МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
Катерина НЕСТЕРЕНКО
“____”_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
“БАКАЛАВР”

Тема: “Цифрова платформа інформаційної підтримки малого бізнесу у сфері надання послуг”

Виконавець: Ковецький Валерій Віталійович

Керівник: к.т.н. Задонцев Юрій Вікторович

Нормоконтролер: Варнавський В’ячеслав Володимирович

Київ 2024

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення
Освітній ступінь бакалавр
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Програмне забезпечення систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Катерина НЕСТЕРЕНКО
"___" _____ 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи студента
Ковецького Валерія Віталійовича

1. Тема проекту: «Цифрова платформа інформаційної підтримки малого бізнесу у сфері надання послуг»
затверджена наказом ректора від 08.12.2023 р. № 2483/ст

2. Термін виконання проекту: з 08.12.2024 р. до 29.02.2024 р.

3. Вихідні дані до проекту: веб-платформа для самозайнятих спеціалістів у сфері надання послуг та веб-платформа для клієнтів.

4. Зміст пояснювальної записки:
1. Аналіз поточного стану процесів в предметній області.
2. Вимоги до цифрової платформи інформаційної підтримки малого бізнесу у сфері надання послуг.
3. Структура цифрової платформи інформаційної підтримки малого бізнесу у сфері надання послуг.
4. Прототип програми.

5. Перелік обов'язкових слайдів презентації:
1. Проблематика ведення бізнесу у сфері надання послуг.
2. Наявні підходи до розв’язання проблеми малими бізнесами та самозайнятими спеціалістами.
3. Запропонований варіант для розв’язання проблеми.
4. Способи реалізації цифрової платформи для інформаційної підтримки малого бізнесу у сфері надання послуг.
5. Результати роботи.

6. Календарний план-графік

№ пор	Завдання	Термін виконання	Відмітка про виконання
1.	Складання та затвердження графіку кваліфікаційної роботи, написання допоміжних сторінок - титульної, завдання, графіка, реферат, список скорочень, зміст, вступ, список джерел.	08.12.24 – 21.01.24	
2.	Написання Розділу 1**.** Аналіз поточного стану процесів в предметній області. Написання Розділу 2. Аналіз вимог до платформи.	22.01.24 – 28.01.24	
3.	Написання Розділу 3. Побудова структури цифрової платформи інформаційної підтримки малого бізнесу у сфері надання послуг.	29.01.24 – 04.02.24	
4.	Написання Розділу 4. Прототип програми. Розробка тексту доповіді. Оформлення графічного матеріалу для презентації	05.02.24 – 11.02.24	
5.	Проходження нормоконтролю. Отримання відгуку керівника. Відсилка ПЗ для перевірки на плагіат. Друк ПЗ.	12.02.24 – 18.02.24	
6.	Передзахист. Допуск до захисту. Рецензування. Подача документів секретарю ДЕК.	19.02.24 – 25.02.24	
7.	ДЕК. Захист.	26.02.24 – 29.02.24	

7. Дата видачі завдання 08.12.2024

Керівник: к.т.н. Юрій ЗАДОНЦЕВ
Завдання прийняв до виконання: Валерій КОВЕЦЬКИЙ
Дата

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Цифрова платформа інформаційної підтримки малого бізнесу у сфері надання послуг»: 53 с., 16 рис., 18 інформаційних джерел.

ІНФОРМАЦІЙНА ПІДТРИМКА, ЦИФРОВА ПЛАТФОРМА, МАЛИЙ БІЗНЕС, НАДАННЯ ПОСЛУГ, ПРОТОТИП ПЛАТФОРМИ.

Об’єкт розробки – Цифрова платформа інформаційної підтримки, яка призначена допомогти малим підприємствам та фізичним особам підприємцям у сфері надання послуг у вирішенні адміністративно-інформаційних питань.
Мета роботи – розширення можливостей та допомога у вирішенні адміністративних питань малого бізнесу у сфері надання послуг з ведення звітностей, розкладу працівників та клієнтської бази тощо.

ABSTRACT

Explanatory note to the qualification work "Digital platform for information support of small businesses in the field of service provision": 53 pages, 16 figures, 18 information sources.

INFORMATION SUPPORT, DIGITAL PLATFORM, SMALL BUSINESS, SERVICE PROVISION, PLATFORM PROTOTYPE.

The object of development is a digital platform of information support, which is designed to help small businesses and individual entrepreneurs in the field of service provision in solving administrative and informational issues.
The purpose of the work is to expand opportunities and help in solving administrative issues of small businesses in the field of reporting services, employee scheduling and client base, etc.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

БД – база даних

ФОП – фізична особа підприємець

SQL – structured query language

SMS (СМС) – short message service

IAAS – infrastructure as a service

PAAS – platform as a service

AWS – Amazon Web Services

ВСТУП

Сфера надання послуг – одна з найпопулярніших та найприбутковіших сфер бізнесу в наш час. Вона охоплює салони краси, СТО, перукарні, різноманітні гуртки, мовні курси, медицину, та багато іншого. Усі вони надають ті чи інші послуги, а клієнти – звичайні фізичні особи, вік яких залежить від сфери послуг. Частка малих бізнесів постійно збільшується, утворюючи конкурентне середовище на ринку.

З кожним роком дедалі більше з'являється самозайнятих осіб, що надають свої послуги як спеціалісти тих чи інших справ, нерідко працюючи в маленьких орендованих приміщеннях або ж з власного житла. Самозайняті спеціалісти – найвразливіша частина малого бізнесу, оскільки, як ніхто інший, потребують підтримки в розвитку та розголошенні суспільством. Держава, своєю чергою, всіляко підтримує їх, надаючи спрощені системи оподаткування та навіть грошові гранти на розширення. І це не дивно, оскільки малий бізнес та самозайняті особи напряму сприяють розвитку інновацій та зміцненню економіки країни, тим самим покращуючи рівень життя. Лише за 2023 рік було зареєстровано понад 300 тисяч нових ФОП в Україні, що випереджає навіть довоєнні показники на 7%.

Рис. 1.1. Графік реєстрації нових ФОП в Україні за останні 3 роки.

Ми живемо в час цифровізації послуг і вже не уявляємо своє життя без них. Оплачуємо послуги картками та віртуальними рахунками іноземних платіжних систем. Товари переважно замовляємо через інтернет, а замість

паперових документів використовуємо Дію. Це все безперечно робить наше життя простішим, безтурботнішим та заощаджує левову частку нашого часу, який є безцінним людським ресурсом. На сьогодні, Україна є однією з найбільш диджиталізованих країн світу, і впевнено зростає у рейтингу з кожним роком. Більша частина населення, від найстаріших до наймолодших, впевнено користується смартфонами та інтернетом. Навіть спілкування з найдорожчими для нас людьми вже перейшло від телефонних дзвінків до месенджерів. Тому не дивно, що частка послуг та процесів, що набувають автоматизації та цифровий вид, невпинно зростає. Це той тренд, що залишиться одним з найпопулярніших та найприбутковішим на довгі роки вперед. А ті, хто виявляться найкреативнішими та найініціативнішими, отримають славу та величезну частку ринку.

На плечі будь-якого бізнесу покладається велика відповідальність та багато обов’язків. Залежно від розміру бізнесу, варіюється ступінь проблем, з якими вони стикаються, і шляхи їх вирішення. Давайте детальніше розглянемо проблематику ведення бізнесу у сфері надання послуг для малого бізнесу та самозайнятих осіб в наш час в Україні.

РОЗДІЛ 1.
АНАЛІЗ ПОТОЧНОГО СТАНУ ПРОЦЕСІВ В ПРЕДМЕТНІЙ ОБЛАСТІ.

1.1. Проблематика з точки зору малого бізнесу.

Якщо подивитись на ситуацію з точки зору малого підприємця, що працює у сфері надання послуг, то одними з найскладніших моментів є ведення звітностей, розкладу працівників та клієнтської бази. Кожен спеціаліст, що надає послуги, має свої робочі дні та години та надає лише певний спектр послуг. Залежно від досвіду працівника, ціна на послуги також може змінюватись. Для певних малих бізнесів справедливим фактом є також й те, що заробітна плата працівників частково або повністю залежить від виконаного обсягу роботи за певний проміжок часу, тож вкрай необхідно мати облік всіх наданих послуг кожним з працівників. На додаток до цього, кожен підприємець зобов'язаний вести облік фінансових операцій, робити звітність прибутку, та вчасно сплачувати податки. У середніх та великих бізнесів за це відповідають окремі особи, що мають відповідні знання та навички. У малого ж бізнесу, і тим більш у самозайнятих осіб, багатьма процесами може займатись одна особа, і не рідкість, коли ця особа – власник.

В умовах сучасної ринкової конкуренції, важливим є залучення нових клієнтів та щонайдовше утримання постійних. Для цього, окрім як надання щонайкращих послуг, важливо також мати налагоджену двосторонню комунікацію з клієнтами, бути відкритими та прозорими. В наш час, в еру

				ПОСЛУГ	

цифрових технологій, дані потреби частково закриваються за допомогою соціальних мереж, особливо Instagram. Це прекрасний інструмент для бізнесу будь-якого розміру, аби продемонструвати свої послуги, прейскурант та отримувати зворотну відповідь від клієнтів. Проте, як і слід очікувати, це лише додатковий інструмент, який чудово поєднується в сумі з іншими, та розрахований лише на певну категорію людей. Йому (Instagram), як бізнес-інструменту, бракує досить великої кількості функціонала, щоб дати бізнесу можливість повноцінно себе зарекомендувати потенційному клієнтові. По-перше, це робота з відгуками. Клієнтові, перед тим як звернутись та отримати певну послугу, вкрай необхідно мати впевненість, що він отримає бажаний результат та не залишиться розчарованим. Якщо ще 10-15 років назад люди здебільшого отримували відгуки через газети або знайомих, то наразі цю функцію виконує інтернет. Повертаючись до Instagram, в ролі відгуків можуть слугувати коментарі до публікацій на сторінці, проте вони не є централізованими, а що ще більш важливо – повністю підконтрольні власнику. Тобто отримавши, скажімо, поганий відгук на сторінці, автор має повну можливість його видалити, а вас – заблокувати. Це робить Instagram абсолютно нерелевантним джерелом рейтингу та відгуків. Безсумнівно, що для виконання цієї ролі є також немала частина виділених під це вебсайтів, проте навряд чи там вийде знайти відгук на невелику перукарню біля житлового будинку, або ж тим більш на самозайнятого майстра, що працює “на дому”.

Приймання записів від клієнтів – теж не найлегший процес. Зазвичай, приймання записів здійснюють або робітники, або ж сам власник бізнесу, використовуючи телефонні дзвінки або повідомлення в месенджерах. Це додає певні складності до процесу та забирає немалі ресурси, оскільки з кожним клієнтом потрібно індивідуально узгодити дату та час, та впевнитись в можливості присутності потрібного фахівця.

Це все додатковий час, на який виділяються чималі ресурси щодня, які могли б бути спрямовані на інші, більш складні та вибагливі до навичок процеси. На жаль, автоматизувати цей процес за допомогою елементарного чат-бота також не має змоги, оскільки для побудови вільних дат та годин має бути доступ до графіка всіх спеціалістів, а також до вже зібраних записів, тобто фактично – до всього внутрішнього обліку. Якщо розглядати більш примітивний варіант імплементації чат-бота, що просто збирає бажані дати клієнтів, то це не сильно покращить ситуацію, оскільки ручний процес з опрацювання запитів все одно залишиться.

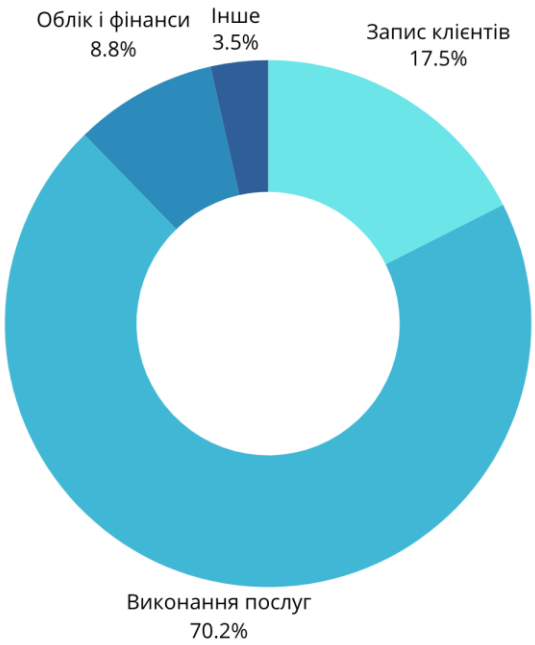

Рис. 1.1. Розподіл часу у спеціалістів зі сфери надання послуг.

Якщо аналізувати підходи середніх та великих бізнесів, то стає чітко зрозуміло, що всі вище перелічені проблеми не є для них такими. Частіше за все у подібних підприємств або розроблені власні програмні продукти, або є ліцензії на готові рішення. Це дозволяє їм максимально автоматизувати всю ручну роботу та збільшити виділення людського ресурсу на більш трудозатратні задачі. Ведення обліку клієнтів, побудова графіку роботи спеціалістів та фінансова звітність для податкової – лише невеликий перелік тих функцій, до яких мають доступ середньовеликі бізнеси. Розробити власний

програмний продукт, або купити ліцензію на готове рішення може бути досить складною задачею для малого підприємця, особливо якщо підприємець і є спеціалістом, що надає послуги. Це, в першу, але не останню, чергу – фінансова сторона питання, а також неповне усвідомлення усіх переваг, що надають подібні інструменти для автоматизації.

1.2. Проблематика з точки зору клієнта.

Розглядаючи ситуацію під іншим кутом, а саме – клієнта, можемо чітко усвідомити найголовніший недолік, що є справедливим для всіх без винятку розмірів бізнесу у сфері надання послуг – запис на послугу. Як ми вже трохи раніше визначили, основним джерелом для приймання записів і досі залишаються телефонні дзвінки. Використовуючи гугл в пошуках найближчого до вас СТО або салону краси, ви скоріш за все вийдете на номер телефону підприємства. Саме ним ви й скористуєтесь, аби отримати базову інформацію про доступні послуги, вартість, та що найголовніше – вільні дати для запису. Більш сучасні підприємства можуть мати власну сторінку в соціальних мережах або навіть власний вебсайт, де можна частково знайти інформацію, що вас цікавить. Проте для уточнення графіку роботи та запису ви все одно вимушені подзвонити.

Можна довго перелічувати повний список причин, з яких ви не маєте змогу зателефонувати цієї миті. Це і неможливість дзвінків на стаціонарні номери телефону з вашого тарифного плану, і відсутність гарного мобільного зв’язку у вашій локації, і навіть найбанальніше – несприятливі умови для голосової розмови в цю хвилину. Враховуючи поточну еру цифрових технологій і те, як сильно наразі поширені текстові повідомлення, нікого не має дивувати той факт, що телефонні дзвінки стають поволі застарілим способом зв’язку, відходячи на задній план. Дедалі більше можна зустріти в наш час підприємств, що впроваджують приймання записів та спілкування з клієнтами через месенджери, проте це досі не настільки поширено, як хотілось би, і, відверто кажучи, все одно не є ідеальним рішенням для клієнта, оскільки

відповідь від представника бізнесу можна очікувати іноді годинами, а особливо у випадку з малим бізнесом або самозайнятими спеціалістами.

Також не забуваємо й про те, що клієнт має витратити власний час, аби повноцінно проаналізувати всі підприємства на основі їх вебсайту, оголошень чи сторінок в соціальних мережах, або ж прочитати відгуки на виділених для цього вебплатформам. Далеко не кожна людина буде витрачати на це час, і скоріш довіриться красивим заголовкам та фотографіям підприємства з першого в видачі Google вебсайту. Якщо ж клієнт після отриманої послуги залишиться незадоволеним, він буде в пошуках платформи, щоб залишити свій негативний досвід та застерегти інших потенційних клієнтів. Навіть якщо таку платформу він і знайде, від його негативного коментаря буде не так багато ефективності, оскільки інші люди скоріш за все так само не витрачатимуть час на їх пошук. Це все приводить нас до того, що клієнт не має централізованого рішення, де міг би знайти одразу всю цікаву для нього інформацію: рейтинг бізнесу, незалежні відгуки від попередніх клієнтів, загальну інформацію про підприємство, його послуги, ціни, та, безумовно, можливість запису без додаткового спілкування з представниками.

1.3. Розв'язання проблеми.

Більша частина малих підприємців, задля мінімізації витрат, наразі використовують досить примітивні способи розв'язання вищеописаних проблем: ведення графіку працівників здійснюють за допомогою звичайного застосунку календаря або ж Excel таблиць. Клієнтську базу, облік витрат, доходів та наданих послуг також найчастіше ведуть за допомогою Excel таблиць або аналогічних ним, оскільки оберти не настільки великі, і даний варіант цілком підходить, хоч і бракує багато функціонала для автоматизації процесів. Приймають записи від клієнтів же за допомогою телефонних дзвінків або через месенджери (Instagram, Telegram, WhatsApp). Це, як ми впевнились трохи раніше, додає певних складнощів як потенційним клієнтам, так і самим спеціалістам, оскільки вони мають час від часу відволікатись на

повідомлення/дзвінки та витрачати час на планування свого графіка, у той час як потенційні клієнти мають витрачати час на дзвінок/повідомлення та будувати в голові вільні місця та години спеціаліста, зіставляючи їх зі своїми власними планами.

Потрібне комплексне рішення, що зможе розв'язати усі вищеописані проблеми як для малих підприємців, так і для клієнтів. Підприємцям вкрай важливо автоматизувати весь процес ведення записів, графіку та клієнтської бази, а клієнту важливо мати зручний інструмент для отримання інформації про кожну послугу, її тривалість, вартість та вільні години майстра, а також – оформити запис. Рішенням проблеми має слугувати онлайн-платформа, на якій підприємці можуть зареєструвати свій бізнес та автоматизувати всі вищеперелічені задачі. Клієнти ж своєю чергою, використовуючи унікальний QR код або вебпосилання, переходитимуть на сторінку підприємця, де матимуть змогу переглянути всю необхідну їм інформацію, таку як: рейтинг, відгуки, список послуг, що надаються, та вільні години, на які можна одразу ж оформити запис онлайн.

1.4. Аналіз українського ринку.

Наразі на українському ринку не існує фактично жодного рішення, що вирішувало б одразу всі вищеописані проблеми, та слугувало б платформою як для бізнесу, так і для клієнтів. На сьогодні найбільш розвинуті саме онлайн-платформи для бізнесу, що надають можливості з будування звітностей, будування графіків роботи працівників та веденню обліку клієнтської бази та фінансових операцій. Найпопулярніша онлайн-платформа в Україні, що надає подібний спектр послуг – Altegio[1].

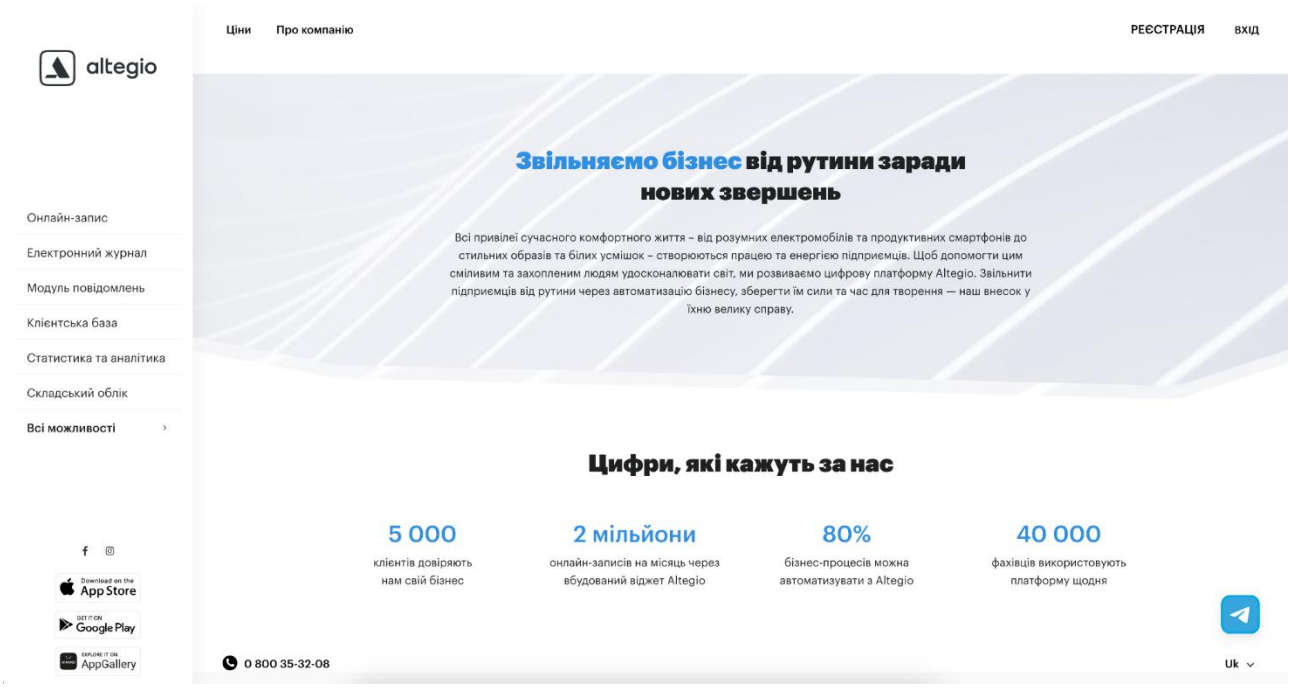

Рис. 1.2. Цифрова платформа Altegio.

У сервісу, окрім вже перелічених, присутня і можливість зі створення сторінки для прийняття онлайн-записів від клієнтів. Це додаткова функція, яка є опційною і вмикається через особистий кабінет. На основі тих послуг і працівників, що вказані у вас в налаштуваннях, сервіс будує унікальну вебсторінку, через яку клієнти мають змогу оформити онлайн-запис на зацікавлену послугу без додаткового спілкування з будь-ким. Сторінка може працювати як незалежно, так і бути інтегрованою з вашим наявним вебсайтом підприємства. Також Altegio пропонує купу різних інтеграцій зі стороннімі системами, включаючи фінансові інструменти для безготівкових операцій та автоматизації з ведення каси. Сервіс надає свої послуги по моделі щомісячної підписки, вартість якої напряму залежить від кількості спеціалістів, що надають послуги у вашому підприємстві. Одразу хочу зазначити той факт, що дана платформа, яка нагадаю є найпопулярнішою на сьогодні в Україні, має повністю російське коріння. Її власники та розробники – росіяни, які після 24 лютого 2022 року зробили ребрендинг для подальшої роботи на європейському ринку. Хоча вони й повністю заперечують своє відношення до Росії, проте

навіть неозброєним оком можна побачити, що між європейською та російською версіями немає жодної відмінності окрім іншої літери на логотипі та назви.

З суто українських аналогів найпопулярнішим рішенням є платформа Bookon[2]. Вона надає схожий спектр інструментів для автоматизації ведення бізнесу у сфері надання послуг та чудово підходить як самозайнятим, так і великим мережам.

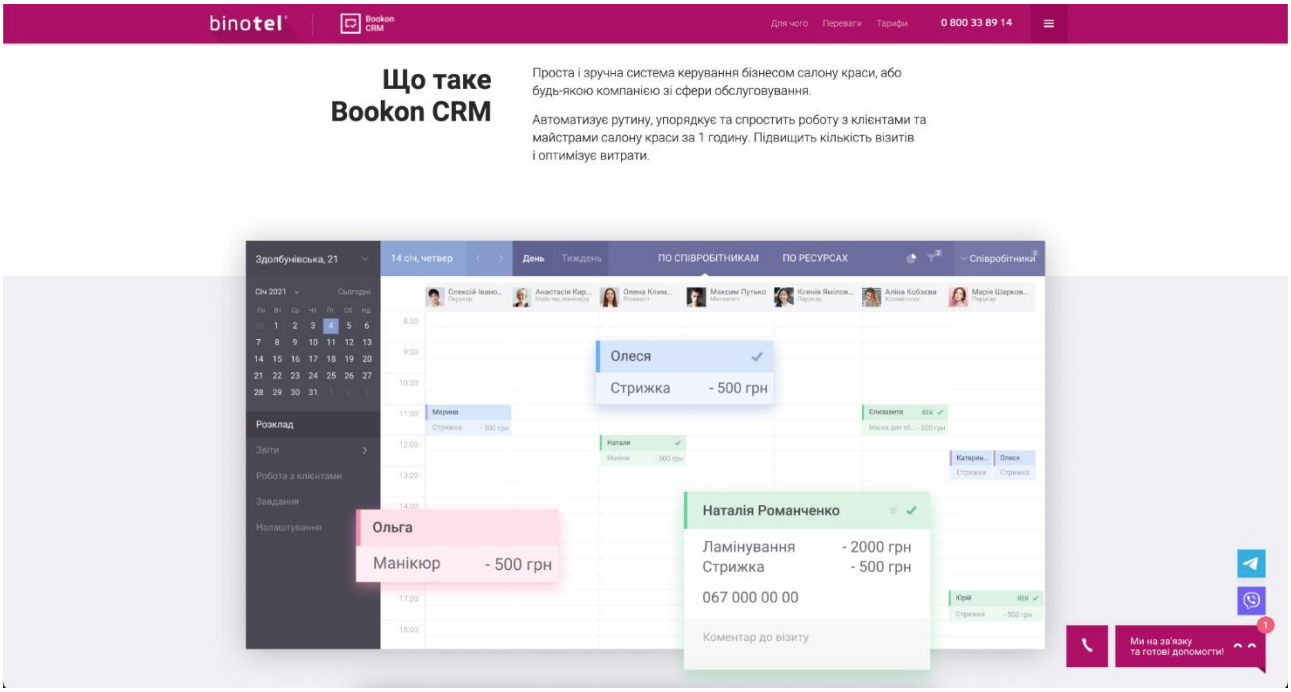

Рис. 1.3. Цифрова платформа Bookon.

Працює за ідентичною моделлю підписки, де вартість залежить від кількості спеціалістів, що працюють у вашому бізнесі. Проте сервісу бракує багатьох функцій попереднього, включаючи створення індивідуальної сторінки для прийняття онлайн-записів. Як перша, так і друга платформа, свою увагу приділяють саме бізнесу, та майже не надають жодних переваг для клієнтів, що звертатимуться за послугами. Майже всі проблеми клієнтів, які ми обговорювали раніше, залишаються актуальними й у випадку використання бізнесами цих платформ.

Хочу також зазначити, що як перша, так і друга платформа, хоч і підходять для малого бізнесу (або ж самозайнятих осіб), проте не надають

пільгового доступу до своїх інструментів, чим фактично спонукають таких осіб шукати альтернативи та використовувати інструменти, що не відповідають повним їх потребам (календар, Excel тощо.). Це і є найголовнішою причиною того, що серед українських самозайнятих спеціалістів та малого бізнесу не існує єдиного зручного інструменту, що надавав би й можливість з автоматизації рутинних завдань бізнесу, так і зручного рішення для їх клієнтів, де вони мали б можливість для ознайомлення з послугами, їх цінами, відгуками від інших людей та, що найголовніше – можливість відразу створити онлайн-запис без додаткового спілкування з представниками підприємства.

Якщо розглянути інші сфери бізнесу, то ситуація зовсім інакша. Наприклад, товарний бізнес. Наразі в Україні працює чимало цифрових платформ для підтримки малого бізнесу в цій сфері, з найпопулярніших: OLX, Izi, Prom. Всі вони надають безплатний інструмент для власників малого бізнесу на розміщення своїх товарів, а клієнтам – зручну та безпечну купівлю в інтернеті. Так само ці платформи мають і систему рейтингів з відгуками, що дозволяє клієнтам впевнитись в надійності продавця та убезпечити себе від неприємного досвіду. Схожа ситуація присутня і в ресторанному бізнесі. З недавнього часу Monobank запустив свою власну онлайн-платформу для підтримки бізнесу в цій сфері - Expirenza[3]. Суть полягає в тому, що власники кафе або ресторану створюють свій унікальний QR-код, який розміщується на кожному столі в залі. Клієнти, відсканувавши його, мають все меню закладу у себе в телефоні, з фотографіями страв та детальним описом. По закінченню трапези, клієнти знову сканують той самий QR-код та мають змогу оплатити рахунок банківською карткою без присутності офіціанта. Також є можливість залишити чайові в бажаному розмірі, та залишити свій відгук. Даний сервіс спрощує життя абсолютно всіх в ресторанному бізнесі. Офіціанти більше не переплутають суму на терміналі, гостю закладу не потрібно чекати офіціанта для оплати банківською картою, а власник може переглянути статистику у власному кабінеті та прочитати відгуки залишені клієнтами. До цієї системи

вже долучилось чимало закладів харчування в Україні, створивши таким чином єдину уніфіковану платформу.

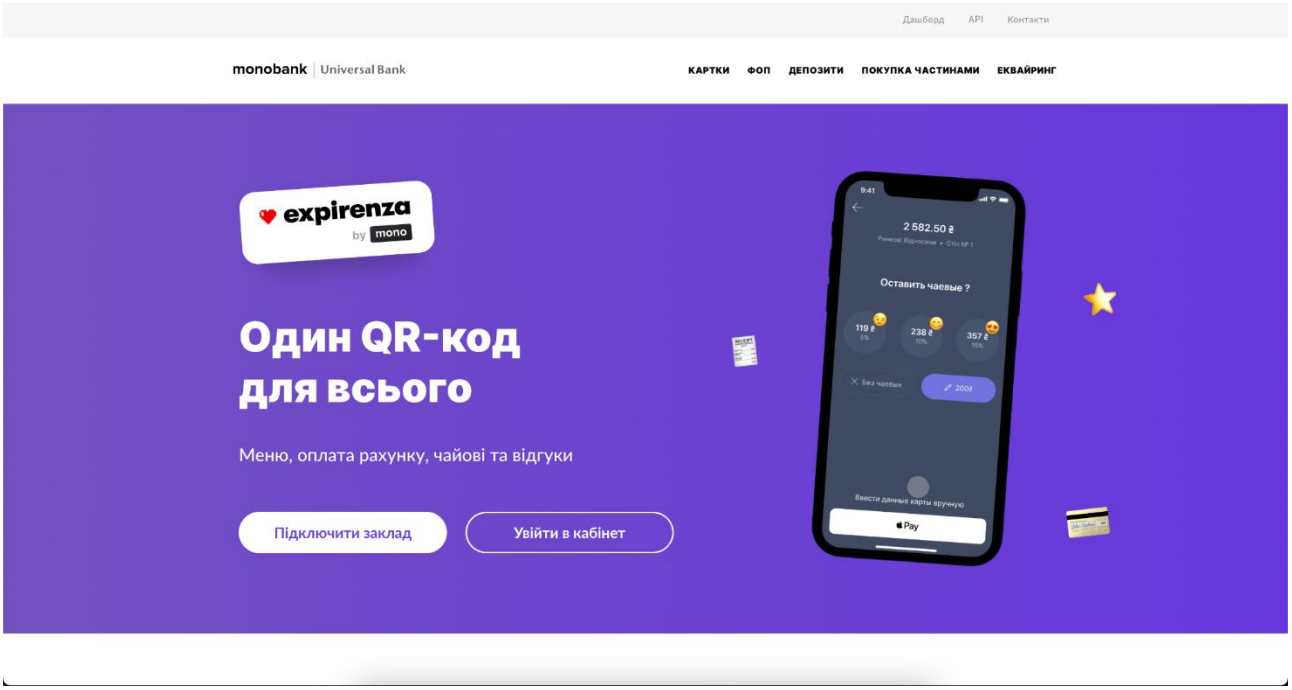

Рис. 1.4. Онлайн платформа Expirenza.

1.5. Аналіз західного ринку.

Цікавіша ситуація відбувається на європейському та американських ринках. Там є досить велика варіативність онлайн-платформ для бізнесів у сфері надання послуг. Розглянемо найпопулярніші з них. Vagaro[4] - онлайн-платформа як для бізнесу, так і їх клієнтів. Надає великий спектр по автоматизації усіх внутрішніх процесів, такі як облік наданих послуг та фінансових операцій, будування звітностей для податкової та інтеграції з платіжними системами для прийняття безготівкових платежів. Також Vagaro має платформу для клієнтів, де вони можуть знайти потрібну їм послугу у вибраному районі, подивитись відгуки по кожному майстру та зробити запис онлайн. Наразі платформа працює тільки у Сполучених Штатах, Великобританії та Австралії. Компанія має дуже цікаву історію заснування, оскільки вона бере свій початок з 1999 року, коли майбутній засновник вперше звернув свою увагу на проблематику бізнесу у сфері надання послуг. Його не

влаштовував підхід з тим, що для запису обов’язково потрібно телефонувати у заклад, і фактично немає можливості дізнатись його рейтинг, особливо знаходячись в іншому місті або країні. Оскільки на той час не було достатньої кількості технологій для розв'язання проблеми, ця ідея так і залишилась в його думках аж до 2008 року, коли вперше був представлений iPhone та почалась нова ера мобільних застосунків. Для подальшого втілення своєї ідеї в реальність, Фред, він же засновник компанії, вирішив придбати 3 салони краси у своєму рідному місті, аби більше поглинутись та зрозуміти усі проблеми, що існують у цій сфері. У 2009 році платформа була запущена в її першому вигляді, а на сьогодні має капіталізацію більш як 1 мільярд доларів США та надає свої послуги більш ніж 208 тисячам бізнесам у сфері надання послуг у США, Великобританії та Австралії.

Ще однією з дуже популярних онлайн-платформ є Fresha[5]. Він доступний у більшості країн світу, переважна більшість з яких саме європейські. Сервіс є дуже схожим по функціоналу з попереднім та надає платформу як бізнесу, так і клієнтам. Найголовніша відмінність цієї платформи від інших та її “візитна картка”, на якій вони роблять акцент у, без перебільшення, всіх своїх рекламних кампаніях – безплатність. Сервіс є повністю безплатним для бізнесів будь-якого розміру у сфері надання послуг, та може використовуватись без обмежень за кількістю працівників чи обертам бізнесу. Проте, як і слід очікувати, безоплатний доступ надається тільки до базових функцій сервісу, а більш комплексні, такі як будування аналітичних звітностей чи управління персоналом, доступні тільки преміум клієнтам, плата за статус якої стягується додатково. Також, сервіс може брати певний відсоток з вартості кожної наданої послуги, що була оброблена їх платформою. Загалом, сервіс має гарну репутацію та велику кількість задоволених клієнтів, яка з моменту запуску у 2015 році піднялась до відмітки у 100 тисяч.

Висновки за розділом 1.

Підсумовуючи, можемо чітко зазначити, що тема платформи для підтримки малого бізнесу у сфері надання послуг є вкрай актуальною сьогодні, особливо в Україні, оскільки наразі існує не так багато готових рішень, що повноцінно закрили б вимоги як бізнесу, так і їх клієнтів. Вважаю цілком доцільну розробку та підтримку подібної платформи на внутрішньому ринку, створивши таким чином чудову альтернативу європейським аналогам, що є абсолютно не пристосованими для українського ринку та не відповідають його вимогам.

Як вже зазначалось, саме сфера надання послуг наразі є найменш розвинутою з автоматизації в Україні та не має сучасних, уніфікованих рішень. Це чудова можливість переосмислити наявні підходи в цій сфері, та створити нові тенденції. Це поштовх в майбутнє, вплив на розвиток самозайнятих осіб і малого бізнесу, та заохочення нових спеціалістів доєднатись до нього.

РОЗДІЛ 2.
АНАЛІЗ ВИМОГ ДО ЦИФРОВОЇ ПЛАТФОРМИ ІНФОРМАЦІЙНОЇ ПІДТРИМКИ МАЛОГО БІЗНЕСУ У СФЕРІ НАДАННЯ ПОСЛУГ.

2.1. Аналіз бізнес вимог до платформи.

Головна мета платформи – докорінно розв'язати найболісніше питання малого бізнесу та самозайнятих осіб в Україні, а саме надати уніфіковане, просте та зручне рішення по автоматизації записів, веденню календаря робітників та клієнтської бази. А клієнтам – зручну, сучасну платформу для пошуку закладів, що надають потрібну їм послугу, можливість перегляду відгуків, рейтингу, списку послуг, їх вартість та можливість залишити запис на вільну дату та годину. На основі аналізу поточного стану ринку, можемо чітко стверджувати, що пріоритетним завданням є надання безплатного доступу до базових інструментів платформи, тим самим спонукаючи все більше самозайнятих осіб в Україні доєднатись та створити уніфіковану платформу онлайн-записів. Це безперечно вплине як на покращення двосторонньої комунікації між клієнтами та бізнесом, так і на оптимізацію використання ресурсів бізнесів.

На основі проведеного дослідження предметної області, сформуємо список бізнес-вимог до платформи:

1. Оптимізація ресурсів бізнесу у сфері надання послуг. Задля мінімізації ручної праці власників бізнесу, платформа має надати зручні інструменти для автоматизації обліку наданих послуг, фінансових операцій, ведення графіку спеціалістів та прийняття онлайн записів.
2. Популяризація малого бізнесу та самозайнятих осіб. Платформа, надаючи зручний інструмент зі створення власної публічної сторінки, сприяє популяризації бізнесу серед зацікавлених осіб та слугує візитною карткою.
3. Покращення сервісу. Маючи публічну сторінку бізнесу, власник має змогу отримувати позитивні або негативні відгуки від клієнтів, що скористались їх послугами, та робити зміни/покращення в послугах, що надаються.
4. Робота з клієнтами. Платформа має надати інструмент для побудови звітів по різним категоріям клієнтів, що своєю чергою дозволить власникам опрацьовувати їх в індивідуальному порядку та отримувати відгуки, або ж пропонувати спеціальні пропозиції (знижки, ваучери тощо).
5. Доступність. Власникам малого бізнесу або самозайнятим спеціалістам доступ до базового функціонала платформи має надаватись безоплатно.
6. Ефективність для клієнтів. Зручна єдина платформа, де клієнти матимуть доступ до отримання інформації про послуги, що їх цікавлять, ціни, відгуки, та можливість запису онлайн. Це має полегшити як і вибір закладу, так і зменшити час, що витрачається на запис, оскільки немає потреби в індивідуальній розмові з представником бізнесу.
7. Місцева адаптація. Розробка та підтримка платформи має базуватись на особливостях українського ринку та мати повну українську локалізацію.
8. Безпека та конфіденційність. Усі дані бізнесу та клієнтів мають надійно зберігатись для забезпечення високого рівня конфіденційності.

9. Підтримка багатьох напрямів. Сервіс має бути універсальним для всієї сфери бізнесу з надання послуг. Має бути підтримка різних напрямків, таких як: медицина, краса, навчання, спорт тощо.

2.2. Вимоги до платформи з боку власників бізнесу.

Маючи чітко визначені бізнес-вимоги до платформи, можемо скласти список функціональних та нефункціональних вимог до неї. Почнемо з платформи для бізнесу. Перш за все, обов'язковою складовою має бути вебсторінка, що слугуватиме візитною карткою проекту та розповідатиме про всі переваги платформи. Має бути вказаний перелік найважливіших інструментів, що надаються платформою, та яким чином вони можуть позитивно вплинути на подальший розвиток того чи іншого бізнесу у сфері надання послуг.

Наступною важливою функціональною вимогою є специфіка реєстраційного процесу. Цей етап має бути швидким та простим, а для досягнення цієї мети розіб'ємо його на 3 послідовних маленьких етапи:

- збір деталей для авторизації – електронна адреса, пароль.
- Збір деталей про бізнес – назва закладу/підприємства на його напрямок (медицина, краса, спорт тощо).
- Збір деталей про власника – ім'я та контактний номер телефону.

Таким чином, власник бізнесу, що розпочне процес реєстрації, зможе просто та швидко його завершити та перейти до користування платформою. Довгі процеси реєстрації з великою кількістю запитань та полями для вводу можуть негативно вплинути на досвід використання платформи та навіть відштовхнути потенційного клієнта.

Створивши бізнес-акаунт, користувач має побачити головну сторінку його власного кабінету, де інтерактивними спливаючими віконцями платформа розповість йому про основний набір інструментів та як ними користуватись. За бажанням, віконця можна одразу закрити, якщо користувач хоче ознайомитись з платформою самостійно. Це чудово допоможе зробити короткий огляд

функціонала для нового користувача, тим самим до мінімуму зменшивши кількість заплутаних та розгублених клієнтів.

Не менш важливим аспектом є й служба підтримки. Через особистий кабінет власники бізнес-акаунту повинні мати можливість під'єднатись до служби підтримки платформи, де зможуть поставити цікаві для них питання та отримати на них відповіді. За бажанням, клієнти можуть запросити онлайн-дзвінок, де представник платформи на онлайн-дзвінку разом з клієнтом детально пройдуться по всім функціям платформи та допоможуть її налаштувати під потреби саме їх бізнесу.

Внесення інформації про послуги, спеціалістів, та їх графіки роботи. Вкрай необхідно створити інтуїтивно зрозумілий інтерфейс з внесення інформації про послуги, що надаються бізнесом чи особою, їх вартість та тривалість, а також інформацію про кожного спеціаліста, що надає послуги, а саме: ім’я, фото (опційно), графік роботи (дні та години) та список послуг, що виконує.

Календар з графіком роботи спеціалістів. На головній сторінці власного кабінету власник має бачити велику секцію з календарем на поточний день з можливістю вибрати будь-який інший. Календар поділений на часові проміжки, де різноколірними смужками позначені записи до кожного зі спеціалістів, та вид послуги. Натиснувши на будь-який часовий проміжок, користувач може власноруч додати запис, заповнивши для цього всі необхідні поля: інформація про клієнта, послуга та спеціаліст, що буде залученим.

Графіки зі звітами за поточний та попередні місяці. Одним з інструментів платформи має бути сторінка, на якій секціями представлені графіки з різними метриками за вибраний місяць. Основні з метрик – прибуток бізнесу за календарний місяць, кількість обслугованих клієнтів та графік популярності послуг. Це надасть власникам можливість для подальшого аналізу та опрацювання цих метрик в будь-яких цілях.

Робота з відгуками. Ще однією дуже важливою складовою є система рейтингу та відгуків. Власник має бачити у своєму особистому кабінеті свій

загальний рейтинг за 5-ти бальною шкалою, що рахується як середнє значення всіх виставлених оцінок за весь період часу. Також мають відображатись відгуки клієнтів, що скористались послугами, на які можна залишити відповідь від імені бізнесу, проте без можливості видалення коментарів.

2.3. Вимоги до клієнтської платформи.

Перейдемо тепер до вимог щодо клієнтської платформи. Платформа для клієнтів являє собою окремі публічні сторінки підприємців, з якими вони можуть взаємодіяти. Кожен спеціаліст має власний ідентифікатор, який міститься у посиланні. Зайшовши на сторінку спеціаліста, клієнт може виконувати наступні дії:

1. Перегляд загальної інформації про спеціаліста.
2. Перегляд портфоліо робіт.
3. Перегляд відгуків від клієнтів, що скористались послугами.
4. Залишення власного відгуку про спеціаліста.
5. Перегляд наявних послуг, інформація про них, та створення запису.

Почнемо з першого пункту. Загальна інформація про спеціаліста містить в собі:

1. Публічна назва бізнесу.
2. Фотографія.
3. Графік роботи на поточний день.
4. Номер телефону.
5. Загальний рейтинг по 5-ти бальній шкалі.
6. Про себе (довільний текст).

Загальна інформація відображається одразу, як користувач заходить на сторінку та залишається статичною при переході клієнтом на інші вкладки. Даної інформації достатньо, аби користувач зробив перші висновки щодо того, чи підходить йому даний спеціаліст, та сконтактував з ним напряму, якщо це потрібно.

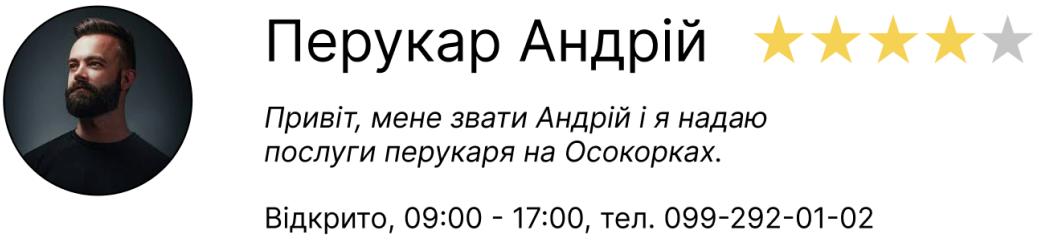

Рис. 2.1. Дизайн-макет для відображення загальної інформації.

Наступний пункт – портфоліо робіт. Портфоліо знаходиться на окремій вкладці та складається з масиву фотографій виконаних робіт майстром, між якими можна перемикатись, використовуючи стрілочки “вліво” та “вправо”. При натисканні на будь-яку з фотографій, вона відкривається на повний екран, що дає змогу детальніше її роздивитись. Весь задній фон затемнюється, фокусуючи всю увагу саме на фото. Елементи інтерфейсу для переходу між фото зберігаються. Портфоліо є універсальним та всім знайомим підходом для демонстрації своїх робіт. Гарно та правильно оформлене портфоліо надасть більше інформації для майбутніх клієнтів, та значно збільшить шанси на їх звернення.

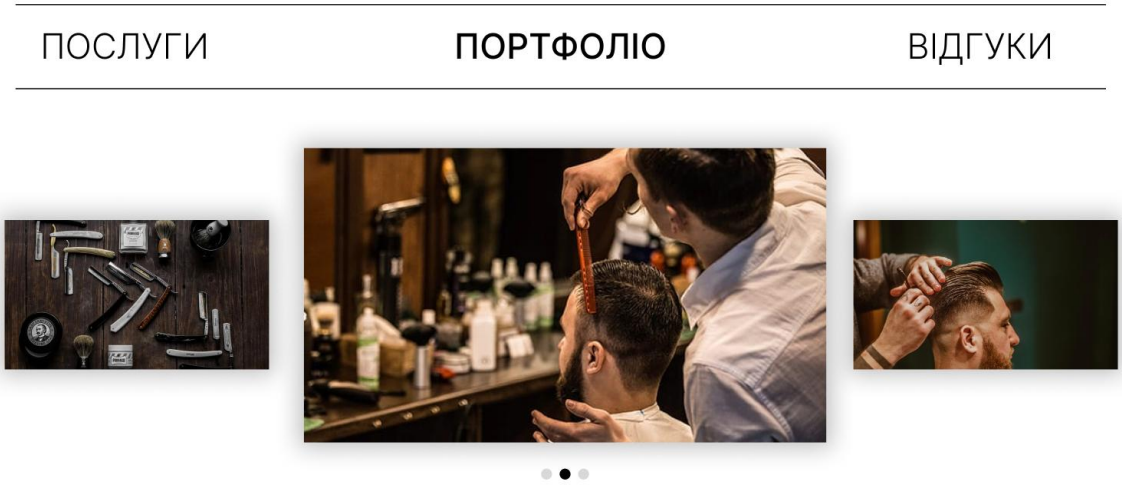

Рис. 2.2. Дизайн-макет секції з портфоліо робіт.

Відгуки є не менш важливим функціоналом для клієнтів. Вони так само як і портфоліо знаходяться на окремій вкладці, та містять список з текстових

відгуків від клієнтів, що вже скористались послугами даного майстра. Кожен відгук містить в собі наступну інформацію:

1. Ім’я клієнта, що скористався послугами. Або “Анонім”, якщо клієнт вирішив його не публікувати.
2. Рейтинг, який клієнт залишив для майстра по 5-ти бальній шкалі.
3. Текстовий відгук (опційно).
4. Дата, коли відгук було залишено.
5. Назва послуги, якою скористався клієнт.

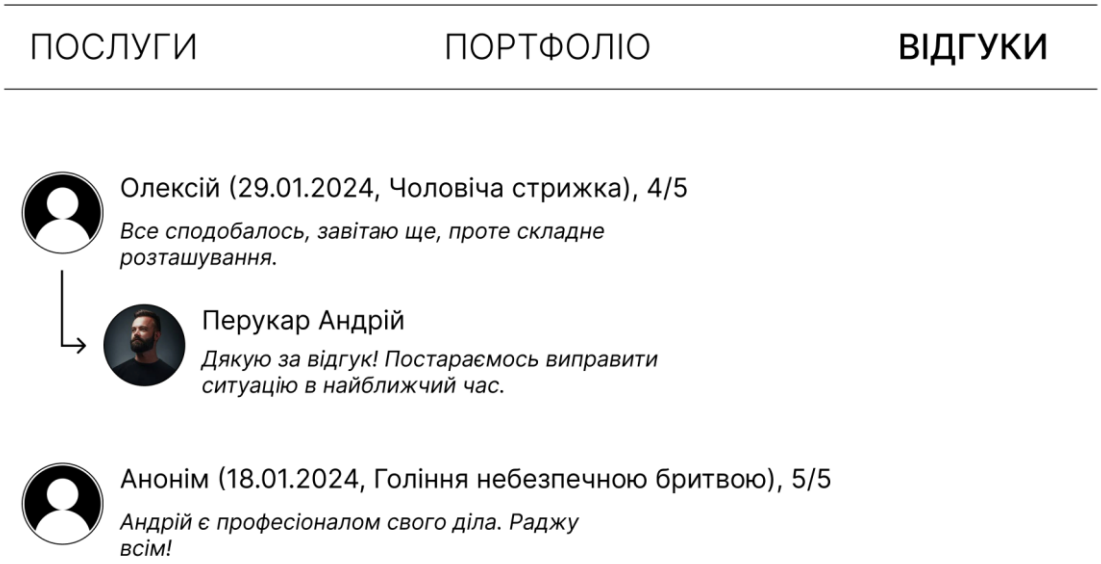

Рис. 2.3. Дизайн-макет секції з відгуками.

Залишити власний відгук можливо виключно після того, як спеціаліст підтвердив виконання послуг по запису. Клієнт отримує електронний лист або СМС-повідомлення з унікальним посиланням, яке дійсне протягом наступних 3-х діб. Перейшовши по ньому, клієнт має змогу залишити свою оцінку щодо наданих послуг (по шкалі від 1 до 5), та опційно вказати своє ім’я та довільний текст. Після відправлення відгуку, внутрішня система аналізує написаний текст на наявність ненормативної лексики та робить відгук публічним для перегляду іншим користувачам. На кожен відгук від клієнта, власник бізнесу може залишити 1 відповідь. Це дозволить як пояснити спірну ситуацію, що описана у

відгуку, так і подякувати клієнту за залишений позитивний відгук. Відгуки слугують, без перебільшення, найефективнішим інструментом для клієнтів, щоб визначити доцільність звернення до того чи іншого майстра, та для опублікування свого власного досвіду. Підхід з залишенням відгуків по унікальному посиланню дозволить забезпечити їх правдивість та унеможливить їх підробку.

Перегляд інформації про послуги та запис на них – найважливіший функціонал системи. Послуги відображаються на окремій вкладці, яка завантажується за замовчуванням, коли клієнт відкриває сторінку майстра. Послуги представлені списком, де кожен елемент списку містить наступну інформацію:

1. Фото.
2. Назва послуги.
3. Вартість.
4. Тривалість.

Рис. 2.4. Дизайн-макет відображення наявних послуг.

Також поряд з кожною послугою міститься кнопка “Запис”, натиснувши на яку відображається додаткове вікно поверх поточної сторінки, де безпосередньо

розпочинається процес запису на послугу. Процес запису відбувається у три кроки. На першому етапі клієнт вибирає зручну для нього дату та час. Вільні години майстра для зручності розбиті на “Ранок”, “День” та “Вечір”. Знизу відображається кнопка “Далі”, що розпочинає наступний етап запису.

Рис. 2.5. Дизайн-макет першого етапу для запису на послугу.

На другому етапі клієнт вводить свої контактні дані, опційно додає коментар до запису та натискає кнопку “Підтвердити запис”. Знизу також має знаходитись кнопка “Назад”, що повертає користувача до попереднього кроку з вибором вільної дати та часу.

Рис. 2.6. Дизайн-макет другого етапу для запису на послугу.

Після успішного запису, клієнт бачить на екрані підтверджений запис з усіма деталями та кнопку для швидкого переходу до Google карт для побудови маршруту до місця знаходження спеціаліста.

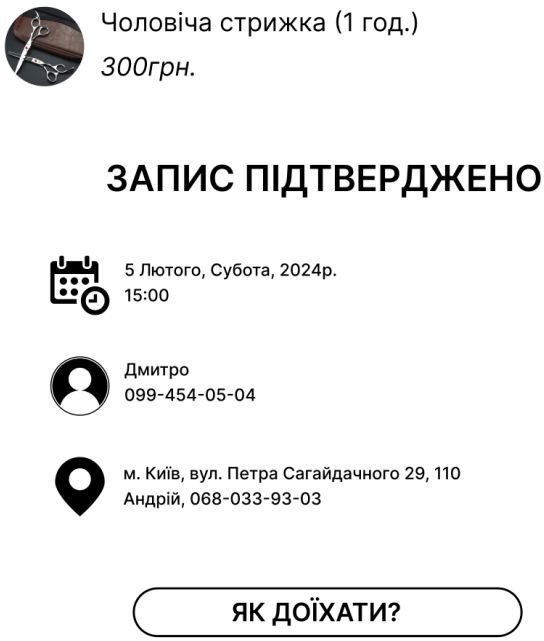

Рис 2.7. Дизайн-макет підтвердженого запису на послугу.

Висновки за розділом 2.

Отже, ми детально розібрали весь функціонал платформи та бізнес-вимоги до неї, а також детально прописали кожен варіант використання клієнтської частини з дизайн-макетами кожної сторінки. Це дає нам можливість рухатись далі та перейти до побудови структури платформи та її складових.

Підсумовуючи вимоги, можна ще раз затвердити, що ключовими складовими платформи, на яких потрібно максимально сконцентрувати увагу є:

1. Простота та зручність користувацького інтерфейсу. Підтримка основних сучасних платформ відтворення контенту: ПК, ноутбук, планшет, смартфон.
2. Швидкість підвантаження даних для забезпечення найкращого користувацького досвіду. Оскільки платформа націлена на надання клієнтам швидкого доступу до найважливішої інформації про спеціалістів, її підвантаження та відображення має займати щонайменше часу.
3. Набір безоплатного унікального функціонала для залучення найбільшої кількості малих бізнесів та самозайнятих спеціалістів, створивши таким чином першу уніфіковану платформу для підтримки малого бізнесу у сфері надання послуг в Україні.

РОЗДІЛ 3.
ПОБУДОВА СТРУКТУРИ ЦИФРОВОЇ ПЛАТФОРМИ ІНФОРМАЦІЙНОЇ ПІДТРИМКИ МАЛОГО БІЗНЕСУ У СФЕРІ НАДАННЯ ПОСЛУГ.

3.1. Структура моделей бази даних.

Визначившись з функціоналом та послідовністю дій користувачів на веб-сторінці, можемо підібрати найбільш ефективну структуру для наших моделей даних. Перше, що треба зауважити, це те, що більшість запитів, що буде надходити на наш сервер, будуть саме операції читання, тому вони мають бути максимально швидкими та оптимізованими. Далеко не кожен клієнт, що зайде на сторінку майстра, вирішить оформити запис. А для тих, хто оформить запис, буде виконуватись досить тривіальна та нескладна операція запису. Це все підводить нас до того, що для нашої платформи найкраще буде слугувати No-SQL База Даних, оскільки потрібна максимальна оптимізація операцій читання та абсолютно не потрібен функціонал транзакцій. No-SQL База Даних дозволить нам зберігати всю інформацію про спеціалістів у денормалізованому вигляді, тим самим зменшивши час на отримання та відображення інформації. З усіх No-SQL Баз Даних, що наразі існують на ринку, найкраще себе зарекомендувала MongoDB. Вона має як потужний рушій, так і величезну кількість функціонала, що дає можливість виконання практично будь-якого рівня складності задачі. Варто зауважити й той факт, що ця БД прекрасно

горизонтально масштабується в порівнянні зі звичними SQL базами, що також дає не аби яку перевагу у випадках великого навантаження. Також вона є Open-Source проєктом, що дає можливість її використовувати без додаткової платної ліцензії, а величезна спілка розробників допоможе у вирішенні будь-яких запитань.

Першою та найголовнішою моделлю нашої системи є Specialist. Дана модель об’єднує в собі як основну публічну інформацію про спеціаліста, що буде доступною для перегляду клієнтами, так і внутрішню інформацію, що потрібна для функціонування системи та її алгоритмів. Оскільки з функціональних вимог ми пам’ятаємо, що спеціалісти можуть створювати власні розклади роботи та оперувати ними, нам потрібно зберігати цю інформацію в БД. Наша система повністю зав’язана на часові рамки, починаючи тривалістю послуг і закінчуючи робочими годинами майстра. Тож перший тип даних, що ми опишемо, буде зберігати інформацію про часовий проміжок:

```
public record struct TimeFrame(
   TimeOnly Start,
   TimeOnly End);
```

Для такого функціонала як записи нам потрібно додатково до часового проміжку зберігати дату, тож створимо додатковий тип даних, що буде інкапсулювати в собі часовий проміжок:

```
public record struct TimeSlot(
   TimeFrame TimeFrame,
   DateOnly Date);
```

Маючи готові типи для роботи з часовими проміжками, створимо тип даних, що містить інформацію про розклад спеціаліста:

```
public record struct Schedule(
   TimeFrame TimeFrame,
   DayOfWeek[] WorkingDays);
```

DayOfWeek є вбудованим типом даних, що являє собою “перелічення” днів тижня. Масив днів тижня ми зберігатимемо для того, щоб надавати спеціалістам більшу гнучкість у побудові власних графіків роботи.

Далі в нас йдуть послуги. Вони напряму прив’язані до конкретного майстра і є унікальними, тож зберігатимуться вони також разом. Опишемо тип даних, що містить інформацію про послуги:

```
public record ServiceDetails(
   string PublicPhotoKey,
   string Title,
   int Price,
   int Duration);
```

Тепер, коли в нас є всі потрібні типи даних для побудови Specialist, опишемо його:

```
public record Specialist(
   string Id,
   string Name,
   string PublicTitle,
   string PhoneNumber,
   string Email,
   string Description,
   string PublicPhotoKey,
   DateTime RegisteredAt,
   int TotalRatings,
   int RatingsSum,
   IEnumerable<string> PortfolioPhotoKeys,
   IEnumerable<ServiceDetails> Services,
   IEnumerable<Schedule> Schedules);
```

Давайте трохи детальніше пройдемось по ньому. PortfolioPhotoKeys являє собою масив із зображень, а точніше ключів зображень, по якім їх можна дістати з локального або хмарного сховища. TotalRatings та RatingsSum – поля, що потребуються для вирахування середнього значення рейтингу спеціаліста. Нагадаю, що швидкість операцій читання для нас є пріоритетом, тож задля уникнення підрахунку агрегатними функціями, ми оновлюватимемо ці поля кожен раз, коли додається новий відгук, і прораховуватимемо середнє значення кожен раз, коли на сервер приходить новий запит. Schedules – масив графіків, оскільки в різні дні тижня у спеціаліста можуть змінюватись години роботи. Наприклад, з понеділка по п’ятницю розклад з 09:00 до 17:00, а в вихідні – з 12:00 до 16:00.

Перейдемо до нашої наступної колекції, яка буде зберігати записи. Для зручності, винесемо всю потрібну нам інформацію про клієнта в окремий підтип:

```
public record CustomerDetails(
   string Name,
   string PhoneNumber,
   string? Email);
```

І тепер на основі наших дизайн-макетів та функціональних вимог опишемо сам тип, що зберігає інформацію про запис:

```
public record Reservation(
   string Id,
   string SpecialistId,
   ServiceDetails Service,
   CustomerDetails Customer,
   TimeSlot Slot,
   string? Note,
   DateTime ReservedAt);
```

Остання колекція, яка буде зберігатись в нашій базі даних – Reviews (відгуки). Знаючи функціональні вимоги до відгуків, не складно описати тип даних для них:

```
public record Review(
   string Id,
   string SpecialistId,
   string CustomerName,
   string ServiceTitle,
   int Rating,
   string? Text,
   DateTime CreatedAt);
```

Варто також зазначити, що оскільки ми будемо вичитувати дані з колекцій за допомогою властивості SpecialistId, то більш ніж доречно буде створити індекс на цю властивість. Це досить легко зробити, використовуючи пакет MongoDriver для C#:

```
var collection = db.GetCollection<Reservation>("Reservations");

collection.Indexes.CreateOne(
   new CreateIndexModel<Reservation>(
       Builders<Reservation>.IndexKeys.Descending(r => r.SpecialistId)));
```

3.2. Сучасні підходи для розгортання веб-ресурсів.

З розвитком технологій в галузі веб-розробки, створення та розгортання веб-ресурсів стає все більше оптимізованим та доступним. Одним з підходом для розгортання веб-систем є використання власних потужностей, тобто виділених серверів. Таке рішення надає можливості одноразового інвестування коштів у обладнання та подальшу експлуатацію веб-сервера з оплатою тільки послуг центру обробки даних (сервісне обслуговування, енергоресурси та Інтернет-трафік).

На платформі виділеного серверу (Dedicated Server) доцільно розгорнути систему віртуалізації, що надає уніфікацію середовищу розгортання програмного забезпечення веб-сервера та ефективне рішення резервного копіювання (Backup) всіх даних на рівні файлу сховища віртуальної машини (VPS, Virtual Private Server). Більшість систем серверної віртуалізації є комерційними або потребують кваліфікованого персоналу системних адміністраторів. Відзначимо Proxmox VE – відкриту систему віртуалізації, що підтримує технології QEMU/KVM та Linux-контейнерів LXC. Ця система швидко розгортається, має зручний веб-інтерфейс керування віртуальними машинами та дозволяє масштабувати рішення, як у межах одного серверу, так й серверної ферми.

Іншим підходом до організації серверного середовища може стати рішення на базі технології віртуалізації Docker та, у разі масштабування такого рішення, кластеру Kubernetes. Однак, цей підхід більш себе виправдовує у разі застосування методології DevOps і неперервного циклу інтеграції та доставлення веб-застосунків користувачам (CI/DI, Continuous Integration / Continuous Delivery), що фактично стає стандартом в процесах розробки веб-продуктів.

Таким чином, можна отримати на стороні серверної платформи зручні засоби розгортання операційної системи у середовищі віртуальної машини та залучити наявні засоби управління й резервного копіювання на рівні самого серверу. Проте не варто забувати те, що всі вищеперелічені варіації з використання виділених серверів є нетривіальними, та вимагають досить

поглиблених навичок та знань, а також часу для підтримки безперервного функціювання сервера. Одним з не менш важливих пунктів є й гарантування безпеки даних, що зберігаються на сервері. Невпинна поява нових вразливостей вимагає частих оновлень як операційних систем, так і встановлених програмних продуктів. Це ті фактори, що можуть вплинути на відмову від утримання власного сервера, тим більш для стартапів та малих програмних продуктів, що не вимагають такого рівня гнучкості в налаштуванні.

Простішим та легшим у підтримці є підхід з використання хмарних технологій. На сучасному ринку існують як варіанти IAAS (Infrastructure as a service), так і PAAS (Platform as a service). Обидва варіанти хмарних технологій не потребують завчасних вкладень коштів, оскільки плата за використання знімається з урахуванням витрачених хвилин на використання того чи іншого продукту. IAAS являє собою платформу, на базі якої можна побудувати власноруч будь-яку бажану інфраструктуру. Це чудовий варіант для середніх та великих по розміру компаній, які не бажають будувати або орендувати цілі датацентри, але натомість потребують великої гнучкості у налаштуванні власної інфраструктури. При використанні IAAS можна керуватись тими самими підходами, що й з виділеними серверами. Єдине, що відрізняє IAAS від виділених серверів – відсутність витрат на придбання заліза та гнучкість у збільшенні або зменшенні потрібних ресурсів.

PAAS (Platform as a service) – побудований на можливостях IAAS та є ключовою моделлю хмарних обчислень, яка надає сервіси для розробки, тестування та розгортання програмного забезпечення без необхідності управління інфраструктурою. Основними характеристиками PAAS є:

1. Середовище розробки. PAAS надає розробникам зручне середовище для створення програмного забезпечення. Це може включати інтегровані середовища розробки (IDE), мови програмування та інші інструменти, що полегшують процес створення коду.
2. Розгортання та управління інфраструктурою. Один із ключових аспектів PAAS – це автоматизоване розгортання та управління інфраструктурою.

Розробники можуть концентруватися на написанні коду, а PAAS відповідає за налаштування, масштабування та управління серверними ресурсами.

3. Бази даних та суміжні функції. PAAS зазвичай охоплює готові до використання бази даних, кеш-системи, автентифікаційні сервіси та інші складові, що полегшують розробку та розгортання додатків.
4. Масштабованість та висока доступність. Платформи PAAS забезпечують автоматичне масштабування, що дозволяє пристосовувати ресурси залежно від обсягів роботи. Це робить систему більш ефективною та гнучкою.
5. Безпека та управління доступом. PAAS зазвичай надає засоби для забезпечення безпеки додатків, включаючи контроль доступу, шифрування та інші механізми захисту.

Всі ці аспекти підтверджують, що наразі підхід з PAAS є одним з найкращих варіантів для невеликих програмних застосунків та стартапів, а також у випадках, де навантаження на систему є непрогнозованим. А завдяки моделі оплати, де кошти знімаються лише за використані хвилини та ресурси, не потребує великих завчасних інвестицій.

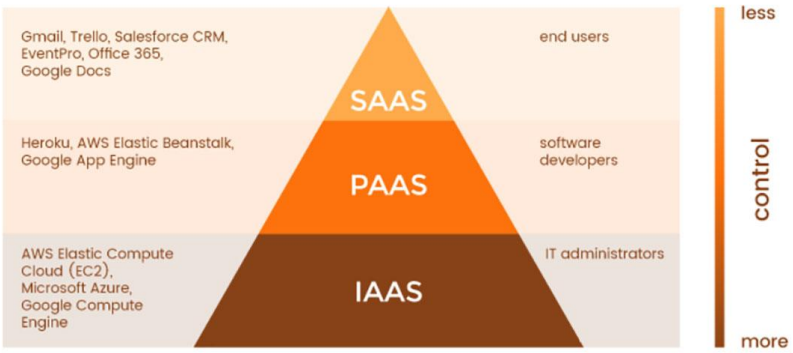

Рис. 3.1. Зони відповідальності PAAS, IAAS та SAAS.

3.3. Вибір платформи та інструментів розробки.

Існує чимало різних мов програмування та інструментів для розробки програмного забезпечення. Кожна технологія краще пасує в тих чи інших випадках, проте через їх велике різноманіття немає стандартних правил щодо вибору. Наша платформа вимагає трьох основних компонентів:

1. Клієнтська вебплатформа.
2. Вебплатформа для представників бізнесу.
3. API для зберігання та обробки даних.

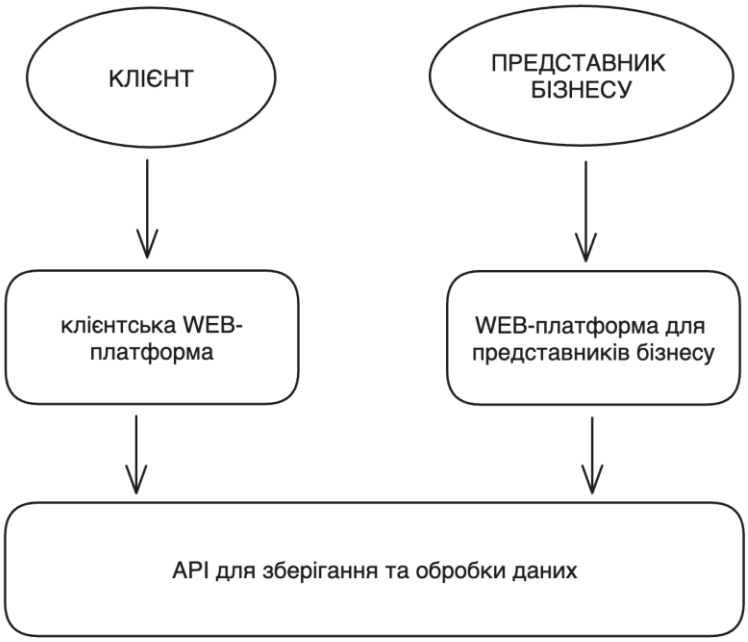

Рис. 3.1. Схема основних компонентів та взаємодії з ними.

Подібне розподілення компонентів дозволить відокремити розробку та розгортання платформ для клієнтів та представників бізнесу, що є вкрай доречним, оскільки обидві платформи мають різний функціонал, охоплення та навантаження на систему. Водночас API слід залишити об’єднаним для обох платформ через використання суміжних типів даних та логіки, а також для спрощення усієї архітектури.

Одним з найцікавіших та сучасних рішень по розробці Front-End застосунків є Angular. Це фреймворк для розробки вебдодатків, заснований на TypeScript. Його розробка почалася в компанії Google, і він став спадкоємцем AngularJS, що також був розроблений Google. Однак, Angular є повністю новим

продуктом, написаним з нуля, не маючи багатьох архітектурних обмежень свого попередника. Angular був представлений у 2010 році, а офіційний реліз відбувся у 2016 році. Початково розробка велася під кодовою назвою "Angular 2", що вказувало на значний стрибок у функціональності та архітектурі порівняно з AngularJS. Angular розвивався відкритою спільнотою розробників та командою Google. Його постійно оновлюють, додаючи нові функції та вдосконалюючи наявні. Мова програмування TypeScript, яка є суперсетом JavaScript, стала основною мовою для розробки на Angular, надаючи переваги, такі як строга типізація та покращена підтримка ООП. Angular завоював велику популярність через кілька ключових аспектів:

1. Модульність та Компонентний Підхід: Angular підтримує модульний підхід та використовує компоненти для розділення функціональності. Це полегшує розробку, тестування та підтримку коду.
2. Динамічність та Зв'язок Даних: вбудована система двостороннього зв'язку даних дозволяє легко взаємодіяти з елементами сторінки та маніпулювати даними на стороні клієнта.
3. Компіляція та Оптимізація: Angular використовує AOT (Ahead-of-Time) компіляцію, що призводить до оптимізованого та ефективного коду, покращуючи продуктивність додатка.
4. Широкий Функціонал: Angular надає багато готових функцій, таких як маршрутизація, форми, HTTP-запити та інші, що робить розробку додатків простішою та ефективнішою.
5. Активна Спільнота та Підтримка: є велика спільнота розробників Angular, що сприяє розв'язанню проблем, обміну досвідом та розвитку екосистеми фреймворку.

Визначившись з технологією для Front-End застосунків, перейдемо до частини Back-End. У рамках розробки Back-End частини для онлайн платформи було прийнято стратегічне рішення використовувати фреймворк AspNet Core WebAPI. Цей вибір обумовлений кількома ключовими аспектами, які визначають його ефективність та придатність для вирішення завдань даного

проєкту. Перш за все, це кросплатформеність даного фреймворку. Він надає можливості для розробки та запуску застосунків на будь-якій сучасній операційній системі: MacOS, Windows та Linux. Це забезпечить гнучкість та легкість у розробці, а також для подальшого розгортання системи. Оскільки фреймворк має нативну підтримку Linux, його можна легко й ефективно розгорнути як Docker контейнер, отримавши всі переваги, що надає підхід з контейнеризацією.

Наступним фактором є висока продуктивність. Технологія AspNet Core базується на мові програмування C#, яка відома своєю високою продуктивністю та великою варіативністю у підходах до написання коду, надаючи практично повний контроль над пам’яттю, що використовується програмним кодом. З використанням асинхронного програмування та оптимізованими механізмами обробки HTTP-запитів, ми маємо можливість надати користувачам ефективну та приємну взаємодію з платформою.

Останньою, та не менш важливою, причиною є активна підтримка та велика спільнота розробників. Microsoft, що є засновниками мови програмування C# та фреймворку AspNet Core, ведуть активну підтримку з щорічними оновленнями, що додають новий функціонал та покращують загальну продуктивність. Враховуючи популярність мови програмування, не дивно, що для неї існують тисячі загальнодоступних користувацьких пакетів, які додають інтерфейси для інтеграції з іншими технологіями, такими як бази даних чи інструменти хмарних сервісів. А велика спільнота розробників, що використовують C# та AspNet Core, допоможуть розібратись з будь-якими питаннями, що виникнуть під час розробки.

3.4. Архітектура програмного продукту.

Визначивши технології для розробки Fron-End та Back-End частин, а також структуру БД, створімо та проаналізуймо архітектуру усієї платформи (рис. 3.2.).

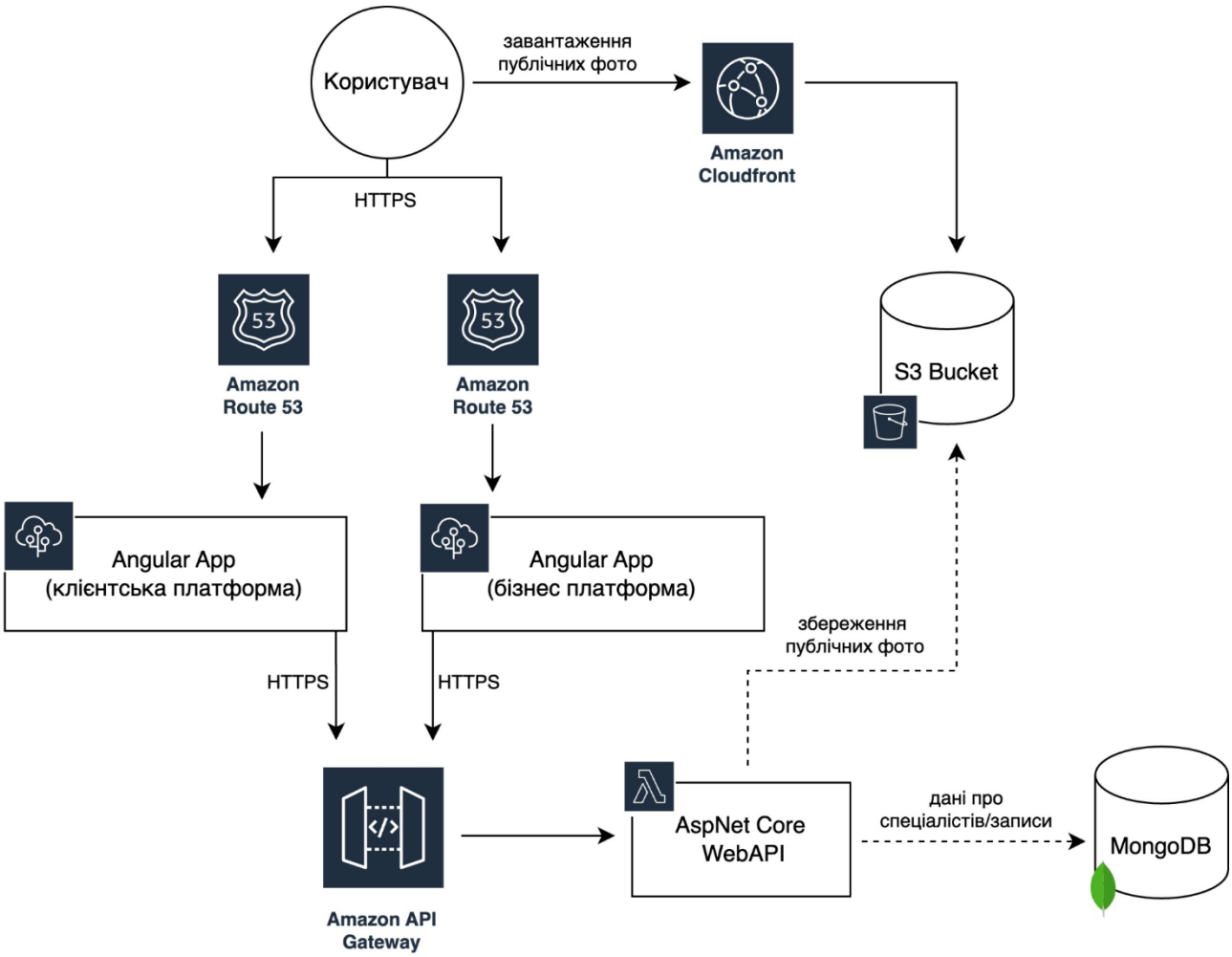

Рис. 3.2. Архітектура програмного продукту.

Angular застосунки, що слугують Front-End частиною для клієнтської та бізнес платформ, розгортуються на платформі PAAS від Amazon – AWS Elastic Beanstalk. Це рішення, що дозволяє легко розміщувати програмний код для подальшого публічного доступу через інтернет. Elastic Beanstalk бере на себе відповідальність за масштабування та балансування навантаження, а також за безперебійний доступ до застосунку за допомогою розгортування сервісів у декількох датацентрах одночасно. Доступ до Angular застосунків буде здійснюватись протоколом HTTPS через сервіс Amazon Route53 для більш практичного та зрозумілого доменного імені.

Back-End частина розміщена на платформі AWS Lambda. Це serverless рішення від Amazon, що дозволяє суттєво зменшити витрати на розміщення

застосунків, оскільки застосунок запускається окремо під кожен запит клієнтської частини. Як і у випадку з Beanstalk, AWS Lambda бере на себе усю відповідальність за масштабування та забезпечення безперервного доступу, дозволяючи максимально сконцентрувати увагу розробників на написанні коду. Доступ до Back-End буде здійснюватись за допомогою публічного AWS Gateway, що прийматиме HTTPS запити від Front-End застосунків.

Збереження даних, як і обговорювалось раніше, відбуватиметься у нереляційній базі даних – MongoDB, що розміщуватиметься на PAAS платформі MongoAtlas. Це офіційне рішення від розробників MongoDB, що дозволяє у лічені хвилини створити власний кластер та дозволити платформі дбати про безпеку, масштабованість та оновлення. Публічні фото, що використовуватимуться нашою платформою, такі як фото спеціалістів, портфоліо тощо, зберігатимуться в Amazon S3. Це ще один потужний сервіс від Amazon, призначений для зберігання необмеженої кількості даних будь-якого формату. Безпека та надійність даних гарантується вбудованим шифруванням та реплікацією у різні датацентри. З боку користувачів платформи, відображення публічних фотографій буде здійснюватись через CloudFront – CDN сервіс, що використовує дані з S3 для їх подальшого кешування та реплікації у різні регіони для найшвидшого відтворення користувачами в незалежності від їх географічного розташування.

Висновки за розділом 3.

Отже, ми визначили структуру даних, що зберігатиметься у базі даних, а також технології, на базі яких буде відбуватись реалізація застосунків. Ми також детально описали архітектуру усієї платформи, та як саме вона буде розгорнута. Завдяки вибору хмарних сервісів від Amazon ми зможемо значно зменшити першочергові інвестиції у запуск платформи, а також моментально масштабувати її залежно від навантаження. Це також значно зекономить час як на старті, так і впродовж усього циклу підтримки платформи, оскільки відповідальність за безперебійну роботу застосунків складається на плечі

постачальників хмарних сервісів, а розробникам залишається тільки фокусуватись на програмному коді.

Вибір найпопулярніших технологій та фреймворків дозволить нам побудувати швидку та сучасну платформу, де безпека гарантується передовими алгоритмами шифрування, а безперебійність – світовими лідерами у сфері хмарних технологій. Кросплатформені фреймворки стирають обмеження у виборі операційної системи, а величезне ком’юніті розробників допоможе у розв'язанні питань будь-якої складності.

РОЗДІЛ 4.
ПРОТОТИП ПРОГРАМИ.

4.1. Реалізація алгоритмів.

Основним алгоритмом нашої Back-End системи є калькуляція вільних годин для запису на послугу, тож детальніше розглянемо його. Для визначення вільних дат та годин потрібно мати наступні вхідні дані:

1. Тривалість послуги, на яку планується запис.
2. Робочі дні та години майстра.
3. Поточні записи до майстра.

Використовуючи мову програмування C# опишемо інтерфейс, що буде використовуватись для калькуляції вільних дат та годин:

```
public interface ISchedulerCalculator
{
   IEnumerable<TimeSlot> CalculateFreeSlots(IEnumerable<Schedule> schedules,
       IEnumerable<TimeSlot> reservedSlots, int minutesToReserve, int
daysToCalculate);
}
```

Маємо публічний метод, що повертатиме колекцію типів TimeSlot, що описувався у попередньому розділі. У якості вхідних параметрів ми приймаємо активні розклади майстра, колекцію зарезервованих днів та годин, тривалість для бронювання та кількість днів, для якої треба прорахувати вільні слоти. Також створимо допоміжний інтерфейс, що буде повертати поточну дату та час.

Це допоможе нам в майбутньому для покриття коду юніт-тестами:

```
public interface IDateTimeProvider
{
   DateOnly Today { get; }
   DateTime UtcNow { get; }
}
```

Для нашого алгоритму також знадобиться метод для визначення чи не перетинаються два проміжки часу, тож додамо метод розширення для типу TimeFrame:

```
public static class Extensions
{
   public static bool OverlapsWith(this TimeFrame origin, TimeFrame comparer)
   {
       return !(comparer.End <= origin.Start || origin.End <= comparer.Start);
   }
}
```

Маючи всі необхідні додаткові інтерфейси та методи, перейдемо до реалізації алгоритму калькуляції вільних дат та годин. Від наявного інтерфейсу ISchedulerCalculator створимо клас ScheduleCalculator та за допомогою ін’єкції залежностей передамо інтерфейс IDateTimeProvider в конструктор класу:

```
public class ScheduleCalculator : ISchedulerCalculator
{
   private readonly IDateTimeProvider _dateTimeProvider;

   public ScheduleCalculator(IDateTimeProvider dateTimeProvider)
   {
       _dateTimeProvider = dateTimeProvider;
   }
}
```

Тепер додамо метод CalculateFreeSlots та напишемо його реалізацію:

```
public IEnumerable<TimeSlot> CalculateFreeSlots(IEnumerable<Schedule> schedules,
   IEnumerable<TimeSlot> reservedSlots, int minutesToReserve, int daysToCalculate)
{
   var freeSlots = new List<TimeSlot>();
   var date = _dateTimeProvider.Today;
}
```

На початку метода виділяємо пам’ять під колекцію із вільними слотами, що буде вертатись в кінці. Оскільки ми робимо калькуляцію для конкретної кількості днів, то додамо цикл for, що буде виконуватись від 0 до daysToCalculate.

```
for (int i = 0; i < daysToCalculate; i++)
{
   var appropriateSchedule = schedules.FirstOrDefault(s =>
s.WorkingDays.Contains(date.DayOfWeek));
   if (appropriateSchedule == default)
   {
       date = date.AddDays(1);
       continue;
   }
}
```

Першою дією в циклі є отримання найпершого розкладу майстра для поточного дня тижня. Якщо такий розклад відсутній, то поточний день вважається неробочим, поточна дата збільшується на один день та цикл розпочинає наступну ітерацію.

```
var startTime = appropriateSchedule.TimeFrame.Start;
while (startTime.AddMinutes(minutesToReserve) <= appropriateSchedule.TimeFrame.End)
{
   var timeFrame = new TimeFrame(startTime, startTime.AddMinutes(minutesToReserve));
   var firstOverlap = reservedSlots.FirstOrDefault(rs => rs.Date == date
                                                         &&
rs.TimeFrame.OverlapsWith(timeFrame));
   if (firstOverlap != default)
   {
       startTime = firstOverlap.TimeFrame.End;
       continue;
   }
   freeSlots.Add(new TimeSlot(timeFrame, date));
   startTime = timeFrame.End;
}
```

Якщо розклад для поточного дня тижня існує, то ми розпочинаємо наступний цикл, що починається від часу початку робочого дня майстра й до останнього можливого часу, коли б майстер зміг надати послугу. В кожній ітерації даного циклу виконується наступний порядок дій:

1. Створення часового проміжку від поточного часу до часу закінчення запису.
2. Пошук найближчого зарезервованого слота з яким перетинається наш створений часовий проміжок.

Якщо такого слота не знайдено, то ми вважаємо наш часовий проміжок вільним, тож додаємо його у нашу колекцію, виставляємо поточний час годиною закінчення запису, та починається наступна ітерація циклу. У випадку,

коли ми знайшли слот, що перетинається з нашим проміжком, виконуємо наступні дії:

1. Виставляємо поточний час годиною закінчення того проміжку, що ми знайшли.
2. Розпочинаємо наступну ітерацію циклу.

Закінчивши цикл, останньою дією є додавання одного дня до поточної дати. Завершальною дією нашого методу є повертання колекції з вільними датами та годинами для запису.

4.2. Автоматизація тестування алгоритмів.

Задля впевненості в коректній роботі алгоритмів та його стабільній роботі під час майбутніх оновлень програмного коду, використаємо фреймворк для написання автоматизованих юніт-тестів. Для мови програмування C# є два основних фреймворки для написання подібних тестів: NUnit та XUnit. Оскільки вони не мають суттєвих відмінностей та лише мають різний синтаксис, використаємо NUnit. Для цього створимо новий проєкт в середовищі розробки з типом “Unit Test Project”.

Наступним етапом є визначення сценаріїв, які будуть перевірятись. Однією з загальноприйнятих конвенцій з найменування сценаріїв є “Should_<ExpectedResult>_When_<Condition>”, де <ExpectedResult> визначає бажаний результат роботи алгоритму, а Condition – умова, за якої відбувається виконання алгоритму. Для алгоритму з пошуку вільних дат та годин створимо наступні сценарії:

1. Should_Return_1_Slot_When_Schedule_13_00_To_13_30_With_Service_30_Minutes. Бажаним результатом роботи алгоритму є знаходження 1 вільного слота, коли графік роботи майстра – з 13:00 до 13:30 та послуга для резервування займає 30 хвилин.
2. Should_Return_3_Slots_When_Schedule_13_00_To_16_00_With_Service_60_Minutes. Бажаним результатом роботи алгоритму є знаходження 3

вільних слотів, коли графік роботи майстра – з 13:00 до 16:00 та послуга для резервування займає 60 хвилин.

3. Should_Return_0_Slots_When_Schedule_13_00_To_13_30_With_Service_60_Minutes. Бажаним результатом роботи алгоритму є знаходження 0 вільних слотів, коли графік роботи майстра – з 13:00 до 13:30 та послуга для резервування займає 60 хвилин.

Оскільки при реалізації алгоритму ми використовували інтерфейс для надання поточної дати та часу, нам потрібно створити нову реалізацію, що буде використовуватись при тестуванні. Це є вкрай важливим, оскільки при виконанні тестів поточна дата та час завжди мають бути однаковими.

```
public class FakeDateProvider : IDateTimeProvider
{
   public DateOnly Today => DateOnly.Parse("2024-01-01"); // Monday
   public DateTime UtcNow => DateTime.Parse("2024-01-01 09:00:00"); // Monday
}
```

Тепер створимо метод, що буде виконувати тестування бажаного сценарію:

```
[Test]
public void
Should_Return_1_Slot_When_Schedule_13_00_To_13_30_With_Service_30_Minutes()
{
   var calculator = new ScheduleCalculator(FakeDateProvider);
   var schedules = new List<Schedule>
   {
       new (new TimeFrame(TimeOnly.Parse("13:00"), TimeOnly.Parse("13:30")),
           new[] { DayOfWeek.Monday })
   };
   var reservedSlots = Enumerable.Empty<TimeSlot>();

   var result = calculator.CalculateFreeSlots(schedules, reservedSlots, 30,
1).ToList();

   Assert.That(result.Count, Is.EqualTo(1));
   Assert.That(result[0].Date, Is.EqualTo(FakeDateProvider.Today));

   Assert.That(result[0].TimeFrame.Start, Is.EqualTo(TimeOnly.Parse("13:00")));
   Assert.That(result[0].TimeFrame.End, Is.EqualTo(TimeOnly.Parse("13:30")));
}
```

На початку метода ми конструюємо бажану умову для виконання алгоритму, а точніше – графік роботи майстра. Оскільки наш сценарій не вимагає того, щоб

були присутні зарезервовані слоти, ми виділяємо під них порожню колекцію. Виконуємо метод CalculateFreeSlots з параметрами, що описали вище, та за допомогою конструкції Assert перевіряємо бажаний результат. Першою перевіркою є звірення кількості елементів колекції, що повернув метод. Трьома ж наступними перевірками ми запевняємось, що вільним проміжком для бронювання є саме той, що нам потрібен.

По аналогічній схемі ми додаємо всі наступні сценарії для перевірки та запускаємо їх за допомогою вбудованого функціонала середовища розробки. Результат представлений у вигляді окремого віконця зі списком всіх сценаріїв та зеленим маркуванням у випадку успішного його проходження, та червоним хрестом, якщо результат виявився незадовільним (рис. 4.1.).

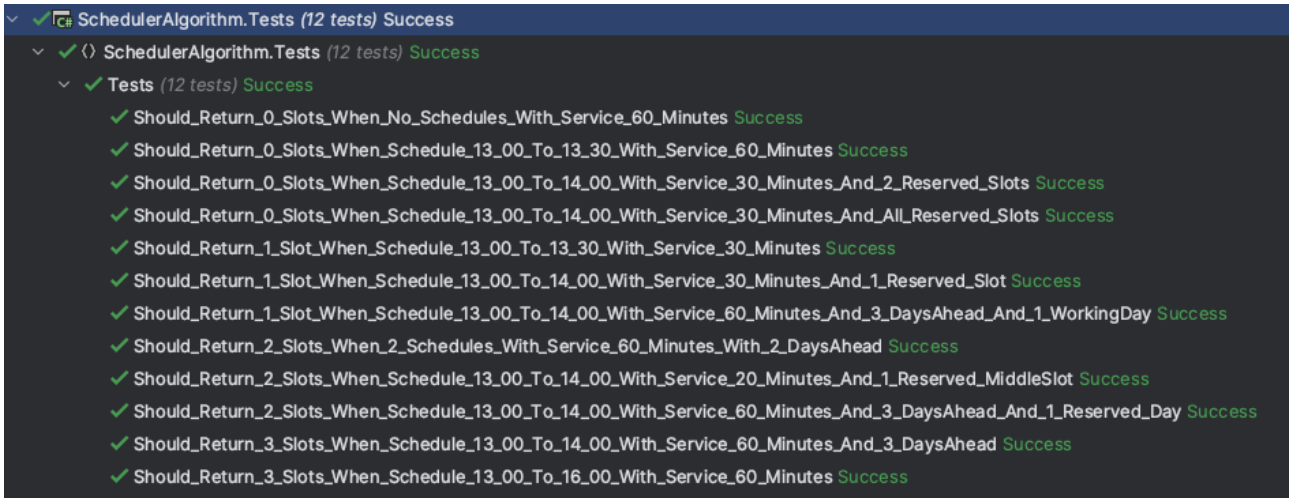

Рис. 4.1. Результат виконання автоматичних тестів.

Висновки за розділом 4.

На даному етапі ми маємо функціонуючий алгоритм для знаходження вільних дат та годин до майстра на послугу, а також детально описану структуру як даних, що зберігатимуться у БД, так і всього програмного продукту. Це відкриває нам можливості до подальшого розширення платформи та додавання графічного інтерфейсу, а також введення системи в експлуатацію. За допомогою написаних автоматичних юніт-тестів ми значно зменшили

вірогідність того, що при наступних оновленнях програмного коду виконання алгоритму зміниться.

ВИСНОВКИ

У ході виконання роботи був детально проаналізований поточний стан процесів у сфері надання послуг малим бізнесом та самозайнятими спеціалістами. Детально вивчивши це питання, було визначено проблематику та запропоновано створити сучасну онлайн платформу для підтримки подібних бізнесів. Запропоноване рішення надихалось наявними західними застосунками та було допрацьоване під український ринок та його особливості. Платформа для підтримки малого бізнесу у сфері надання послуг націлена на оптимізацію часу власників бізнесу та їх спеціалістів, а також надання їх клієнтам уніфікованого рішення для створення онлайн-записів на послуги, перегляд інформації про них, та роботи з відгуками.

Під час створення структури майбутньої платформи було використано найсучасніші підходи зі створення програмного забезпечення, включно з мовами програмування, базами даних та фреймворками для побудови графічного інтерфейсу. Для розгортання застосунку було прийняте рішення використовувати хмарні технології від однієї з передових технологічних компаній – Amazon. Використання serverless архітектури та хмарних сервісів загалом дозволить уникнути першочергових інвестицій та значно зменшити ціну за обслуговування, а також миттєво масштабувати платформу на основі навантаження в реальному часі.

Детально визначивши всі складові програмного продукту, було реалізовано найважливіший алгоритм, на якому зав’язаний основний функціонал системи – пошук вільних дат та годин для запису на послуги до спеціаліста. Алгоритм базується на графіках роботи майстра, поточних записах до нього, а також тривалості послуги, на яку відбувається запис. Завершальним етапом була реалізація автоматичних юніт-тестів до алгоритму для

забезпечення його належного функціювання та зменшенню ризиків його деградації під час майбутніх оновлень програмного коду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Цифрова платформа Atlegio – https://alteg.io/uk/info/about
2. Українська система Bookon – https://bookon.binotel.ua/
3. Платформа для ресторанного бізнесу Expirenza – https://expz.monobank.ua/
4. Американська платформа Vagaro – https://www.vagaro.com/en-gb/pro/about-us
5. Європейська платформа Fresha – https://www.fresha.com/uk/for-business
6. Fitzsimmons, J. A., & Fitzsimmons, M. J. (2013). Service Management: Operations, Strategy, Information Technology. McGraw-Hill Education.
7. Normann, R. (2001). Reframing Business: When the Map Changes the Landscape. Wiley.
8. Ari Lerner, Felipe Coury, and Nate Murray (2017). Ng-book: The Complete Book on Angular.
9. Gion Kunz (2016). Mastering Angular 2 Components.
10. Joseph Albahari (2020). C# 9.0 in a Nutshell: The Definitive Reference.
11. Valerio De Sanctis (2020). ASP.NET Core 5 and Angular: A practical guide to the most powerful full-stack web development framework.
12. James Chambers, David Paquette, Simon Timms (2017). ASP.NET Core Application Development: Building an application in four sprints.
13. Peter Sbarski (2017). Serverless Architectures on AWS: With examples using AWS Lambda.
14. Naoya Hashimoto (2015). Amazon S3 Cookbook.
15. Amazon Web Services Documentation (2022). Amazon CloudFront: Developer Guide.
16. Shannon Bradshaw, Eoin Brazil, Kristina Chodorow (2019). MongoDB: The Definitive Guide.

17. Aurobindo Sarkar (2018). Learning AWS: Design, build, and deploy responsive applications with AWS CloudFront, DynamoDB, Lambda, S3, and Cloudwatch.
18. Kyle Banker, Peter Bakkum, Shaun Verch, Douglas Garrett (2016). MongoDB in Action.