Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

Система управління клієнтами (CRM) авто-ділера

Текстова частина до курсової роботи
за спеціальністю „Інженерія програмного забезпечення ” 121

Керівник курсової роботи
кандидат фізико-математичних наук, старший викладач
Сініцина Р. Б.

_____

(Підпис)
“___” _____ 2021 року
Виконав студент ІПЗ-БПЗ
Романенко М. О.
“___” _____ 2021 року

Київ 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ

Зав. Кафедри мережних технологій,

професор, д.ф-м.н.

_____ Г.І. Малашонок

*(підпис)*_ _____

_“____” _____ 2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

Студенту Романенко Михайлу Олександровичу факультету інформатики 3 курсу

ТЕМА: Система управління клієнтами (CRM) авто-ділера

Зміст ТЧ до курсової роботи:

Календарний план

Перелік умовних позначень

Аннотоція

Вступ

Аналіз предметної області та постановка завдання

Теоретичні відомості (про задачу, методи і підходи до її розв’язку тощо)

Опис реалізації програми

Висновки

Список використаної літератури

Дата видачі “____” _____ 2021 р.

Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання роботи

Тема: Р Система управління клієнтами (CRM) авто-ділера

№	Назва етапу	Термін виконання	Примітка
1.	Отримання теми курсової	20.11.2020	
2.	Ознайомлення з предметною областю	Листопад 2020	
3.	Пошук та ознайомлення з корисною літературою	Грудень 2020 - січень 2021	
4.	Планування структури та архітектури практичної частини роботи	Січень 2021 – лютий 2021	
5.	Програмування застосунку	Лютий-квітень 2021	
6.	Написання текстової частини роботи	Квітень-травень 2021	
7.	Створення презентації та доповіді до неї	16.05.2021	

8.	Здача курсової роботи на перевірку	17.05.2021	

Студент Романенко М. О.

Керівник Синіцина Р. Б. “____” _____

ЗМІСТ

Анотація

Протягом виконання даної роботи було проаналізовано підходи для створення і впровадження CRM-систем для різних типів бізнесу порівняно їх переваги, недоліки і сценарії застосування, розроблено власну CRM-систему на базі платформи Terrasoft Studio Creatio для автодилерського центру. Реалізовані можливості: планування тест-драйвів, облік звернень і відслідковування виконання роботи по ним в автосервісі, додавання продажів, відслідковування їх ходу, робота з лідами (потенційними клієнтами), процес конверсії ліда в клієнти.

Вступ

Актуальність обраної теми

В умовах стрімкого розвитку ринку дедалі нагальнішою стає потреба в автоматизації якомога більшої кількості процесів бізнесу. Всі засоби покращення бізнесу, як правило, впроваджуються масово внаслідок ринкового конкурування, тому в перспективі майже кожна компанія зробить свій внесок у створення колосального попиту на ці засоби. Перспективним напрямком у сфері автоматизації бізнесу є управління відносинами з клієнтами (CRM).

Царина CRM охоплює такі аспекти ведення бізнесу, як: реєстрація і облік клієнтських звернень, управління відносинами з лідами (потенційними клієнтами), управління продажами та будь-якими іншими процесами, у яких задіяні клієнти, процес проведення . Сьогодні існує велика кількість інструментів для впровадження CRM-системи: розробка власного рішення, застосування готового продукту, інтеграція рішень на базі спеціалізованих платформ для створення інструментів бізнесу. Впровадження рішень для автоматизації є необхідним для всіх типів бізнесу, адже воно надає можливість оптимізувати ведення бізнесу майже у кожному його аспекті:

- збільшити кількість продажів;
- централізувати розподіл задач серед співробітників;
- мінімізувати вірогідність людської помилки;
- автоматизувати процеси, для виконання яких в компанії задіяні люди, таким чином зменшивши у перспективі вартість роботи компанії;
- пришвидшити роботу компанії в цілому;
- регламентувати доступ співробітників до важливих даних за їх ролями;
- надати співробітникам зручний інструментарій для виконання їх обов’язків, планування задач та інше.

Мета та завдання роботи

Метою даної роботи є:

1. Створення CRM-системи для використання в автодилерському центрі
2. Огляд і порівняння підходів для розробки такої системи (реалізація з нуля, реалізація на основі платформи для автоматизації бізнес-процесів), їх сценаріїв використання.

Об’єкт дослідження

Об’єкт дослідження у даній роботі – компанія-автодилер, її бізнес-процеси та засоби їх оптимізації за допомогою CRM-системи.

Предмет дослідження

Предмет дослідження у даній роботі є CRM-системи, а також підходи до їх впровадження на реальному бізнес ринку.

Використані технології

- Terrasoft Studio Creatio –платформа для реалізації CRM-застосунку;
- BPMN.io – онлайн-застосунок для створення діаграм бізнес-процесів у нотації BPMN 2.0 (Business Process Modelling Notation – формальний запис для моделювання бізнес-процесів);

РОЗДІЛ 1 Аналіз предметної області та постановка завдання

1.1 Постановка задачі

Користувачі системи

Клієнтом системи є компанія-автосалон, яка надає наступні послуги:

- проведення тест-драйвів;
- продаж автомобілів;
- автосервіс для клієнтів.

У системі є 3 типи користувачів:

- Менеджер з продажів

  Основне робоче місце у компанії. В обов’язки менеджера з продажів входять комунікації з клієнтами та потенційними клієнтами, планування і проведення тест-драйвів, продаж автомобілів покупцям. Менеджер з продажів має доступ до наступної інформації: моделі авто, з якими працює автосалон, автомобілі в наявності, ліди, заявки на тест-драйв, розклад тест-драйвів, продажі та пов’язані з ними задачі – дзвінки, підготовка автомобілів до продажу і т. д..

- Адміністратор

  Адміністратор має доступ до всієї інформації, до якої має доступ менеджер з продажів. Окрім цього, адміністратор має можливість переглянути статистику за всіма послугами, що надає компанія, додавати моделі автомобілів.

- Працівник сервісу

  Задача працівника сервісу – реєстрація звернень клієнтів у сервіс, надання сервісних послуг клієнтам. Працівник сервісу має доступ лише до сервісної частини застосунку.

1.2 Опис бізнес-процесів компанії за допомогою нотації BPMN 2.0

Шлях від потенційного клієнта до клієнта («конверсія ліда»)

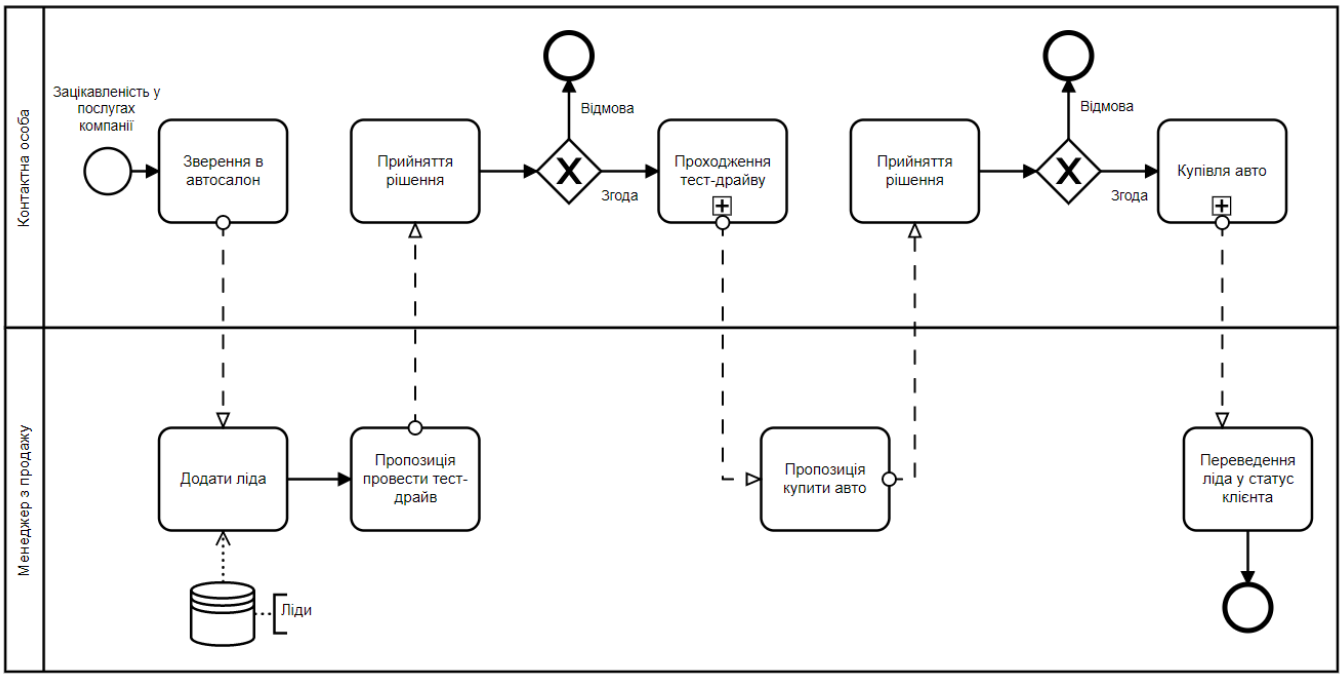

Рисунок 1.2.1 Діаграма процесу «конверсія ліда»

Lead, лід – потенційний клієнт, потребу якого може втілити компанія. Конверсія ліда – процес переведення потенційного клієнта у категорію «клієнти» шляхом надання йому послуг з продажу авто. Статус виконання цього процесу має відслідковуватися з пропонуванням подальших дій (див. рис. 1.2.1).

Планування і проведення тест-драйву

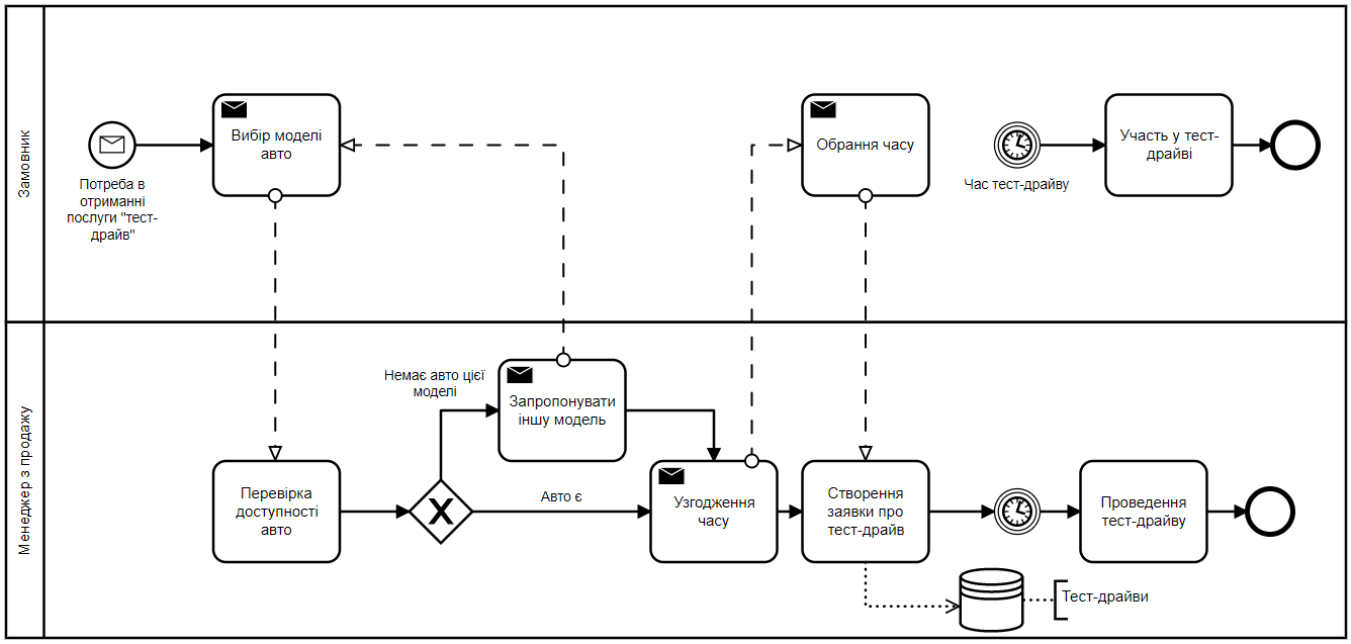

Рисунок 2.2.3. Діаграма бізнес-процесу «планування і проведення тест-драйву».

Для проведення тест-драйву замовнику пропонується одне з виставкових авто, що є у салоні. Виставкові авто не мають бути доступними для продажу, у той час як авто для продажу не мають використовуватися для тест-драйвів, тому що новий автомобіль може бути пошкоджений під час тест-драйву, що зробить його непридатним для продажу. Після проходження тест-драйву потенційний клієнт отримує від менеджера пропозицію купити авто (див. рис. 1.2.2).

Продаж авто

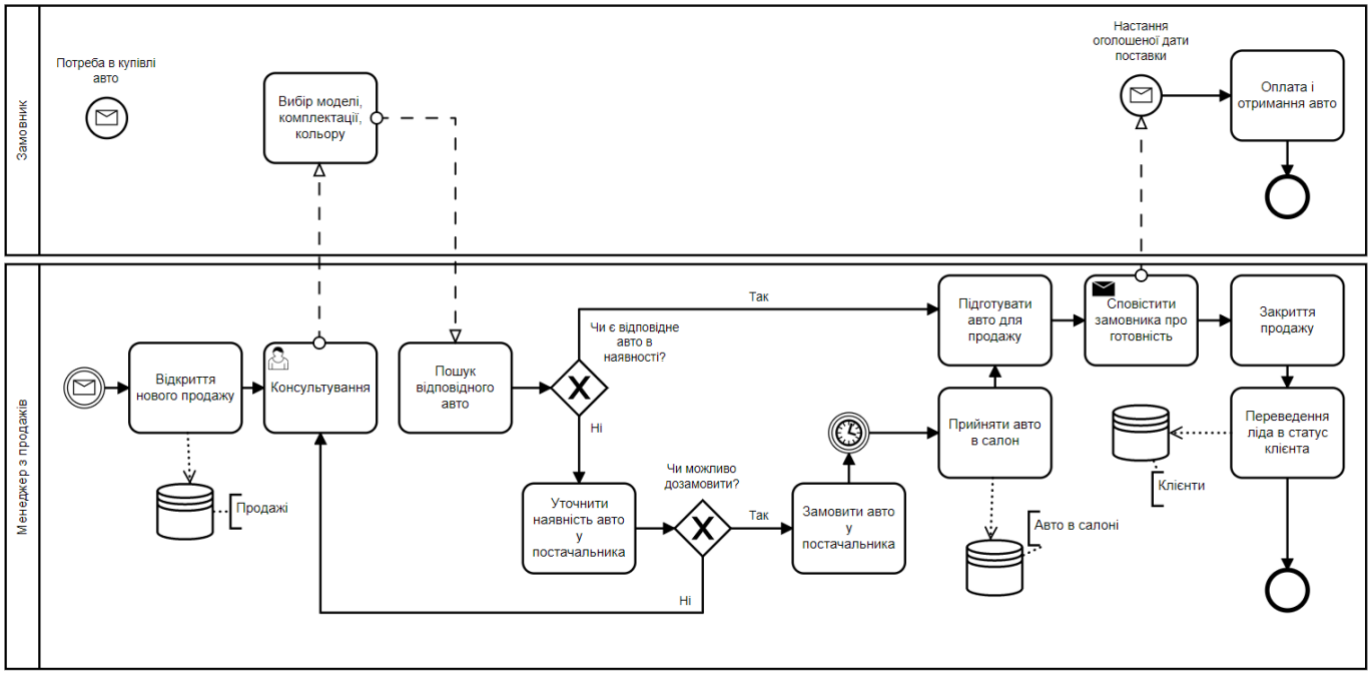

Рисунок 1.2.4. Діаграма бізнес-процесу "продаж авто"

Під час продажу авто менеджерові необхідно отримати інформацію від клієнта про бажану модель, комплектацію і колір автомобіля. Якщо у салоні відсутнє авто, що відповідає цим вимогам, то менеджер з продажу може дозамовити бажане авто і продовжити процес продажу. Якщо можливості дозамовити даний автомобіль немає, то менеджер пропонує замовнику обрати інші параметри автомобіля. Якщо такий автомобіль наявний у салоні або є можливість його дозамовити, то менеджер приймає автомобіль, перевіряє його готовність до продажу і сповіщує клієнта. Після цього клієнт оплачує і отримує авто (див. рис. 1.2.3).

Робота сервісу

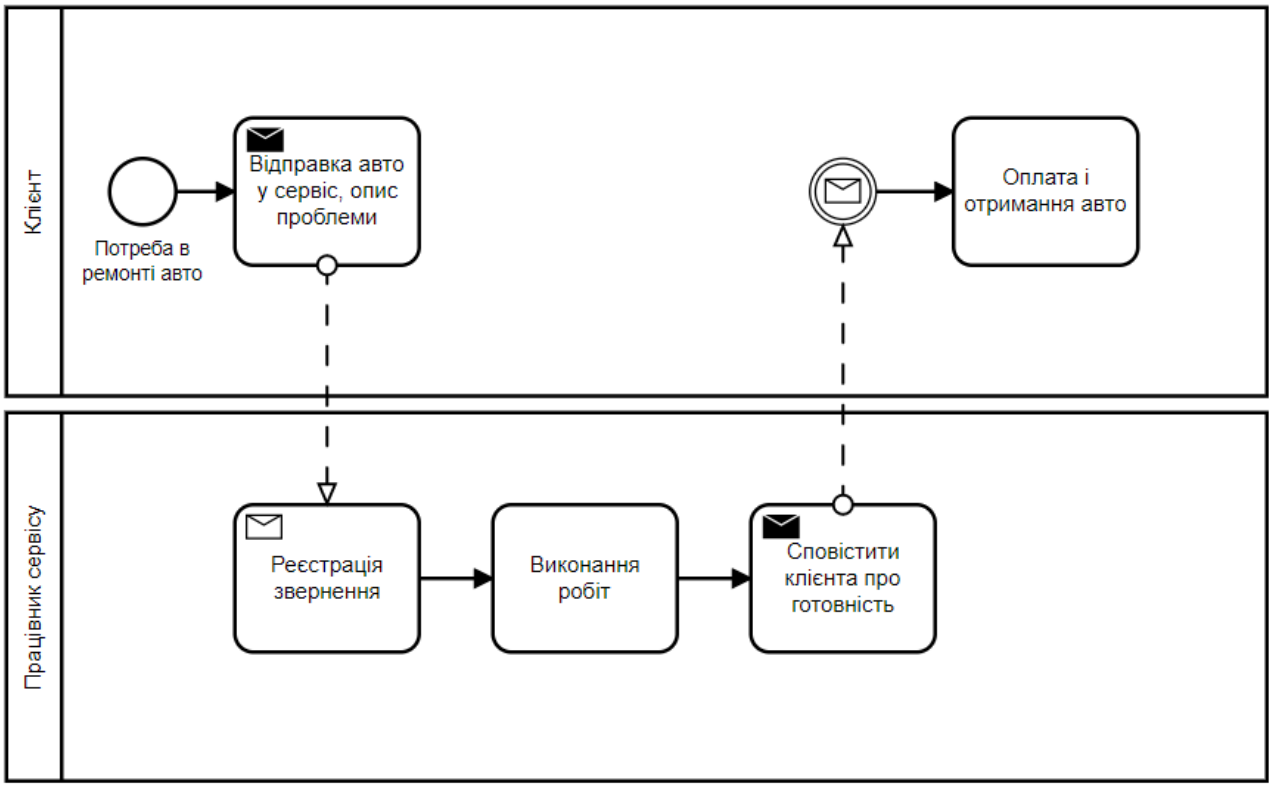

Рисунок 1.2.5. Діаграма бізнес-процесу "сервісне обслуговування"

Робота сервісу виконується за стандартною схемою. Клієнт звертається, надає свій автомобіль та інформацію про проблему. Працівник виконує роботи і сповіщує клієнта про готовність (див. рис. 1.2.4).

1.3 Вимоги до розроблюваної системи

Розроблена система повинна реалізовувати наступний функціонал:

- розподіл прав доступу на розділи програми за роллю співробітника;
- можливість додавати і редагувати інформацію про автомобілі, які продає компанія;
- можливість запланувати тест-драйв обраного потенційним клієнтом автомобіля;
- можливість додавати лідів та відслідковувати їх статус та пов’язані з ними задачі;
- можливість додавати звернення у сервіс і відслідковувати хід виконання робіт по ним;
- можливість відслідковувати процес продажу від додання заявки то видачі авто клієнту;

- втілення процесу «конверсія ліда» без прямої ініціації наступних кроків від користувача системи.

РОЗДІЛ 2 Теоретичні відомості (про задачу, методи і підходи до її розв'язку тощо)

2.1 Аналіз існуючих підходів до вирішення проблеми

Загалом, підходи до розробки CRM-системи можна розділити на дві умовні категорії: розробка рішення з нуля і розробка рішення на основі існуючої платформи. Далі буде розглянуто варіанти втілення цих підходів, порівняно їх переваги та недоліки, сценарії застосування.

2.2 Розробка рішення з нуля

До появи на ринку BPM-платформ (BPM – Business Process Management – концепція управління, яка в основі діяльності підприємства ставить поняття бізнес-процесу, має на меті максимальну формалізацію і прозорість його представлення) розробка власного рішення з нуля була єдиним можливим варіантом впровадження інструментів автоматизації бізнесу, у тому числі і CRM-систем. Цей підхід сьогодні хоч і поступається у популярності розробці на базі BPM-платформ, та все ж має набір переваг, які роблять його пріоритетним рішенням для певного набору сценаріїв (сценарії використання будуть описані нижче).

Переваги розробки з нуля:

- повна свобода вибору технологій;
- гнучкість, можливість реалізувати будь-який функціонал у будь-якій формі;
- свобода у розробці користувацького інтерфейсу в противагу шаблонному вигляду інтерфейсів, реалізованих на готових платформах;
- прозорість роботи системи, можливість дороблять не тільки бізнес-логіку і рівень даних, а і будь-який елемент системи, оптимізувати код рішення на будь-якому рівні;
- не-покладання на сторонні компанії в питаннях безпеки. Цей пункт, залежно від розміру компанії, може бути або перевагою, або недоліком: для великих компаній перекладення відповідальності за безпеки на третіх осіб може бути небезпечним, так як може існувати потреба;

- швидкість роботи – рішення, створене спеціально під виконання певної задачі, завжди буде швидше, ніж універсальне рішення, конфігуроване під виконання цієї задачі. Створене з нуля рішення не матиме всіх тих складних абстракцій, які необхідні готовій платформі для забезпечення універсальності.

Недоліки розробки з нуля:

- набагато більший об’єм задач – побудова архітектури, API, покриття тестами, забезпечення безпеки даних, реалізація логіки з нуля для виконання шаблонних бізнес-задач;
- у роботі задіяно велику кількість спеціалістів – команди розробників, аналітиків, дизайнерів, тестувальників;
- високий поріг компетенції розробників для створення рішень, що відповідають enterprise-стандартам;
- проблематичність у внесенні змін у майбутньому – для адаптації застосунку під нові вимоги необхідно буде знову задіяти розробників, провести тестування.

і, як наслідок з вищеописаних пунктів:

- висока ціна – розробка складного рішення з нуля може коштувати до сотень тисяч доларів;
- тривалий час розробки – за оцінкою компанії-розробника «Avada-media», час створення CRM-системи просунутого рівня командою професіоналів становить близько 1000 годин роботи.

Час та гроші – основні ресурси будь-якого бізнесу. Тож великі витрати цих ресурсів мають бути виправданими. Обґрунтованим вибір цього підходу роблять такі фактори, як:

- наявність складної нішевої бізнес-логіки, нешаблонних задач;
- високе навантаження на систему, що робить необхідною максимальну оптимізацію її роботи;
- необхідність повного контролю над кожним аспектом роботи програми;

- потреба в специфічному, складному графічному інтерфейсі.

Враховуючи ціну і час розробки, і сценарії застосування, можна зробити висновок, що розробка системи з нуля є найбільш оптимальною опцією для великого бізнесу або за відсутністю альтернатив для реалізації особливої логіки.

2.3 Архітектура та обрання технологій для розробки з нуля

Класична трирівненва архітектура «клієнт-сервер-база даних». Клієнтський рівень містить логіку користувацького інтерфейсу і слугує для представлення даних, які зберігаються на рівні бази даних і обробляються рівнем сервером. Нижче наведено можливий набір технологій для кожного з рівнів цієї архітектури.

Клієнт

Найкращим форматом для CRM-системи буде веб-додаток, адже він надає наступні переваги порівняно зі звичайним застосунком:

- кросплатформеність – незалежно від ОС клієнтських пристроїв, застосунок буде доступний будь-де, де є браузер, а реалізація мобільної версії клієнта від десктопної відрізнятиметься мінімально: зовнішнім виглядом інтерфейсу;
- легкість в адмініструванні – веб-клієнт не потребує самостійних оновлень, окремого процесу встановлення.

Типовим вибором для реалізації клієнта є такі фронтенд-фреймворки, як Angular.js, React.js, Vue.js

Сервер

Існує багато технологій для реалізації сервера. Ось деякі з них:

- Node.js

  Станом на 2021 р. Node.js розвивається стрімкими темпами і стає дедалі більш актуальним в ентерпрайз-проектах. Було проведено багато оновлень, які значно пришвидшили його роботу, розширивши спектр можливих застосувань.

Існує кілька десятків фреймворків для розробки серверів на Node.js, найпопулярнішим з яких можна вважати Express.js – обгортку над стандартним сервером, яка значно пришвидшує процес розробки. Приклад розвитку Node.js у сфері ентерпрайз – фреймворк Nest.js – абстракція над Express, що надає готову архітектуру з високим використанням інверсії управління (inversion of control) для створення слабко зв'язаних компонентів, а також підтримки TypeScript і великої кількості засобів декларативного стилю для виконання шаблонних задач, таких як валідація вхідних даних з HTTP-запитів, дозволяє створювати добре структуровані, масштабовувані та відмовостійкі додатки, що відповідають потребам бізнесу.

- Python

Наступним за популярністю в ентерпрайз проектах являється Python фреймворк Django, адже згідно з щорічною статистикою відомого серед розробників порталу StackOverflow, Django є десятим за популярністю веб-фреймворком [1].

Основна мета Django – полегшити розробникам створення складних веб-сайтів, які керуються складними БД [2]. Основою концепції Django є те, що даний фреймворк реалізує весь необхідний користувачу функціонал без підключення додаткових бібліотек. Як приклад - масштабний механізм шаблонізованих класів, автентифікація, URL роутінг, власний ORM функціонал включені в структуру Django, без підключення додаткових бібліотек [3]. Користувачами цього веб-фреймворку є такі корпорації, як Instagram, Mozilla, National Geographic, Pinterest, та інші.

- Java

Java є класичним вибором для побудови додатків подібного типу. Розроблений на основі Java фреймворк з відкритим кодом Spring Framework надає більшість необхідного функціоналу для розробки клієнт-серверних додатків: засоби для побудови сервера за патерном MVC (Model, View, Controller – рівень model відповідає за моделювання сутностей і зв'язок з

базою даних, controller містить бізнес-логіку, а логічний рівень view відповідає за представлення даних і логіку користувацького інтерфейсу.), власну ORM, модуль Security для реалізації захисних механізмів, таких як автентифікація і авторизація, та інші інструменти для розробки застосунків enterprise-рівня.

База даних

Реляційна модель надає найширший арсенал засобів для моделювання складних взаємозв'язків між сутностями і забезпечення підтримки цілісності даних засобами, такими як транзакції та обмеження цілісності, тож системи керування базами даних, такі як MySQL, PostgreSQL та інші подібного типу є найкращим вибором для використання у CRM-системі.

2.4 Розробка на базі існуючої платформи

Такий підхід об'єднує у собі два методи: **no-code** і **low-code** [6].

- No-code – підхід до розробки, який полягає у використанні готового інструментарію – графічних інтерфейсів, що дозволяють моделювати, сутності та процеси, задіяні в компанії.
- Low-code – підхід до розробки, який полягає у використанні готового інструментарію, аналогічно no-code, але з можливістю контролювати процес розробником на рівні коду і навіть змінювати код вже реалізованих програмних блоків для адаптації його під потреби конкретного проекту.

З 2017 р. ринок програмного забезпечення переживає бум популярності low-code та no-code технологій. Ці підходи не є новими, та масовості вони набули саме у цей час завдяки розвитку інструментів для побудови складних інтерфейсів у браузерному середовищі.

Підхід no-code поступається у популярності low-code з очевидних причин: неможливо вмістити в графічний інтерфейс всі ситуації, які може покрити проста інтеграція коду. Таким чином, no-code підхід у чистому вигляді є практично неможливим і здатен покрити лише малий відсоток задач. Тим не менш, no-code підхід є максимально ефективним для таких задач, як: створення сторінок

інтерфейсу програми, таблиць\сутностей, поверхневі налаштування програми, побудова простої утилітарної логіки.

Low-code підхід покладається переважно на ті самі інструменти, що і no-code, але надає можливість напряму користуватись кодом для вирішення будь-якої задачі. Приклади застосування low-code включають в себе SQL-сценарії для складних запитів до бази даних, обрахування значень формул, нетривіальну логіку валідації.

Переваги застосування існуючої платформи:

- можливість розробки командами з аналітика або аналітика і розробника;
- наявність готових рішень для великої кількості шаблонних бізнес - задач;
- наявність програмного фундаменту, який дозволяє сконцентрувати розробку на бізнес-логіці;
- неодноразово протестований базовий функціонал;
- постійна підтримка компанією-провайдером (найпоширеніша модель розповсюдження таких платформ – SaaS (Software as a service), яка передбачає постійну залученість компанії-провайдера, адже плата за послуги надається періодично);
- легка інтеграція сторонніх сервісів (веб-API, SMTP, телефонія і т. д.);
- легкість у внесенні змін до розробленої логіки;
- можливість розгортання системи не тільки на власних серверах, а і на серверах компанії-провайдера, що значно полегшує процесс підтримки;
- радикально нижча ціна розробки порівняно з розробкою рішення з нуля;
- менший час розробки.

Недоліки застосування існуючої платформи:

- шаблонний вигляд застосунку;
- непрозорість принципів роботи системи;
- менша швидкість роботи через велику кількість надбудов для кастомізації;
- відносна обмеженість функціоналу.

Оскільки, як вже було зазначено, основним критерієм для обрання бізнесом системи є час та вартість її створення та інтеграції, за відсутністю особливих сценаріїв, описаних у частині «Розробка рішення з нуля», пріоритетним підходом буде розробка рішення на базі існуючої платформи.

2.5 Огляд платформи Creatio

Платформа Terrasoft Creatio побудована за класичною трирівневою архітектурою:

- рівень представлення
- рівень застосунку
- рівень даних

Зараз розглянемо більш детально кожен з цих рівнів.

Рівень представлення

Цей рівень відповідає за відображення і логіку користувацького інтерфейсу. Основні технології – HTML, CSS, Javascript. Елементи користувацького інтерфейсу представлені схемами, реалізованими за допомогою Javascript. Схема кожного розділу інтерфейсу наслідується від базової схеми, успадковуючи від неї вбудовану логіку інтерфейсу.

Розробка на даному рівні виконується двома способами: за допомогою інтерфейсу користувача і шляхом додання нової логіки в схему засобами клієнтського Javascript - додаючи нові функції і властивості та перевизначаючи функції батьківської схеми.

Рівень застосунку

Серверна частина реалізована за допомогою мови програмування C# і .Net Framework, деякий вбудований функціонал, такий як моделі машинного навчання, реалізований на Python. Бізнес-логіка застосунку відповідає цьому рівню.

Розробка на даному рівні виконується за допомогою дизайнеру бізнес-процесів, який надає можливості будувати діаграми послідовностей дій системи та її користувачів. Приклади системних дій – читання, редагування і додавання даних, обрахування значень за формулами, виконання сценаріїв, реалізованих на

C# для складної логіки. Приклади дій користувача – відповідь на питання, планування і виконання завдань.

Рівень даних

Terrasoft Creatio надає можливості для роботи з такими СКБД, як Microsoft SQL Server, PostgreSQL, Oracle. Стандартним варіантом для продукту Studio Creatio є Microsoft SQL Server.

2.6 Обрання підходу для реалізації власної системи

Після аналізу предметної області було вирішено, що розробка рішення на базі готової платформи є найбільш відповідним вимогам предметної області та сучасного ринку підходом.

Враховано наступні аспекти:

- кількість користувачів системи – до 20. Система не отримуватиме високого навантаження, тож втрати швидкодії через велику кількість надбудов не є відчутними;
- специфіка предметної області – в основі діяльності автодилера лежить процес продажу. Цей процес є одним із найпоширеніших бізнес-процесів, його хід має багато спільних рис між різними предметними областями. Предметна область не має логіки, реалізацію якої не може забезпечити платформа Terrasoft Studio Creatio;
- можливість легко вносити зміни в систему – в умовах швидкої автоматизації у всіх сферах життя в Україні, система має бути легко та дешево змінюваною. Приклад - 16 лютого 2021 р. Кабінетом міністрів України ухвалено законопроєкт, який дозволятиме автосалонам реєструвати авто на місці, без відвідування клієнтом сервісного центру МВД самостійно. З квітня цього ж року цей процес почав активно впроваджуватися автосалонами. Для цього працівник салону має надіслати заявку з потрібною інформацією в сервісний центр МВД і отримати відповідь. Право на надсилання таких заявок компанія має за умови отримання

цифрового підпису. Використання готової платформи дає можливість мінімізувати складність впровадження цього процесу.

РОЗДІЛ 3 Опис реалізації програми

3.1 Розроблені розділи програми

1. Розділ «Ліди»

Розділ «Ліди» зберігає в собі інформацію про потенційних клієнтів, статус проходження ними процесу «конверсія ліда» - стадії проходження тест-драйву, купівлі авто, переведення у клієнти. Редагувати розділ мають право менеджери з продажу і адміністратори.

2. Розділ «Авто»

Розділ «Авто» зберігає в собі інформацію про моделі автомобілів, які продає автосалон і постачальників цих моделей. Редагувати має право лише адміністратор. Переглядати записи мають право всі користувачі.

3. Розділ «Авто в салоні»

Розділ «Авто в салоні» зберігає інформацію про автомобілі, наявні в автосалоні: модель авто, колір, рік випуску, ціна закупки, ціна продажу. Ціна продажу автоматично розраховується як 130% від ціни закупки. Відсоток націнки є системною змінною і може бути змінений в системних налаштуваннях. Також у розділі зберігається інформація про те, чи є автомобіль екземпляром для демонстрації\тест-драйву або автомобілем для продажу. Авто для демонстрації заборонено використовувати для продажу, авто для продажу заборонено використовувати для проведення тест-драйвів. Переглядати розділ мають право всі користувачі. Редагувати розділ мають право адміністратор і менеджер з продажу.

4. Розділ «Тест-драйви»

Розділ зберігає заявки на тест-драйв. При доданні запису у розділ автоматично створюється задача у розкладі. Містить такі поля, як контакт замовника, авто з салону для проведення. Дата і час проведення зберігаються у картці задачі, що

створюється при додаванні нового тест-драйву. Додавати, редагувати і переглядати записи мають право лише адміністратор і менеджер з продажу.

5. Розділ «Продажі»

Цей розділ містить інформацію про продажі і містить наступні поля: контакт замовника, бажані модель, колір і комплектація, сума до сплати, обране авто з салону. Модель авто, колір і комплектація при заповненні накладають на поле «обране авто» фільтр, який відсіює записи, значення яких у відповідних колонках не співпадають зі значеннями, вказаними у продажі. Якщо продаж додано, а автомобіль з салону не обрано, то користувачеві пропонується дозамовити автомобіль.

6. Розділ «Сервіс»

Цей розділ містить інформацію про звернення клієнтів до автосервісу. Звернення містить такі поля, як: заголовок(коротка назва задачі), модель авто, яке ремонтується, детальний опис проблеми, вартість виконання робіт. Після додання запису у розділ працівникові сервісу пропонується додати у розклад задачу «провести роботи», а після виконання цього завдання буде запропоновано провести дзвінок клієнту для повідомлення про виконання робіт.

3.2Реалізація бізнес-процесів компанії

Бізнес-процеси компанії реалізовані у програмі за допомогою діаграм, створених у конструкторі бізнес-процесів. Окрім дій співробітників, як на діаграмах BPMN 2.0, у процесах із Studio Creatio присутні системні дії, які полегшують робочий процес шляхом розрахунків, підстановок значень у текстові поля і т. д..

Більшість процесів запускаються за допомогою сигналів додавання/редагування записів, що дозволяє автоматично ініціювати наступні дії.

- Обробка додавання нового запису в розділ «Ліди»

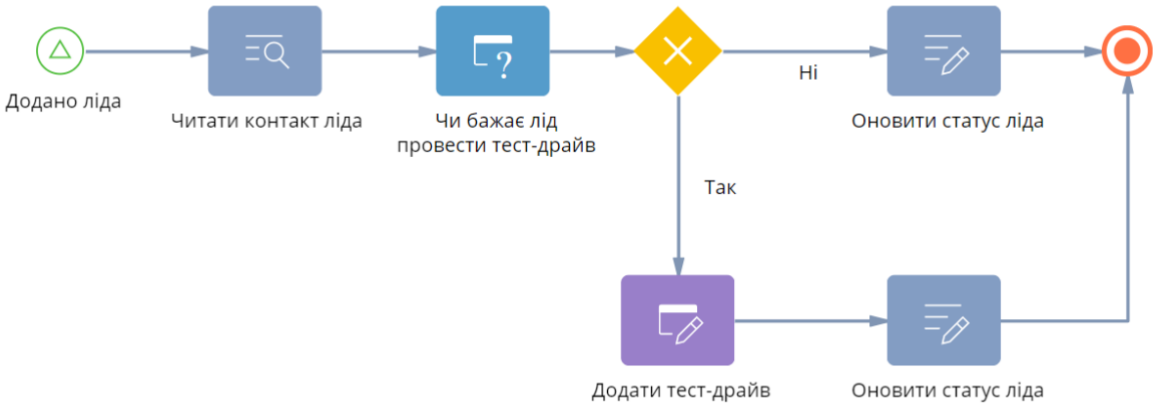

Рисунок 3.3.1. Дії системи після додавання нового ліда

Процес запускається після додавання в систему нового ліда. Система читає інформацію про контакт, по якому створено лід, для використання в наступних блоках. Після цього користувач має відповісти на питання системи, про згоду потенційного клієнта на проведення тест-драйву. При відповіді «так» користувачеві пропонується додати у систему запис про новий тест-драйв, а статус ліда встановлюється як «тест-драйв», що означає, що лід перебуває на цьому етапі процесу «конверсія ліда». У разі відповіді «ні» встановлюється статус «потребу скасовано» (див. рис. 3.3.1).

- Обробка додавання нового тест-драйву

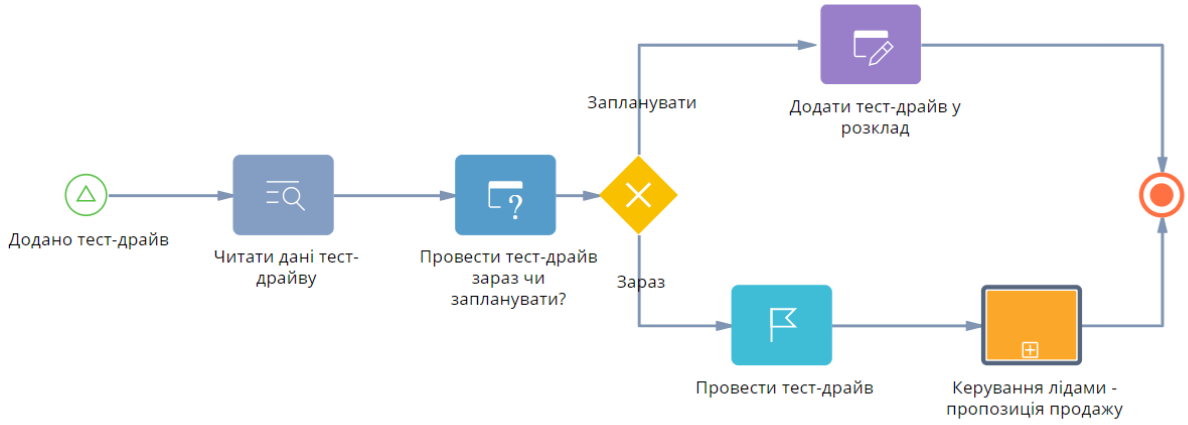

Рисунок 3.3.2 Дії системи після додавання нового тест-драйву

Тест-драйв можна провести двома способами: запланувавши і провівши одразу. Ці варіанти відрізняються тим, який час буде за замовчуванням встановлено для задачі і чи буде вона відображуватись у розкладі. Після проведення тест-драйву запускається наступна стадія процесу – пропозиція продажу. У випадку сьогочасного проведення тест-драйву процес пропозиції продажу запускається з поточного процесу вручну, у випадку запланованого – після встановлення користувачем статусу «виконана» для задачі в розкладі. (див. рис. 3.3.2).

- Обробка додавання нового продажу

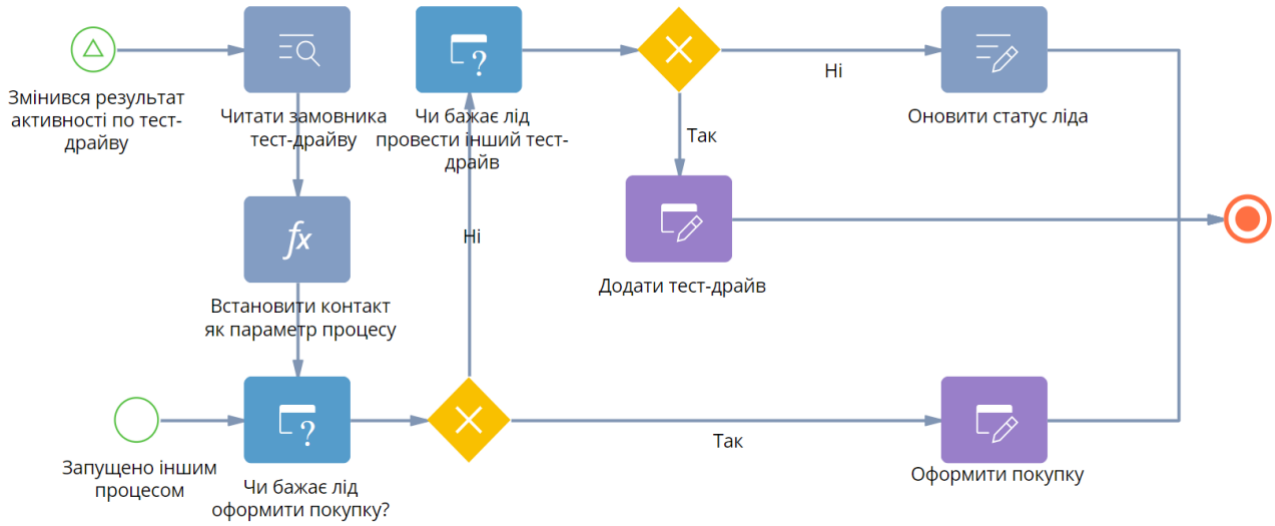

Рисунок 3.3.3 Дії системи після проведення тест-драйву.

Цей процес реалізує можливість повторно запланувати тест-драйв у випадку, якщо потенційний клієнт не задоволений протестованим автомобілем, або перейти до процесу купівлі авто. Він запускається або вручну, або у випадку, коли було встановлено статус «виконано» для завдання «провести тест-драйв» у розкладі. У разі відмови потенційного клієнта від покупки авто і проведення повторного тест-драйву встановлюється статус «потребу скасовано». У випадку згоди ліда на покупку система відкриває сторінку продажу, підставляючи відомі в даний момент дані у необхідні поля (див. рис. 3.3.3).

- Продаж авто

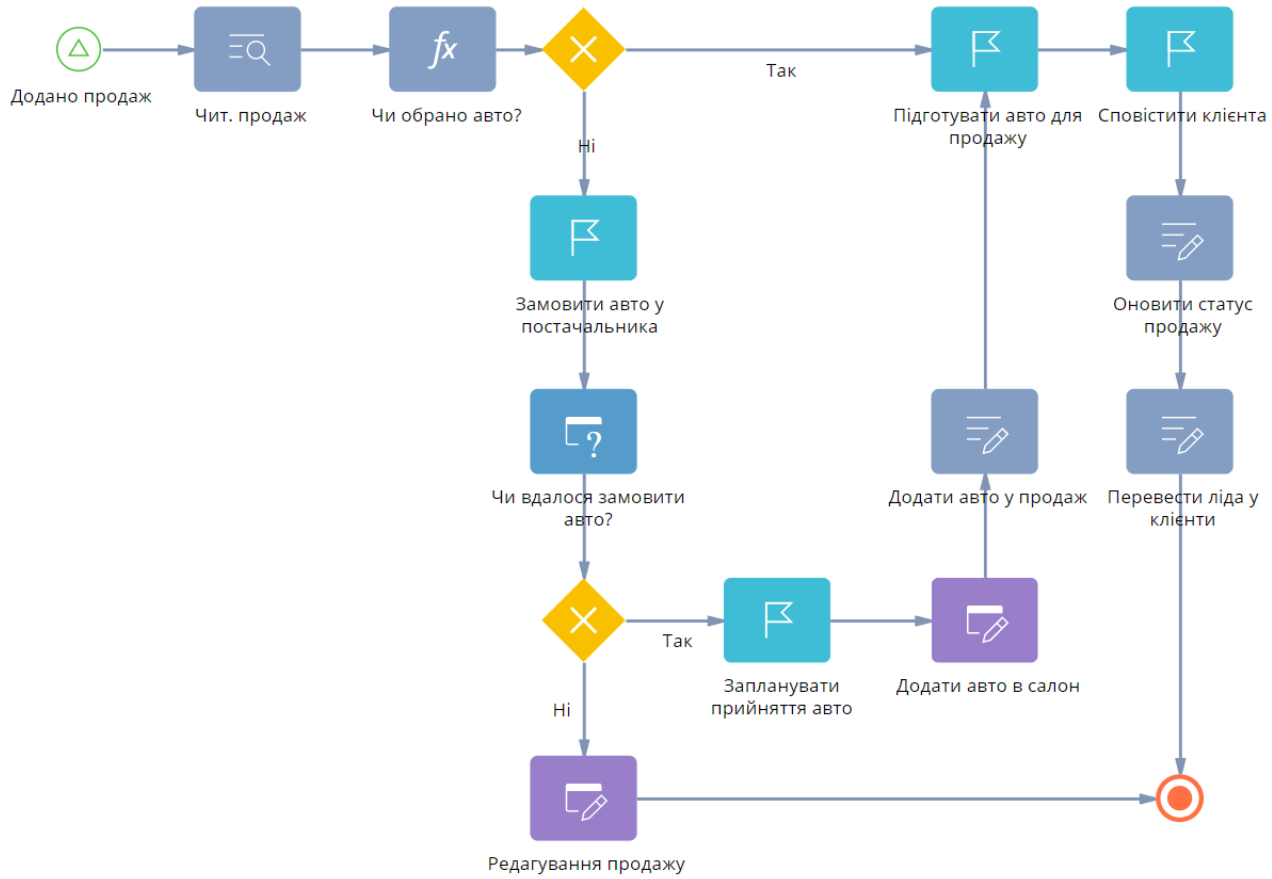

Рисунок 3.3.4 Процес продажу авто.

При заповненні продажу користувач вводить дані про бажану клієнтом модель авто, колір і комплектацію. Кожне з цих полів накладає фільтр на вибірку поля «обране авто», тобто після введення інформації в усі 3 поля у полі «обране авто» можна буде обрати лише ті автомобілі з салону, інформація про модель, комплектацію і колір яких повністю відповідає заданій у продажу.

Якщо користувач залишить поле пустим, система запропонує йому дозамовити автомобіль, зателефонувавши постачальнику авто, інформація про якого зберігається в обраному записі моделі автомобіля. У разі неможливості дозамовити автомобіль користувачеві пропонується повернутися до редагування продажу. У разі успіху користувач має запланувати прийом авто в салон, додати запис про нього у розділ «авто в салоні», підготувати необхідні для продажу документи і зателефонувати клієнту, сповістивши про готовність автомобіля. Після цього статус продажу встановлюється як «завершена», а статус ліда як «переведено у клієнти» (див. рис. 3.3.4).

3.3 Low-code розробка

Приклад застосування коду для вирішення задач, для яких не передбачений no-code функціонал – нетривіальна валідація. В даному випадку – перевірка, чи вже існує тест-драйв на запланований час. У випадку, якщо тест-драйв існує, система заборонить створення ще одного тест-драйву у цей самий час.

```
asyncValidate: function(callback, scope) {
    // Виклик валідатора батьківської схеми. Якщо батьківську валідацію не пройдено, виконання наступоного коду не має сенсу
    this.callParent([function(response) {
    if (!this.validateResponse(response)) {
        return;
    }
    // метод для виконання асинхронного коду - передаємо функцію, що перевірить результат асинхронного виклику
    // і функцію, що викличе зворотній виклик, який обробить отриманий об'єкт response
    Terrasoft.chain(
        function(next) {
            if(this.get('Type').displayValue === "Тест-драйв") {
            this.checkSchedule(function(response) {
                if (this.validateResponse(response)) {
                    next();
                }
            }, this);
        }else {
            next();
        }},
        function() {
            callback.call(scope, response);
        },
        this);
}, this]);
```

Рисунок 3.4.1 Реалізація функції asyncValidate

Функції asyncValidate (див. рис. 3.4.1) і checkSchedule (див. рис. 3.4.2) є властивістями об’єкта methods, що зберігає в собі логіку клієнтської схеми. asyncValidate перевизначає відповідний метод базової схеми, таким чином новий функціонал додається у логіку клієнтських схем.

```
checkSchedule: function(callback, scope) {
    var result = {success: true};
    //Отримуємо об'єкти колонок запису
    var start = this.get("StartDate");
    var end = this.get("DueDate");
    var type = this.get("Type");
    //Terrasoft.EntitySchemaQuery - об'єкт, який дозволяє надсилати на сервер асинхронні запити із заданими фільтрами
    var esq = Ext.create("Terrasoft.EntitySchemaQuery", { rootSchemaName: "Activity" });
    // Один виклик addColumn - одна колонка в SELECT побудованого SQL-запиту
    esq.addColumn("StartDate", "Start");
    esq.addColumn("Type", "Type");
    // логічна операція між предикатами у WHERE
    esq.filters.logicalOperation = Terrasoft.LogicalOperatorType.AND;
    // Колекція filters зберігає об'єкти, кожен з яких відповідає одному предикату в WHERE побудованого SQL-запиту
    esq.filters.addItem(esq.createColumnBetweenFilterWithParameters("StartDate", start, end));
    esq.filters.addItem(esq.createColumnFilterWithParameter(Terrasoft.ComparisonType.EQUAL, "Type", type));
    // Запуск запиту і передача функції зворотнього виклику, яка перевірить наявність тест-драйвів у зазначений в завданні час
    esq.getEntityCollection(function (result) {
        if (result.success && result.collection.getCount() > 0) {
            result.message = "Вже створено тест-драйв на цей час.";
            result.success = false;
        }
        callback.call(scope || this, result);
    }, this);
},
```

Рисунок 3.4.2 Реалізація функції checkSchedule

Функція checkSchedule виконує запит до бази даних з метою пошуку тест-драйвів, що проходитимуть у час створюваного в даний момент. Якщо такий тест-драйв існує, в об’єкті response встановлюється повідомлення про помилку, яке буде відображене на клієнті після відпрацювання зворотнього виклику.

Висновки

Хоча low code\no code підхід і пропонується компаніями-розробниками платформ для створення систем управління бізнесу як такий, що полегшує і пришвидшує розробку застосунку, він потребує високого рівня навичків володіння комплексним інструментарієм, який такі платформи пропонують. Як графічні інструменти, так і бібліотеки класів потребують умілого і розумного використання, існує багато способів вирішення однієї і тієї ж задачі. Незважаючи на зовнішню простоту, розробка на готових платформах є комплексним процесом, який має свої складності.

Розробка рішення з нуля надає більшу прозорість і гнучкість порівняно з використанням існуючої платформи. Цей підхід здатен покрити більшу кількість потреб, але його ціна, час і відносна складність інтеграції і модифікації роблять його непривабливим для бізнесу, поки не існує вагомих причин для його використання.

Протягом виконання даної роботи було розроблено CRM-застосунок, який надає можливості планування тест-драйвів, відслідковування процесу переведення потенційного клієнта в клієнти, ведення продажів і обробки сервісних звернень.

Дану систему в майбутньому можна покращити наступним чином:

- розширити функціонал аналітики;
- застосувати моделі машинного навчання для предиктивної оцінки показників компанії;
- додати можливість реєстрації автомобілів.

Список використаних джерел

1. Web Frameworks [Електронний ресурс] // StackOverflow. – 2020. https://insights.stackoverflow.com/survey/2020#most-popular-technologies.

2. What is Django? [Електронний ресурс] https://realpython.com/tutorials/django/.

3. Django [Електронний ресурс] // Full Stack Python https://www.fullstackpython.com/django.html.

4. Spring Documentation [Електронний ресурс] https://spring.io.

5. CRM системи для бізнесу (рос.) [Електронний ресурс] https://avada-media.ua/services/crm-system/.

6. Low-Code and No-Code: What's the Difference and When to Use What? [Електронний ресурс] // OutSystems. – 2021. https://www.outsystems.com/blog/posts/low-code-vs-no-code/.

7. Creatio Academy [Електронний ресурс] // Terrasoft https://academy.terrasoft.ua/.