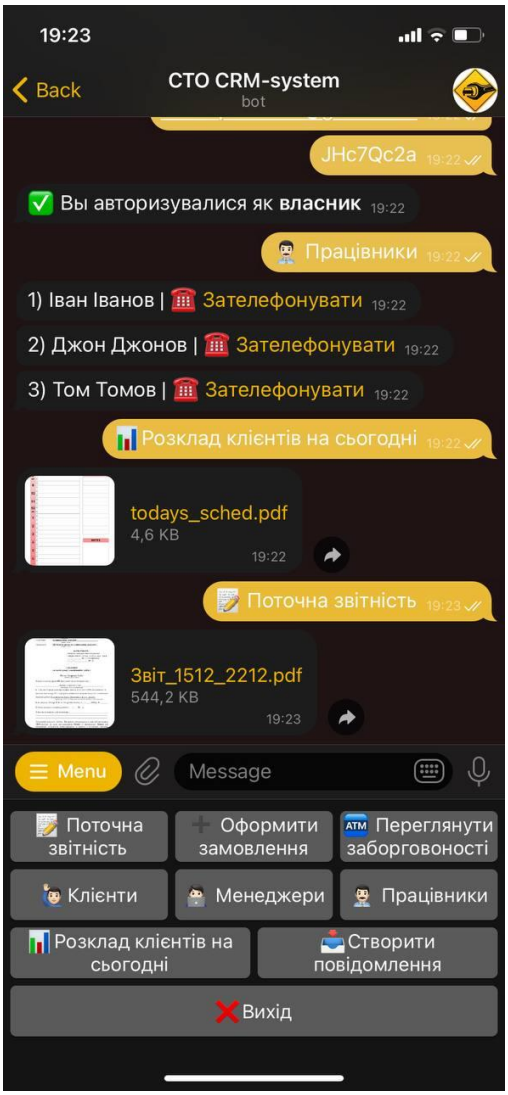


Рис. 4.4. Перегляд поточної звітності

Таким чином, було розроблено зручного Telegram бота, який стане в нагоді кожному причасному до бізнеса працівника.

4.1.2 Система керування базами даних SQLite

На етапі попереднього аналізу було прийнято рішення в якості системи керування базами даних (СКБД) було прийнято рішення використовувати вбудовану компактну систему SQLite. Ця СКБД кроссплатформена та має відкритий код бібліотеки.

Взагалі, слово «вбудована» означає, що SQLite не використовує парадигму клієнт-сервер, тобто SQLite не є окремо працюючим процесом, з яким взаємодія програма, а являє собою бібліотеку, з якою програма

компонується, та сама система SQLite стає основною складовою програми. Таким чином, в якості протоколу обміну використовують виклик функції Application Programming Interface (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати та час відгуку, що в свою чергу призводить до спрощення програми. Система керування базами даних SQLite зберігає в собі таблиці, індекси, данні та визначення, інакше кажучи всю базу даних. Бібліотека зберігає базу даних в стандартному файлі на тому комп'ютері, на якому виконується програма. Така простота реалізації досягається за рахунок того, що перед початком виконання транзакції запису весь файл, що зберігає базу даних, блокується[34].

Декілька процесів або потоків можуть одночасно читати дані з однієї бази даних. Записи в базу даних можна робити тільки в тому випадку, якщо ніяких інших запитів в даний момент не обслуговується.

Також під час аналізу бібліотеки слід ознайомитись з типами даних, що підтримується SQLite та її обмеженнями. Отже, було визначено, що бібліотека підтримує динамічне типізування даних. Можливі типи значень: REAL, INTEGER, TEXT та BLOB. Також підтримується спеціальне значення Null.

Кожне значення в будь-якому полі будь-якого запису може бути будь-якого з цих типів даних, вказаного при ініціалізації полей таблиці. Вказаний під час ініціалізації поля тип даних зберігається в його вихідному написанні, та використовується в якості основи для підходу «type affinity», при виконанні неявних перетворень типів на основі схожості назви типу.

Бібліотека JavaFX включає в себе ряд обмежень. Старі версії SQLite були спроектовані без обмежень, єдиною умовою було те, щоб база даних вміщалась в пам'яті, в яких всі розрахунки проводились за допомогою 32-розрядних цілих чисел. Це викликало деякі проблеми. Із-за того, що верхні межі не були визначені й відповідно певним образом протестовані, часто

з’ясовувались помилки при використанні SQLite в доволі екстремальних умовах. Тому в нових версіях були введені межі, які тепер проводяться разом із загальним набором тестів. Декілька з основних обмежень продемонстровані в табл. 4.1.

Таблиця 4.1

Основні обмеження платформи SQlite

Опис	Значення	Константа в вихідному коді
Максимальна кількість колонок	2 000	SQLITE_MAX_COLUMN
Максимальна довжина SQL-виразу	1 000 000 000	SQLITE_MAX_SQL_LENGTH
Максимальна кількість таблиць в виразом з JOIN	64	
Максимальна кількість аргументів функції	127	SQLITE_MAX_FUNCTION_ARG
Максимальний розмір сторінки бази даних	65 536	SQLITE_MAX_PAGE_SIZE
Максимальна кількість сторінок в файлі бази даних	1 073 741 823	SQLITE_MAX_PAGE_COUNT

Під час компіляції бібліотеки SQLite встановлюються наступні обмеження, які можна, при гострій необхідності корегувати.

Сама система керування базами даних SQLite була написана на мові програмування C, проте існує велика кількість прив’язок до інших мов програмування. Простота вбудовування бібліотеки привели до того, що вона використовується навіть в браузерах, музикальних програвачах й в багатьох інших програмах та фреймфорках.

4.2 Вибір середовища розробки

Популярність Python як мови програмування змінюється рік у рік. Python став мовою року в 2007 і 2010 роках (TIOBE), та й зараз стабільно входить до десятки - а то й п'ятірки - найпопулярніших мов програмування.

Python люблять за лаконічний код, який легко зрозуміти, низький поріг входження та можливість використовувати цю мову практично для будь-яких завдань. Python має велику спільноту, її використовують у багатьох компаніях зі світовим ім'ям: Google, Facebook, Microsoft, Intel тощо.

IDE - Integrated development environment - інтегроване середовище розробки, комплекс програмних засобів, які дозволяють вести зручнішу розробку певною мовою програмування. Зазвичай IDE має текстовий редактор, компілятор або інтерпретатор, налагоджувач та інше програмне забезпечення.

IDE дозволяє збільшити швидкість розробки (за умови попереднього навчання роботі з IDE, звичайно).

PyCharm – це інтегроване середовище розробки для Python, яке має повний комплект засобів, необхідних для ефективного програмування на Python.

Зараз PyCharm поширюється у двох варіантах: платному (PyCharm Professional Edition) та безкоштовному (PyCharm Community Edition).

Безкоштовна версія має відкритий вихідний код і розповсюджується під ліцензією Apache 2. Це полегшене середовище, яке підходить для розробки лише на Python.

PyCharm Professional – це платна версія PyCharm з величезною кількістю готових функцій та можливостей інтеграції. У цьому розділі в основному буде представлено огляд основних функцій та посилання на офіційну документацію, де кожна функція детально обговорюється.

Пам'ятайте, що жодна з наведених нижче функцій не доступна у версії Community.

Платний варіант є більш розширеною та функціональною версією з можливістю розробки в тому числі багатомовних веб-додатків. Professional Edition підтримує різні фреймворки.

Серед переваг варто виділити PyCharm має зручний редактор коду з усіма корисними функціями: підсвічуванням синтаксису, автоматичним форматуванням, доповненням та відступами. PyCharm дозволяє перевіряти версії інтерпретатора мови на сумісність та використовувати шаблони коду.

Тим, хто часто використовує документацію, буде зручно дивитися її у вікні редактора (для елементів) або в браузері (для зовнішньої документації). PyCharm дозволяє швидко проводити рефакторинг коду, а також використовувати зручний графічний налагоджувач. Утиліта підтримує всі нові версії Django, а також IronPython, Jython, Cython, PyPy wxPython, PyQt, PyGTK та багато інших інструментів.

У PyCharm можна проводити інтегроване Unit тестування, використовувати інтерактивні консолі для Python, Django, SSH, відладчика та баз даних. PyCharm має велику колекцію плагінів, і його можна використовувати у зв'язці з різними трекерами на зразок JIRA, Youtrack, Lighthouse, Redmine, Trac і таке інше. Варто відзначити, що ця IDE крос-платформне середовище розробки: можна використовувати на Linux, Windows та Mac OS.

PyCharm можна назвати одним з найкращих IDE для Python. Залежно від своїх можливостей та потреб можна вибрати або платну професійну версію, або безкоштовну версію для спільноти.

4.3 Створення інтерфейсу застосунку

Наступним кроком в розробці CRM-системи для контролю за роботою персоналу СТО за рахунок оптимізації процесів взаємодії з клієнтами стане розробка графічного інтерфейсу[35]. Розробку графічного інтерфейсу слід розпочати з вікна реєстрації нового користувача або авторизації в системі. Використовуючи всі необхідні інструменти та можливості Django та шаблоні затору Jinja був розроблений макет вікна реєстрації для нового користувача. Вигляд цієї сторінки наведено на рис. 4.5.

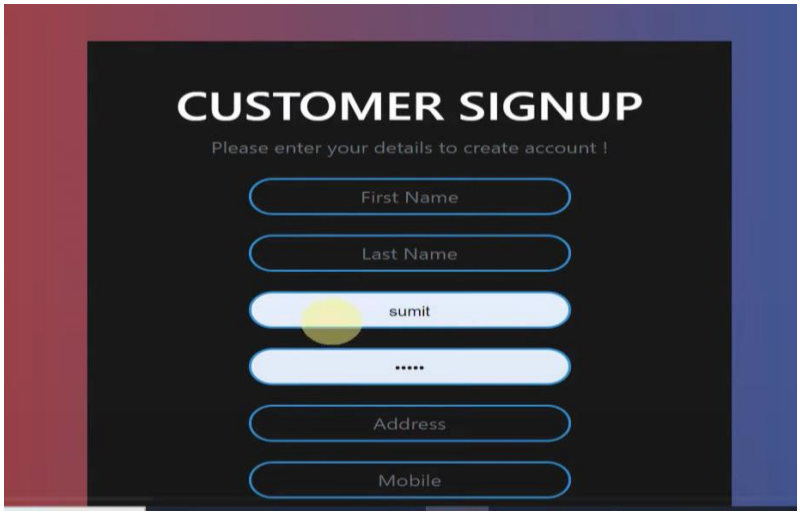


Рис. 4.5. Форма авторизації у системі

Особливу увагу було звернуто на можливість редагування власних данних безпосередньо клієнтом компанії.Оскільки інформація завжди змінюється і наприклад у клієнта сьогодні може бути зовсім інший засіб пересування від того, яким він користувався учора. Впровадження такого роду функціоналу являється доречним. Приклад реалізації відображено на рису 4.6.

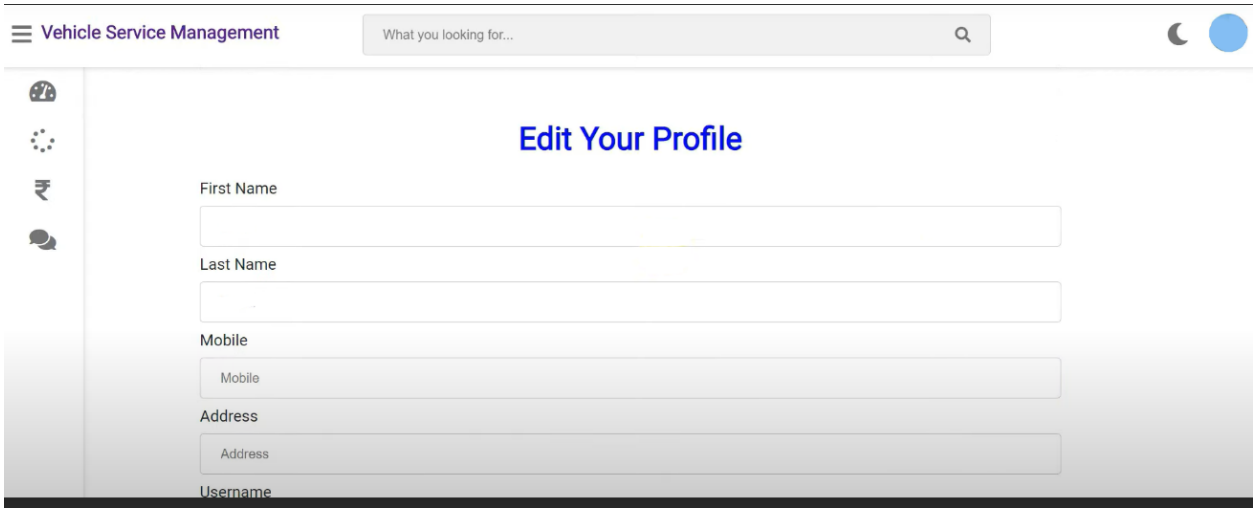


Рис. 4.6. Редагування профілю користувача

У правому верхньому куті на попередньому рисунку можна знайти випадаюче меню, яке було створено для зручної навігації по системі. Ілюстрація меню на рису 4.7

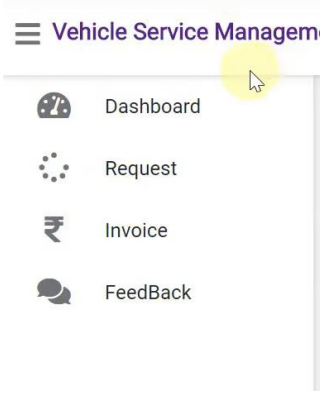


Рис. 4.7. Випадаюче меню

Опція запити віповідає за збереження, контроль та перегляд усіх існуючих зверень власника авто або клієнта до сервісу. Релізована можливість перегляду поточних запитів, схвалених або прийнятих запитів та можливість сторити новий. Ілюстрація реалізації такого функціоналу графічно відображена на рисунку 4.8.

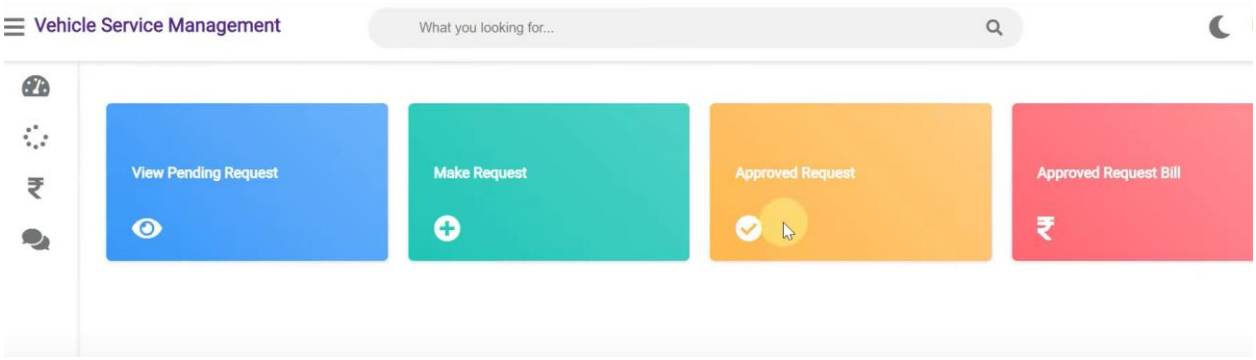


Рис. 4.8. Перегляд запитів клієнта

При бажанні створення нового звернення до сервісу відкриється форма в якій буде необхідно заповнити усі поля та описати причину. Після чого, менеджер сервісу звернеться до клієнта. Форма зображена на рисунку 4.9.

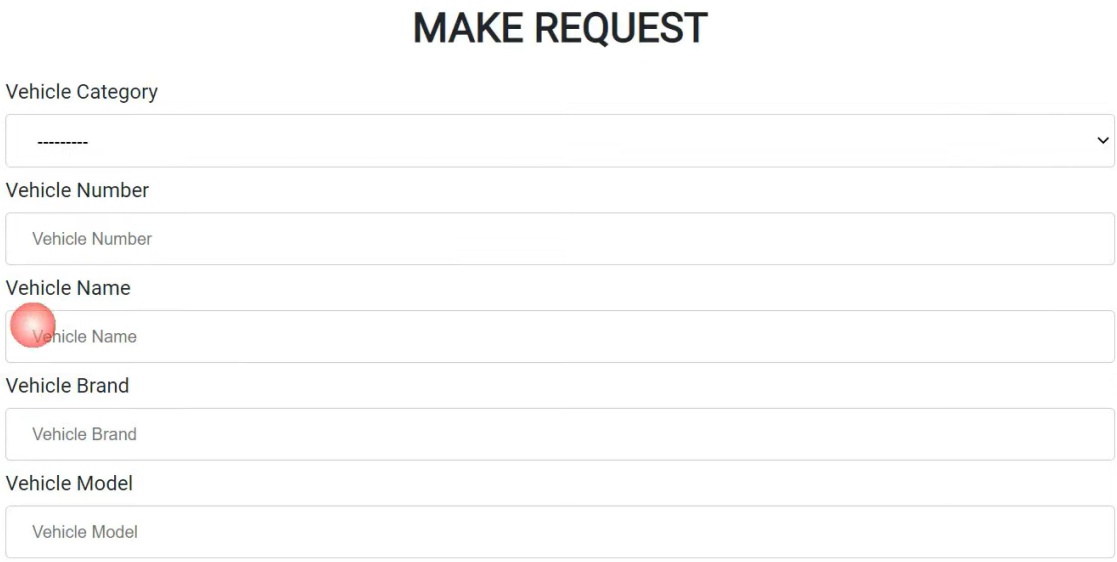


Рис. 4.9. Форма створення нового звернення до СТО

Після створення замовлення, його можна бути знайти на етапі «Оброблюється». У клієнта є можливість видалити своє звернення зі своєї сторони. Реалізація на рисунку 4.10

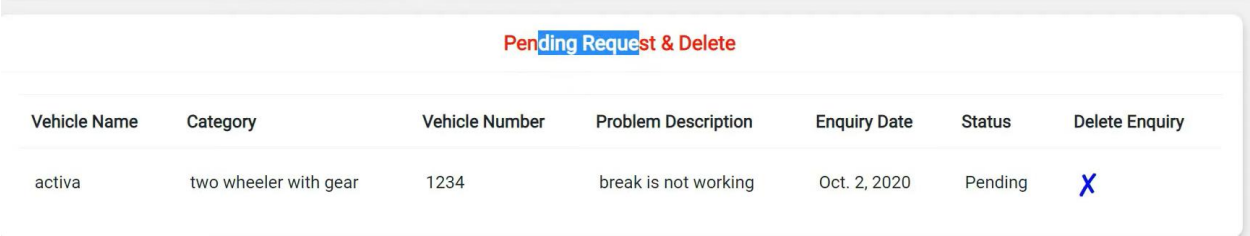


Рис. 4.10. Меню замовлень клієнта

Після розробки графічного інтерфейсу з яким буде працювати клієнт сервісу було зроблено перехід до розробки форми авторизації менеджера та адміністратора. Функціональність системи або статуси які встановлені системою відображаються для працівників системи схожим чином відображається після вдалої авторизації користувча. Зображено на рисунку 4.11.

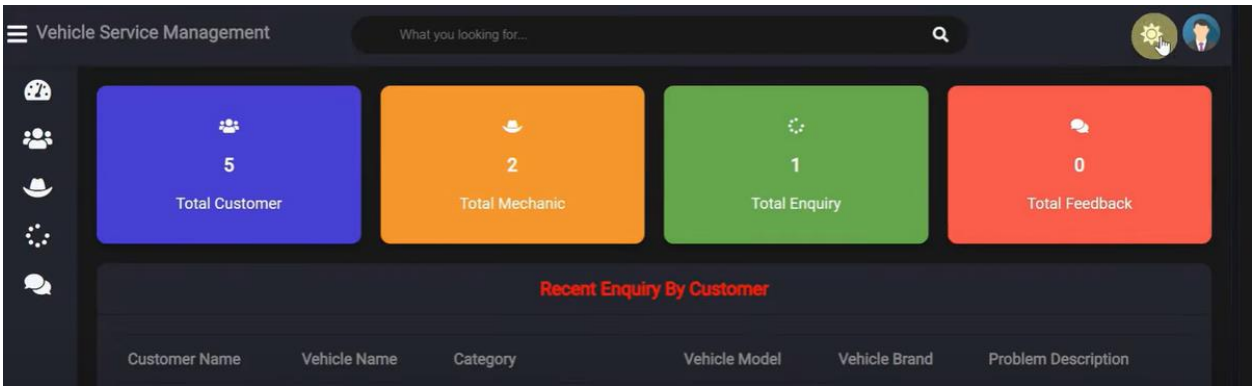


Рис. 4.11. Меню замовлень клієнта

Меню статусів усіх замовлень та користувачів, які відображаються команді працівників після вдалої авторизації. На рисунку зображені переходи на сторінки в яких можна переглянути кількість механіків, клієнтів, запитів та відгуків.

Якщо більш детально переглянути меню механіків ми маємо розуміти, що будь-який бізнес дає робочі місця для професіоналів свого діла. Тому, є сенс реалізувати свого роду відділ кадрів. Де ми зможемо переглядати усі

можливі зміни у робочій ланці підприємства. Реалізація на рисунках 4.11 – 4.12

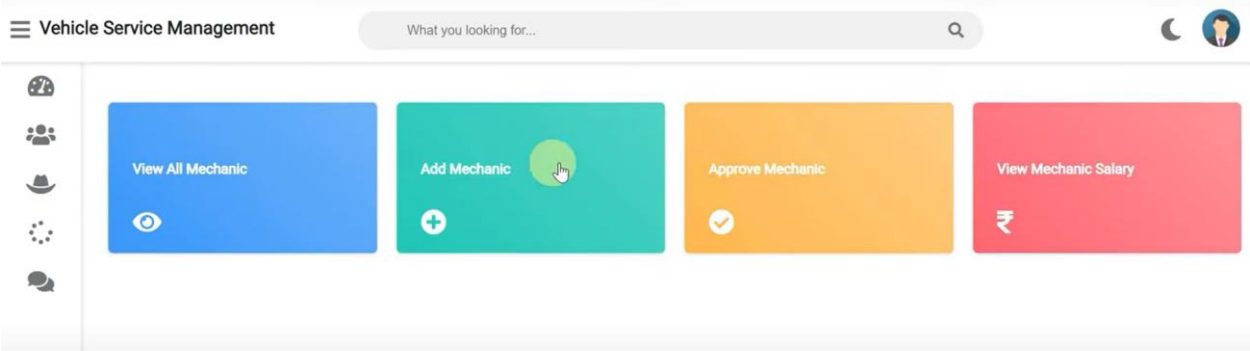


Рис. 4.11. Меню механіків

Рис. 4.12. Списко механіків, яких можна взяти на роботуя

Було розроблено графічний інтерфейс для усіх користувачів системи, а саме клієнтів, менеджерів, механіків та адміністратору підприємства. Оскільки однією з переваг фреймворку Django є шаблонізатор веб-сторінок, то є сенс не відходити від уже встановлених графічних стандартів для усієї системи. Тому, відзначимо, що більшою мірою UI (User Interface) не буде відразінятися для користувачів із разними ролями.

Так, деякі сторінки на були проілюстровані під час написання цього підрозділу магістерської дипломної роботи, саме завдяки цьому. Звісно, в майбутньому планується вдосконалення графічного інтерфейсу користувача і перехід до технології які надають варіативності зовнішньому вигляду сторінки, наприклад Bootstrap Twitter.

На данний момент, «зовнішній вигляд» системи цілком задовольняють усі поставлені задачі.

4.4 Інструкція користувача

Дана система розрахована на власника сто який звик досягати поставленої мети. Тому було прийнято рішення максимально мінімізувати кількість елементів на екрані, щоб не розсіювати увагу працівника, який буде користуватись застосунком на неважливі елементи інтерфейсу. Саме тому, що графічний дизайн застосунку є доволі простим та інтуїтивно зрозумілим, інструкція користувача – не потребує великої кількості уваги.

Детальний огляд на всі можливості веб-застосунку буде наведено у цьому підрозділі.

Перше на що слід звернути уваги, це вікно реєстрації. Оскільки сьогоднішній користувач не сприймає велику кількість інформації на єкрані , було прийнято рішення створити таке вікно реєстрації, щоб не злякати потенційного користувача системи. Беручи до уваги той факт, що системою будуть користуватися вже ті менеджери, яким була видана певна роль у системі через консоль адміністратора, форма авторизації має вигляд який зображено на рисинку 4.13.

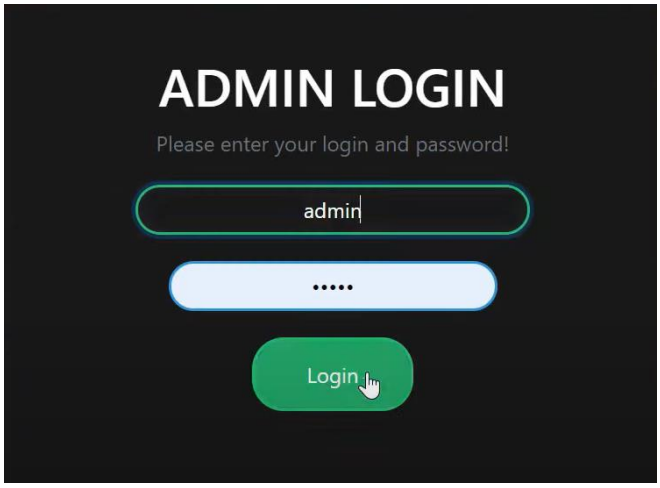


Рис. 4.13. Вікно авторизації

Після авторизації, менеджеру або власнику надається можливість користуватись функціоналом веб-мобільного застосунку в повній мірі. Наприклад, якщо власник станції технічного обслуговування отримає

повідомлення про запит будь-якого працівника, такий наприклад як запит про підвищення заробітної плати, біля аватару його профіля буде відображено коло, що свідчить про наявність нових повідомлень. Звідки, буде доволі легко орієнтуватись у системі. Приклад зображений на рисунку 4.14.

Рис. 4.14. Схвалення запиту на підвищення заробітної плати

На рисунку проілюстрована можливість для адміністратора підвищити зарплату працівнику за його запитом. А ось на наступній ілюстрації, рисунок 4.15, ми можемо побачити оформлення запиту з боку механіку на наряд-замовлення та вартість роботи для клієнта.

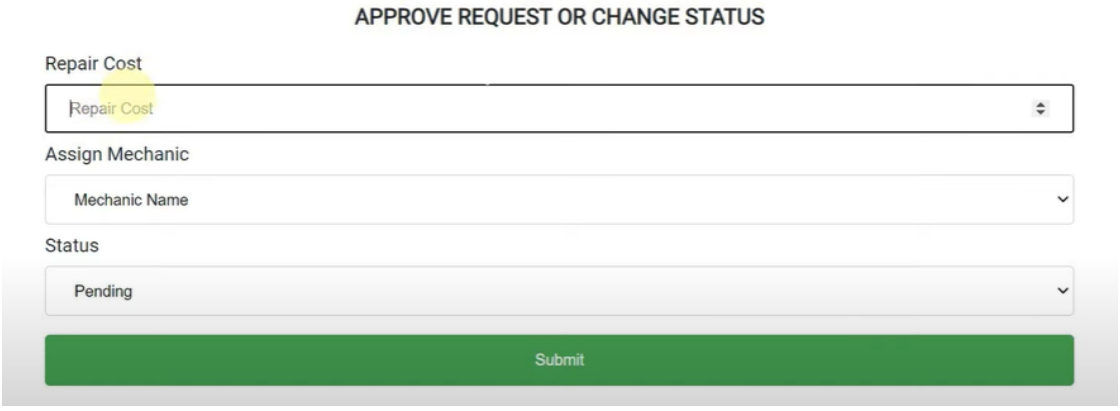


Рис. 4.15. Схвалення запиту на підвищення заробітної плати

Була встановлена задача створити інтуїтивно зрозумілий інтерфейс, тому наявність системи сповіщень є доволі переможним рішенням, оскільки бере на себе відповідальність про подальшу навігацію користувача системи.

Висновки до розділу 4

Основна мета цього розділу це проведення обґрунтування, вибір середовища розробки, базових програмних засобів та технологій розробки програмного забезпечення. Саме ці елементи були освітлені в даному розділі. Крім цього, в даному розділі була розроблена база даних, без якої існування системи не є можливим. Також було проведено ознайомлення користувачів з CRM-системою, а саме з його графічним інтерфейсом та функціоналом. Було визначено та описано керівництво користувача. Результатом виконання та опису даного розділу є розроблений веб-застосунок, що дозволяє власнику підприємства контролювати якість роботи працівників станції технічного обслуговувааня. Оскільки тільки якісна робота та дотримання визначених правил бізнесу зможе привести до покращення взаємодії з клієнтами та подальшлгого розвитку компанії.

ВИСНОВКИ

На підставі проведеного дослідження ми дізнались, що багато підприємців, особливо в сфері сервісів технічного обслуговування в наш час не особливо переймаються процесом оптимізації взаємодії з клієнтами. Це стосується таких речей, як економії часу клієнту та ще ряду причин. Проаналізувавши всі переваги та недоліки ведення справ таким чином, було прийнято рішення розробки системи CRM для контролю за роботою персоналу СТО за рахунок оптимізації процесів взаємодії з клієнтами. Реалізація такого продукту надасть можливість формування власного щоденника для кожного користувача системи користуючись веб-застосунок та ботом, який реалізовано у месенджері Telegram.

Перед початком розробки CRM-системи було виконано аналіз сфери застосування, що вказали на високий попит користувачів, менеджерів з продажу, на використання такого роду системи як основних інструментів за для роблти. Для визначення дійсної необхідності у розробці продукту було проведено пошук та аналіз аналогічних систем та виконано їх детальне вивчення з проведенням перевірок системи на спроможність системи впоратись із задачами, які перед нею поставлені. На основі детального вивчення було визначено ряд функцій, що повинний надавати веб-застосунок.

Як наслідок, на основі повністю спроектованої системи за допомогою обраного підходу та технологій було виконано програмну реалізацію CRM-системи. Для потенційних користувачів системи було складено детальну інструкцію.

Результатом реалізації програмного продукту став повністю функціонуючий застсоунок, яким можна користуватись.

Таким чином, всі поставлені задачі були виконані в повній мірі, мета досягнута, а продукти готові для впровадження.

Спеціальний розділ дипломної роботи містить певні розрахунки, рекомендації та положення щодо використання інформаційних пристроїв, які є актуальними і для розробників програмного забезпечення, і для користувачів. Ці рекомендації дозволяють уникати небезпек для здоров’я та життя, що є найважливішим аспектом життєдіяльності та благополуччя громадян

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Черкашин П.А. Стратегія управління взаємовідносинами з клієнтами (CRM) [Електронний ресурс]/Черкашин П.А. - М: Інтернет Університет Інформаційних Технологій (ІНТУІТ), 2016.- 420 с.
2. Крахоткіна Є.В. Методи та засоби проектування інформаційних систем та технологій: навчальний посібник/ Крахоткіна О.В. - Ставрополь: Північно-Кавказький федеральний університет, 2015. -152 с.І
3. Золотов С. Ю. Проектирование информационных систем : учеб. пособие / С. Ю. Золотов ; Томский гос. ун-т систем управления и радиоэлектроники. - Томск : Эль Контент, 2013. - 86 с.
4. Інформаційні технології та управління підприємством [Електронний ресурс] / В.В. Баронов [та ін]. - Саратов: Профосвіта, 2017. - 327 с.
5. Середовище розробки XAMPP [Електронний ресурс]. - Режим доступу :https://www.apachefriends.org/ru/index.html (дата звернення 17.05.2017)
6. Чистякова В.І. Проектування інформаційних систем. Підручник для студентів закладів вищої професійної освіти/В.І. Чистякова, В.В.Бєлов - М.: Академія, 2013. - 352 с
7. Компания «AutoSoft» [Электронный ресурс]. – Режим доступа : http://www.autosoft.ru (дата обращения 17.05.2017)
8. ГОСТ 34.320-96. Інформаційна технологія. Система стандартів з баз даних. Концепції та термінологія для концептуальної схеми та інформаційної бази.
9. Реінжиніринг бізнес-процесів: навч. посібник/під ред. А. О. Блінова. - М.: ЮНІТІ-ДАНА, 2015. - 343 с.
10. Бізлі Д. М. Мова програмування Python : довідник : пров. з англ. / Д. М. Бізлі. - Київ: ДіаСофт, 2000.
11. Гіфт Н. Python у системному адмініструванні UNIX та Linux: пров. з англ. / Н. Гіфт, Д. Джонс. - СПб. : СимволПлюс, 2009.

12. Лейнінгем І. Освой самостійно Python за 24 години: пров. з англ. / І. Лейнінгем. - М.: Видавничий дім "Вільямс", 2001
13. Ліса А. Python. Керівництво розробника: пров. з англ. / А. Ліса. - СПб. : ДіасофтЮП, 2001.
14. Лутц М. Вивчаємо Python: пров. з англ. / М. Лутц. - СПб. : Символ-Плюс, 2009
15. Лутц М. Програмування на Python: пер.с англ. / М. Лутц. - СПб. : Символ-Плюс, 2002.
16. Саммерфельд М. Програмування на Python 3. Детальний керівництво: пров. з англ. / М. Саммерфельд. - СПб. : Символ-Плюс, 2009
17. Гандзюк М. П., Желібо Є. П., Халімовський М. О. Основи Охорони праці: Підруч. для студ. вищих навч. закл. За ред. М. П. Гандзюка. – К.: Каравела; Львів: Новий Світ-2000, 2003. - 408 с.
18. Сузі Р. А. Python / Р. А. Сузі. - СПб. : БХВ-Петербург, 2002.
19. Сузі Р. А. Мова Python та його застосування: навч. допомога / Р.А. Сузі. - М.: Інтернет-Університет інформаційних технологій: БІНОМ. Лабораторія знань, 2006.
20. Мова програмування Python/Г. Россум [та ін]. - СПб. : АНО "Інститут логіки" - Невський діалект, 2001.
21. Трофімова М.В. Предметно-орієнтовані інформаційні системи: навчальний посібник/Трофімова М.В. - Ставрополь: Північно-Кавказький федеральний університет, 2014. – 188 с.
22. Декомпозиція – Вікіпедія [Електронний ресурс] : [Веб–сайт]. – Режим доступу: http://bit.ly/2L87qAC;
23. Джефф Форсьє, Пол Біссекс, Уеслі Дж. Чан. Django. Розробка веб-додатків на Python– Київ : Символ-Плюс, 2018. – 238 с.
24. Годін В.В. Інформаційне забезпечення менеджерської діяльності– Київ : Вища школа, 2011. – 240 с.

25. Маклаков С.В. Моделювання процесів – Київ : Діалог, 2012. – 223 с.
26. Титаренко Г.А., Автоматизовані інформаційні технології– Київ : Символ-Плюс, 2010. – 159 с.
27. Агафонов В.Н. Логічне програмування – Київ : Символ-Плюс, 1998. – 426 с.
28. Арсаж Ж. Програмування задач на Python– Київ : Пересвіт, 2014. – 521 с.
29. Кнут Є.М. Програмування– Київ, 1978.
30. Роберт Мартін. Чистий код. Створення, аналіз та ре факторинг – СПб : 2015. – 464 с.
31. Пьюрівал С. Основи розробки веб-застосунків– СПб 2016. – 512 с.
32. Дєйт К.Ж. Введення у бази даних– Москва, 2017. – 1328 с.
33. Рижко А.М., Рибніков А.М., Рижко Н.А. Інформаційні системи керування компанією– СПб : Юрайт, 2016. – 356 с.
34. Володимир Дронов: Django 2.1. Практика створення веб-сайтів на Python. – Київ : Освіта України, 2019. – 672 с.
35. Гаррі Персіваль. Розробка, орієнтована на тестування за допомогою Python: використання Django, Selenium і JavaScript, 2-е видання. – Київ : Освіта України, 2018. – 945 с.
36. Майкл Уайт. Опанування Python: машинне навчання, структури даних, Django, об’єктно-орієнтоване програмування та програмна інженерія – Київ : Книга, 2018. – 611 с.
37. Натрі Самулі. Django - простий спосіб: покроковий посібник зі створення веб-сайтів Django. – Київ : Освіта України, 2018. – 206 с.
38. Ендрю Пінкман. Django звільнений. – Київ : Книгосвіт, 2019. – 835 с.
39. НПАОП 0.00-1.28-10 про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин – редакція від 10.04.2010.
40. Про охорону праці: Закон України – редакція від 01.01.2004.

41. Готовський Ю. В. Електромагніта безпека в офісі та приміщенні / Ю. В. Готовський, Ю. Ф. Пєров. – Москва: Імедіус, 1998. – 176 с.;
42. Державні санітарні норми і правила захисту населення від впливу електромагнітних випромінювань. Затвердженно наказом МОЗ України від 01.08.96 року. № 239. – Київ, 1996, 28 с.;
43. Синеок. С. В. Захист від випромінювання мобільних телефонів. – Юпітер, 2002. – с. 36–39.
44. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень
45. Общие санитарно-гигиенические требования к воздуху рабочей зоны (с Изменением N 1) [Текст] : ГОСТ 12.1.005-88 - Вид. офіц. ; введ. 1989-01–01 (Система стандартов безопасности труда (ССБТ))
46. Гандзюк М. П., Желібо Є. П., Халімовський М. О. Основи Охорони праці: Підруч. для студ. вищих навч. закл. За ред. М. П. Гандзюка. – К.: Каравела; Львів: Новий Світ-2000, 2003. - 408 с.

ДОДАТОК А

Лістинг коду

Реалізація моделей

```
from django.db import models
from django.contrib.auth.models import User
# Create your models here.
class Customer(models.Model):
    user=models.OneToOneField(User,on_delete=models.CASCADE)
    profile_pic=
models.ImageField(upload_to='profile_pic/CustomerProfilePic/',null=True,blank
=True)
    address = models.CharField(max_length=40)
    mobile = models.CharField(max_length=20,null=False)
    @property
    def get_name(self):
        return self.user.first_name+" "+self.user.last_name
    @property
    def get_instance(self):
        return self
    def __str__(self):
        return self.user.first_name

class Mechanic(models.Model):
    user=models.OneToOneField(User,on_delete=models.CASCADE)
    profile_pic=
models.ImageField(upload_to='profile_pic/MechanicProfilePic/',null=True,blank
=True)
    address = models.CharField(max_length=40)
    mobile = models.CharField(max_length=20,null=False)
    skill = models.CharField(max_length=500,null=True)
    salary=models.PositiveIntegerField(null=True)
    status=models.BooleanField(default=False)
    @property
    def get_name(self):
        return self.user.first_name+" "+self.user.last_name
    @property
    def get_id(self):
        return self.user.id
    def __str__(self):
        return self.user.first_name


class Request(models.Model):
    cat=(('two wheeler with gear','two wheeler with gear'),('two wheeler
without gear','two wheeler without gear'),('three wheeler','three
wheeler'),('four wheeler','four wheeler'))
    category=models.CharField(max_length=50,choices=cat)

    vehicle_no=models.PositiveIntegerField(null=False)
    vehicle_name = models.CharField(max_length=40,null=False)
    vehicle_model = models.CharField(max_length=40,null=False)
    vehicle_brand = models.CharField(max_length=40,null=False)

    problem_description = models.CharField(max_length=500,null=False)
    date=models.DateField(auto_now=True)
```

```
    cost=models.PositiveIntegerField(null=True)

    customer=models.ForeignKey('Customer',
on_delete=models.CASCADE,null=True)
    mechanic=models.ForeignKey('Mechanic',on_delete=models.CASCADE,null=True)


stat=(('Pending','Pending'),('Approved','Approved'),('Repairing','Repairing')
,('Repairing Done','Repairing Done'),('Released','Released'))

status=models.CharField(max_length=50,choices=stat,default='Pending',null=Tru
e)

    def __str__(self):
        return self.problem_description

class Attendance(models.Model):
    mechanic=models.ForeignKey('Mechanic',on_delete=models.CASCADE,null=True)
    date=models.DateField()
    present_status = models.CharField(max_length=10)

class Feedback(models.Model):
    date=models.DateField(auto_now=True)
    by=models.CharField(max_length=40)
    message=models.CharField(max_length=500)
```

ДОДАТОК Б

Лістинг коду

Реалізація форм, що існують у системі

```
. from django import forms
from django.contrib.auth.models import User
from . import models

class CustomerUserForm(forms.ModelForm):
    class Meta:
        model=User
        fields=['first_name','last_name','username','password']
        widgets = {
        'password': forms.PasswordInput()
        }

class CustomerForm(forms.ModelForm):
    class Meta:
        model=models.Customer
        fields=['address','mobile','profile_pic']


class MechanicUserForm(forms.ModelForm):
    class Meta:
        model=User
        fields=['first_name','last_name','username','password']
        widgets = {
        'password': forms.PasswordInput()
        }

class MechanicForm(forms.ModelForm):
    class Meta:
        model=models.Mechanic
        fields=['address','mobile','profile_pic','skill']

class MechanicSalaryForm(forms.Form):
    salary=forms.IntegerField();


class RequestForm(forms.ModelForm):
    class Meta:
        model=models.Request

fields=['category','vehicle_no','vehicle_name','vehicle_model','vehicle_brand
','problem_description']
        widgets = {
        'problem_description':forms.Textarea(attrs={'rows': 3, 'cols': 30})
        }

class AdminRequestForm(forms.Form):
    #to_field_name value will be stored when form is submitted.....__str__
method of customer model will be shown there in html

customer=forms.ModelChoiceField(queryset=models.Customer.objects.all(),empty_
label="Customer Name",to_field_name='id')
```

```
mechanic=forms.ModelChoiceField(queryset=models.Mechanic.objects.all(),empty_
label="Mechanic Name",to_field_name='id')
    cost=forms.IntegerField()

class AdminApproveRequestForm(forms.Form):

mechanic=forms.ModelChoiceField(queryset=models.Mechanic.objects.all(),empty_
label="Mechanic Name",to_field_name='id')
    cost=forms.IntegerField()

stat=(('Pending','Pending'),('Approved','Approved'),('Released','Released'))
    status=forms.ChoiceField( choices=stat)


class UpdateCostForm(forms.Form):
    cost=forms.IntegerField()

class MechanicUpdateStatusForm(forms.Form):
    stat=(('Approved','Approved'),('Repairing','Repairing'),('Repairing
Done','Repairing Done'))
    status=forms.ChoiceField( choices=stat)

class FeedbackForm(forms.ModelForm):
    class Meta:
        model=models.Feedback
        fields=['by','message']
        widgets = {
        'message':forms.Textarea(attrs={'rows': 6, 'cols': 30})
        }

#for Attendance related form
presence_choices=(('Present','Present'),('Absent','Absent'))
class AttendanceForm(forms.Form):
    present_status=forms.ChoiceField( choices=presence_choices)
    date=forms.DateField()

class AskDateForm(forms.Form):
    date=forms.DateField()


#for contact us page
class ContactusForm(forms.Form):
    Name = forms.CharField(max_length=30)
    Email = forms.EmailField()
    Message =
forms.CharField(max_length=500,widget=forms.Textarea(attrs={'rows': 3,
'cols': 30}))
```