ЗВІТ З ПЕРЕВІРКИ НА ПЛАГІАТ

ЦЕЙ ЗВІТ ЗАСВІДЧУЄ, ЩО ПРИКРПЛЕНА РОБОТА

Грицаєнко М.О. ІПЗ-111м

БУЛА ПЕРЕВІРЕНА СЕРВІСОМ ДЛЯ ЗАПОБІГАННЯ ПЛАГІАТУ MY.PLAG.COM.UA І МАЄ:

СХОЖІСТЬ

9%

РИЗИК ПЛАГІАТУ

52%

ПЕРЕФРАЗУВАННЯ

2%

НЕПРАВИЛЬНІ ЦИТУВАННЯ

0%

Назва файлу: Грицаєнко М.О. ІПЗ-111м.docx

Файл перевірено: 2023-06-24

Звіт створено: 2023-06-24

МІНІСТЕРСТВО ОСВІТИ НАУКИ УКРАЇНИ

ПрАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ» (essuir.sumdu.edu.ua)

Кафедра інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА

Зав. кафедри _____

д.е.н., доц. С.І. Левицький

МАГІСТЕРСЬКА ДИПЛОМНА РОБОТА

ПРОГРАМУВАННЯ АНАЛІТИЧНОЇ СИСТЕМИ МЕДИЧНОГО ЗАКЛАДУ

Виконав

ст. гр. ІПЗ – 111м _____ М.О. Грицаєнко

Науковий керівник

Доцент _____ О. В. Шляга

Запоріжжя

2023 р.

ПРАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ

ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра інформаційних технологій

ЗАТВЕРДЖУЮ

Зав. кафедрою

(essuir.sumdu.edu.ua)

д.е.н., доцент

Левицький С.І.

_____

03.10.2022 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ ДИПЛОМНУ РОБОТУ

Студенту гр. ІПЗ – 111м, спеціальності 121 «Інженерія програмного забезпечення»

Грицаєнко Максим Олегович

1.Тема: Програмування аналітичної системи медичного закладу _____ затверджена наказом по інституту № 02-16 від 03.10.2022 р.

2. Термін здачі студентом закінченої роботи: (www.diva-portal.org) 12.06.2023 р.

3. Перелік питань що підлягають до розгляду:

1. проаналізувати предметну область;

2. змоделювати розробку з використанням відповідного інструментарію;

3. описати розробки з боку адміністратора та керівництво користувача програмного продукту приватної лікарні «Діасервіс»;

4. зробити аналіз розроблених результатів роботи;

5. оформити результати дослідження в єдину пояснювальну записку згідно вимог до магістерських дипломних робіт.

4. Календарний графік (77.93.36.128) магістерських дипломних робіт

№ етапу	Зміст	Терміни виконання	Готовність по графіку %, підпис керівника	Підпис керівника про повну готовність етапу, дата
1 (library.econom.zp.ua)	Формулювання теми магістерської дипломної роботи (збір практичного матеріалу за темою магістерської дипломної роботи)	20.10.22		
2	І атестація І розділ магістерської дипломної роботи	27.10.22		
3 (library.econom.zp.ua)	ІІ атестація ІІ розділ магістерської дипломної роботи	17.11.22		
4	ІІІ атестація ІІІ та IV розділ магістерської дипломної роботи, висновки та рекомендації, додатки, реферат, перевірка програмою «Антиплагіат»	29.12.22		
5	Доопрацювання магістерської дипломної роботи, підготовка презентації, отримання відгуку керівника і рецензії	10.01.22		
6	Попередній захист магістерської дипломної роботи	12.01.22		
7	Подача магістерської дипломної роботи на кафедру	за 3 дні до захисту		
8	Захист магістерської дипломної роботи	18.01.22		

Дата видачі завдання: 03.10.2022 р.

Керівник бамагістерської дипломної роботи _____ О. В. Шляга

Завдання отримав до виконання _____ М.О. Грицаєнко

РЕФЕРАТ

Магістерська дипломна робота містить (77.93.36.128) 62 сторінки, 19 рисунків, 12 бібліографічних посилань.

Метою роботи є розробка додатку для аналітичного супроводження роботи медичного закладу.

Відповідно до мети поставлені і вирішені такі завдання:

– здійснити аналітичний огляд за темою дослідження;

– зробити обґрунтований вибір інструментів для розробки додатку аналітичного супроводу сервісу приватної лікарні;

– реалізувати розробку з боку адміністратора та керівництво користувача програмного продукту приватної лікарні «Діасервіс».

Об’єкт дослідження – процеси і технології управління медичним закладом.

Предмет дослідження – моделі, методи та інструменти програмування веб-додатків для закладів медичної сфери.

TYPESCRIPT, ANGULAR, NEST.JS, АНАЛІТИЧНА СИСТЕМА, МЕДИЧНИЙ ЗАКЛАД

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Слово / словосполучення	Скорочення
Автоматизована система	АС
База даних	БД
Система управління базами даних	СУБД
Application Programming Interface (openarchive.nure.ua)	API
Dependency Injection	DI
Backend-a-as-Service	BaaS
Entity Framework	EF
API, спроектоване так, що його використання є лише ланцюжком викликів потрібних методів	Fluent API
JavaScript	JS
JavaScript (англ. JavaScript Object Notation) (openarchive.nure.ua)	JSON (www.sandoba.com)
Система доступу до пов’язаних між собою документів на різних комп’ютерах, підключених до Інтернету (znayshov.com)	WEB
Language-Integrated Query	LINQ
HyperText Transfer Protocol	HTTP
Integrated Development Environment	IDE
Single Page Application	SPA
Software Development Kit	SDK
Material Design	MD
Node Package Manager	npm (www.sandoba.com)
Cascading Style Sheets	CSS
Sass CSS	SCSS
Без SQL	NoSQL

Structured Query Language	SQL

ВСТУП

Більшість сучасних компаній переходить в інтернет простір, а їх системи стають все більшими. Тому уніфікація додатків для роботи на будь-яких платформах стає все більш необхідною. Саме зручне рішення для уніфікації – це веб-додаток. Сучасні браузери підтримують абсолютну більшість операційних систем та платформ, тому співробітники матимуть змогу отримати доступ до системи навіть з дому [1].

Актуальність розробки тільки зросла – мільйони співробітників різних компаній в світі булі змушені працювати з дому. Застаріла інфраструктура, яка не була розрахована на такий режим роботи, просто не змогла забезпечити робочий процес.

Перехід на універсальні веб-додатки є логічнім рішенням для багатьох компаній, які працюють з набором звичайних даних.

Основною ідеєю даного додатку є абсолютна доступність та синхронність роботи всіх актуальних сесій та проходження системи платежів в приватних лікарнях на прикладі багато діагностичній лікарні «Діасервіс». Завдяки розробленим технологіям, будь-які зміни в базі даних будуть передані в актуальні сесії без їх оновлення, що забезпечую чудову синхронність роботи всіх задіяних до оплати послуг клієнтів в актуальних сесіях.

Задачі розробленого проєкта:

1) Зібрати інформацію про подібні системи та сформувати вимоги до додатку;

2) Вибрати технології для розробки веб-додатку з доступних;

3) Сформувати правила взаємодії з BaaS сервісом;

4) Розробити дизайн додатку;

5) Розробити основний функціонал додатку

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Значення приватної лікарні

Приватні лікарні з кожним роком стають все більш популярними в Україні. І хоча медична реформа надала ширші права державним лікарням, це не дивно, враховуючи те, що вона внесла хаос і потрясіння у роботу національної системи охорони здоров'я.

Ще до медичної реформи частина пацієнтів лікувалася у приватних сімейних лікарнях. Реформа дала пацієнтам право обирати сімейного лікаря незалежно від місця проживання. Багато приватних сімейних клінік мають договори з НСЗУ. Оскільки все більше пацієнтів мають доступ до зручного медичного обслуговування для себе та своїх дітей та онуків, вони все частіше звертаються до приватних систем охорони здоров'я.

«ДІАСЕРВІС» – приватна лікарня, яка пропонує комфортні та якісні медичні послуги для всієї родини. Відділення загальної практики оснащене найсучаснішим медичним обладнанням, тому пацієнт може отримати результати в найкоротші терміни та за доступною ціною. У «ДІАСЕРВІС» уважно ставляться до своїх пацієнтів і працюють лише лікарі з великим професійним досвідом. Записатися на прийом можна через форму заявки на сайті або зателефонувавши за номером телефону, вказаним у розділі контакти.

Останнім часом відділення сімейної медицини в центрі уваги, адже тут надається повний спектр медичних послуг – від прийому педіатрів та лікарів загальної практики до оглядових кабінетів та ультразвукової діагностики. Клініки такого типу - надійні помічники для благополуччя і здоров'я всієї родини. Усі члени родини, від малого до великого, можуть знайти домашню клініку, яка допоможе з профілактикою та лікуванням різноманітних захворювань і розладів. Батьки з дітьми різного віку, підлітки та літні бабусі та дідусі.

Крім того, всі приватні клініки мають ряд загальних переваг. Основними перевагами звернення в приватну клініку є:

1. Комфортні умови обслуговування потребують постійного фінансування. Приватні медичні центри не чекають роками, поки держава профінансує ремонт і не дасть руйнуватися, а лікарі не сидять у холодних кабінетах;

2. Новіші пристрої та обладнання доступніше для приватних лікарень, оскільки вони керують власним бюджетом і не чекають «розподілу» бюджетів, які не розподіляються належним чином щороку. Лікуючому лікарю державної поліклініки часто доводиться «питати», які кошти потрібні для підтримки нормальної роботи обладнання;

3. Відсутність черг через прийом бронювання. Крім того, кожному лікарю надається достатньо часу для проведення опитування пацієнтів, оглядів та обстежень, призначення відповідного лікування та, за необхідності, додаткової діагностики;

4. Привітне обслуговування, яке починається на реєстратурі і закінчується лікарем. Здається, майже кожен здригається від думки про грубу стійку реєстрації, втрату медичних карток і результатів аналізів і чекання в черзі, щоб знайти потрібну інформацію або записатися на прийом;

5. У приватному середовищі - здоров’я завжди працюватимуть кваліфіковані кадри. Жоден лікар не буде жити на маленьку зарплату і буде бажати професійно зростати;

6. Прозорі ціни на медичні послуги можна дізнатися за прайсом який розміщений в откритому доступі через веб-ресурси , або на ресепшені у адміністратора. Пацієнти можуть заздалегідь розрахувати вартість тих чи інших медичних процедур, від простих лабораторних досліджень до операції. Після медичної реформи з’явилося багато приватних лікарень. Лікар який лікує уклав договір із МОЗ. Наразі держава оплачує основні медичні витрати пацієнтів, яким лікарі поставили діагноз у цих приватних лікарнях.

Особливою популярністю серед дорослих користуються дитяча ЛОР медицина та акушерство та гінекологія. Після педіатрії дитяча

отоларингологія є однією з найпопулярніших спеціальностей, так як діти часто застуджуються як ускладнення у вигляді отоларингології. ЛОР-лікарі в приватних клініках можуть ближче оглянути дитину за допомогою сучасного обладнання і провести такі медичні маніпуляції, як процедура «кукушка».

Дорослим жінкам важливо звернутися до гінеколога. Цей напрямок медицини поки що частково протипоказано, а відвідування гінеколога – неприємна процедура. Кожна жінка хоч раз у житті стикається з жіночими консультаціями, недбалими і болісними обстеженнями, знеціненням проблем, порушенням етичних меж і зневажливим ставленням, починаючи від улюбленої фрази багатьох лікарів «народжуй і все пройде» незалежно від діагнозу.

Все це сталося завдяки лікарям «старої гарту». Однак сучасне покоління жінок прагне стежити за своїм здоров'ям і не вважає ганебним статеве життя і відвідування гінеколога. А турботлива мама хоче, щоб її дочку водив кваліфікований і чуйний гінеколог, щоб у майбутньому дівчинка не боялася самостійно йти до лікаря.

1.2 Заявки для державних лікарень

Державні клініки є повністю державними і фінансуються повністю з бюджету. Тому такі лікарні повинні працювати суворо за положеннями, затвердженими державним органом МОЗ України, а чиновники часто занадто далекі від реалій того, що відбувається в державних медичних закладах. Державна медицина отримала в цьому плані ряд недоліків.

Основні недоліки національної системи охорони здоров'я:

1. Відсутність кваліфікованих кадрів, вихід у приватний сектор охорони здоров'я;
2. Умовна «безкоштовність» охорони здоров'я та непрозорість цінової політики зумовлені недофінансуванням галузі охорони здоров'я в міському та державному бюджетах;

3. Відсутність координації в складних медичних процедурах, де завданням здачі загального аналізу крові або зняття ЕКГ стає повне обстеження пацієнта;

4. Хамство, грубість та непривітність, які досі супроводжують кожен візит до державної лікарні;

5. Відсутність сучасного медичного обладнання та проведення базових щорічних ремонтів через брак фінансування державних лікарень.

Із запровадженням медичної реформи державні лікарні втратили монополію на видачу офіційних медичних документів, таких як звіти з дитячого садка та школи, лікарняні листи та довідки про одужання. Таким чином, відвідування державних медичних пунктів стало абсолютно необов'язковим.

Єдиний пом'якшувальний фактор на користь поліклініки - фінансова сторона справи. Проте все більше пацієнтів розуміють, що тарифи на обслуговування в приватних сімейних клініках приблизно такі ж, як і в неофіційних, а в державних клініках більше «подяк» і гарантій. Тому з кожним роком все більше пацієнтів обирають послуги приватних клінік.

Цифровізація здоров'я. Масові паперові картки пацієнтів і черги до лікаря вже в минулому. Не тільки в усьому світі, а й в Україні цифрові рішення впроваджуються поступово і безповоротно , підвищуючи ефективність роботи систем охорони здоров'я в рази. Технології швидко розвиваються та впливають на наше життя, зокрема на те, як ми збираємо та зберігаємо інформацію про здоров'я. Згідно з повідомленням GDMI, світовий ринок цифрової медицини оцінювався в 152,5 мільярда доларів США в 2020 році, а до 2026 року, за прогнозами, досягне 456,9 мільярда доларів США.

У новій цифровій реальності системи охорони здоров'я та пацієнти мають нові потреби. Що змінилося за минулий рік і які тенденції зараз змінюють охорону здоров'я в усьому світі?

Сучасні пацієнти більш пильні і уважні. При зверненні до лікаря, крім суб'єктивних скарг на здоров'я, він вже може надати дані, які містять ознаки

фізіологічної активності протягом певного періоду часу. Портативні пристрої та мобільні програми дозволяють людині стежити за пульсом, рівнем глюкози, вагою тощо. Особливо потребують люди з «груп ризику», наприклад, люди з діабетом, астмою або високим кров'яним тиском.

Самоконтроль за станом свого здоров'я – один із напрямів, що швидко розвивається. Pew Research Інтернет Дослідження показують, що 7 з 10 дорослих американців відстежують принаймні один показник здоров'я. 60% відстежують вагу, дієту чи фізичні вправи, а 33% відстежують симптоми здоров'я (артеріальний тиск, рівень глюкози, головні болі). Covid 19 загострила цю звичку через зростання популярності переносних медичних пристроїв для самодіагностики протягом останнього року на тлі страху для здоров'я.

Аналітики прогнозують, що до 2026 року більше 20% медичної діагностики включатиме рішення для самоконтролю з кількісними оцінками, а ринок медичних пристроїв для самоконтролю досягне 265,4 мільярда доларів.

Необхідність соціального дистанціювання привнесла нові риси в повсякденне життя. Багато проблем люди навчилися вирішувати не виходячи з дому. У тому числі проблеми зі здоров'ям. Ці умови стали рушійною силою розвитку телемедицини, яка дозволяє спеціалістам звертатися за медичною допомогою онлайн.

Згідно зі звітом Deloitte, станом на квітень 2020 року 43,5% усіх послуг первинної медичної допомоги в Сполучених Штатах надавалися за допомогою телемедицини. Це відкриває нові можливості, але головною перевагою є доступність. Телемедицина дозволяє отримати допомогу лікаря з будь-якої точки світу, де є підключення до Інтернету.

Це також прискорює процес отримання допомоги, заощаджуючи час як пацієнтів, так і лікарів. У майбутньому планується розробка онлайн-консультаційної кімнати для загальних законодавців . Вже сьогодні індустрія,

яка розробляє спеціалізовані бокси, пристрої та програмне забезпечення для надання високоякісної телемедицини , швидко розвивається.

Цікаво спостерігати, як технології інтегруються в медицину в розваги. Практичні застосування віртуальної реальності та доповненої реальності в галузі медицини дуже різноманітні. Наприклад, VR може допомогти в навчанні хірургів і забезпечити більш комфортні процедури. Ця методика також використовується для лікування різних психічних розладів, таких як посттравматичний стресовий розлад і тривога. Oxford VR повідомляє, що цей тип лікування може зменшити фобії та страхи на 68%.

У підтверджених звітах ринку, очікується величезне зростання медичного ринку VR. У 2019 році він становив лише 2 мільярди доларів, але очікується, що до 2027 року він досягне 34 мільярдів доларів.

Цифровізація охорони здоров'я в Україні. Країна намагається не відставати від світових тенденцій. Наприклад, активно розвиваються сервіси запису до лікаря онлайн і замовлення ліків додому. Ми маємо великий потенціал для цифровізації охорони здоров'я. З одного боку, процес цифровізації держустанов у нашій країні йде стрімко, але не завжди послідовно. Україна стала першою країною у світі, де електронні паспорти мають однакову юридичну силу з фізичними паспортами .

У найближчі кілька років ми будемо повністю безпаперові в усіх державних службах . Зміни відбуваються і в охороні здоров'я. Запроваджено електронні картотеки пацієнтів та лікарень, розроблено проекти електронних рецептів. може записатися на прийом до лікаря онлайн, так як часто така можливість пропонується тільки в приватних клініках.

Економіка в Україні швидше реагує на нові виклики часу. Страхові компанії, наприклад, активно цифровізації в охороні здоров'я і часто є рушійною силою багатьох інновацій. У 2018 році наша компанія запустила мобільний додаток, який дозволяє записуватися на прийом до лікаря та зберігати дані про ваше лікування, такі як результати аналізів та історія відвідувань, на вашому смартфоні. Страхові компанії були першими та

найважливішими партнерами багатьох стартапів у сфері охорони здоров'я. Ricky24, телемедичний проект тощо.

Незабаром Україна різко підвищить рівень національної цифровізації галузі охорони здоров'я. Це підвищить ефективність усієї системи, залучить нові інвестиції та створить найбільш сприятливий ґрунт для розвитку регіональних та глобальних проектів електронного здоров'я.

1.3 Фактори та причини впровадження цифрової охорони здоров'я

Найбільшим каменем спотикання на шляху розвитку цифрової охорони здоров'я є консерватизм і недовіра медичного світу до нових технологій. Старші покоління лікарів, як правило, скептично ставляться до інновацій і віддають перевагу традиційним методам роботи в лікарнях і поводженню з пацієнтами. Друга причина – проблема інтеграції програмного забезпечення від різних виробників і неспроможність країни виступити організатором цих інтеграційних процесів. Ще одним каменем спотикання є те, що державні клініки мають обмежений бюджет і не можуть гарантувати придбання сучасного обладнання та програмного забезпечення.

Водночас перехід на цифрову охорону здоров'я відкриває перед суспільством нові горизонти. Новим трендом стане профілактика захворювань і підтримка здоров'я на належному рівні. І країни, які можуть запровадити цифрову охорону здоров'я, мають можливість бути економічними, медичними та соціальними лідерами. Регуляторні аспекти діяльності приватних клінік в Україні. Від чого залежить розвиток медицини в (www.apteka.ua) країні? Перш за все, на це впливає культурний рівень суспільства (культура існування, спілкування, харчування, поведінка). Саме вона формує ставлення громадян до охорони здоров'я. Тому (www.apteka.ua) держава має приділяти найбільшу увагу формуванню та розвитку соціальної культури, зокрема культури споживання лікарських засобів, ставлення до свого фізичного, психічного та духовного здоров'я.

Реформа в медицині – це насамперед зміна звичайного грошового потоку.

Якщо в медичних закладах працюватимуть освічені та розумні спеціалісти, а держава створюватиме умови, за яких цінуватимуться знання та праця, то з часом в Україні автоматично з'явиться когорта гідних своєї професії лікарів. Це завдання дуже складне. Тому що сьогодні процвітає модель, коли лікарів приймають на роботу з різних причин, в основному через фактори, які належать до відомої статі.

Змін вимагає і керівництво медичних закладів. За словами О.Березюка, лікарі не мають природного таланту до менеджменту. Це ще одна сфера розумової діяльності. Хороші лікарі не є ефективними менеджерами, і навпаки. Менеджмент і освіта – це два чинники, які можуть вивести медицину з безнадійного стану. Але це також потребує фінансових вкладень та зусиль держави. Сьогодні охорона здоров'я є прикладом того, як прозоро та якісно працювати у складний економічний час. І державні клініки мають наслідувати цей приклад.

Впровадження цифрової охорони здоров'я має кілька факторів та причин:

Ефективність та покращення якості: Цифрова охорона здоров'я дозволяє зберігати та обмінюватися медичною інформацією ефективніше. Електронні медичні записи, електронні рецепти та інші цифрові інструменти дозволяють швидше та точніше доступатися до потрібних даних, зменшуючи ризик помилок та забезпечуючи більш високу якість надання медичних послуг.

Зменшення витрат: Цифрова охорона здоров'я може допомогти знизити витрати на зберігання, обробку та передачу медичних даних. Використання електронних систем дозволяє замінити паперові документи та процеси ручного введення даних, що сприяє економії часу та зниженню витрат на адміністративні операції.

Покращений доступ до медичних послуг: Цифрові технології дозволяють підвищити доступність медичних послуг для пацієнтів. Вони

можуть використовувати електронні платформи для здійснення консультацій на відстані, отримання результатів обстежень та доступу до медичної інформації. Це особливо корисно для людей, які проживають у віддалених регіонах або мають обмежений доступ до медичних закладів. (reliefweb.int)

Покращення координації догляду: Цифрова охорона здоров'я сприяє поліпшенню координації догляду за пацієнтами. Різні медичні спеціалісти та заклади можуть обмінюватися даними про пацієнта, що допомагає уникнути дублювання процедур та помилок у лікуванні. Це сприяє збільшенню безпеки та забезпеченню неперервності медичного догляду.

Розвиток медичних досліджень: Цифрова охорона здоров'я надає можливості для збору та аналізу великої кількості медичних даних, що сприяє розвитку медичних досліджень та виявленню нових методів діагностики та лікування. Це може привести до вдосконалення медичних стандартів та покращення результатів лікування пацієнтів.

Загалом, впровадження цифрової охорони здоров'я сприяє покращенню ефективності, доступності та якості медичних послуг, зниженню витрат та розвитку медичних досліджень.

1.4 Стан приватної медицини в Україні

Ринок особистої гігієни переживає глибокий спад. Ні розвитку, ні інвестицій, а багато компаній продовжують платити зарплату в конвертах. Хоча 69% готові розширити або запустити нові послуги, дивно, що набагато менше (25%) стверджують, що їхнє фінансове становище значно покращилося. При прийомі на роботу наявність категорії лікаря важлива лише для 8% респондентів, що є неприпустимим.

На даний момент системного підходу до побудови особистого оздоровчого бізнесу немає. Як і 15 років тому, більшість приватних клінік відкриваються лише для задоволення амбіцій власників, а не для досягнення результату та тривалої присутності на ринку. Звичайно, бізнес повинен приносити прибуток,

але в медичному бізнесі потрібно спочатку інвестувати в якість послуг, а потім повернути вкладені гроші. Приватних лікарень полягає в тому, що їм бракує не лише систематичності, але й зрілості. Про це свідчать результати досліджень. 49% респондентів не використовують у своїй роботі протоколи Департаменту охорони здоров'я. Це означає необґрунтований ризик. Викликає сумнів, чи дійсно 58% клінік, які беруть участь у дослідженні , запровадили внутрішні стандарти якості медичної допомоги у своїй практиці. Лише 39% опитаних оцінюють якість медичної роботи за дотриманням стандартів роботи. Насправді це свідчить про незрілість ринку. У багатьох випадках відкриття медичного закладу - це пошук швидких грошей, але це ніколи не принесе результату. Як правило, це стосується малих закладів, засновниками яких є лікарі. Насправді лікарі не є підприємцями і часто не турбуються про організацію роботи, планування та просування, бо у них уже є свій маленький світ.

Ще однією проблемою для приватного фармацевтичного ринку України є питання агентських угод . Величезна залежність медичного бізнесу від оплати «голови» пацієнта. Ці стратегії чітко демонструють короткострокові плани розвитку та бажання «швидкого» фінансування. Однак медичний бізнес має продовжувати етичний розвиток у довгостроковій перспективі. Більше половини респондентів створили медичну інформаційну систему, і більшість готові використовувати Інтернет для просування своїх послуг. Тут є конфлікт. Багато хто бажає з'явитися у вибраному пацієнтом інформаційному полі, але ми не хочемо надавати цьому пацієнту базову інформацію про його хворобу та лікування. Приватний сектор: міжнародний та український досвід. Сьогодні можна сказати, що існує проблема ефективного використання ресурсів лікарні. Зокрема, це недосконалі механізми фінансування, відсутність системного контролю за якістю медичної допомоги та неефективне планування ресурсів. Вирішити проблему допоможе зміна механізмів фінансування, впровадження відповідних систем управління якістю, планування використання ресурсів.

Написання клінічних протоколів і нагляд за їх виконанням є важливою частиною клінічної діяльності. Для оптимізації формування медичного протоколу (стандарту) необхідно реалізувати три складові: перелік медичних процедур та оперативних втручань; Довідник лікарських засобів, реактивів і витратних матеріалів.

Серед проблем, що гальмують розвиток приватного стаціонару, визначено відсутність ліцензій у хірургічних підрядників на здійснення хірургічної діяльності, що створило б здорове конкурентне середовище. Лікар, який збирається його оперувати, лікар зможе легально отримувати прибуток за оптимальною податковою ставкою. Також рекомендовано дозволити наймання медичних працівників за контрактом, визначити умови оплати праці, встановити показники якості роботи, запровадити додаткові підстави для звільнення.

1.5 Висновок до першого розділу

В першому розділі звіту було розглянуто предметну область дипломної роботи – приватну медицину України, її стан та рівень цифровізації. Також були висвітлені теми державної медицини та її недоліки на фоні медицини приватної.

РОЗДІЛ 2
МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ З ВИКОРИСТАННЯМ ІНСТРУМЕНТІВ ПРИ РОЗРОБЦІ СЕРВІСУ ДЛЯ ПРИВАТНОЇ ЛІКАРНІ

2.1 Інструменти та бібліотеки фронт-частини додатку

Даний додаток створено на мові програмування TS, яка компілюється в JS, за допомогою JS-фреймворку Angular та ui-фреймворку Material Design з використанням пакетного менеджеру npm. Для зручного написання стилів використовується Sass, а саме його версія з фігурними дужками - SCSS.

2.1.1 JavaScript

JavaScript (рис. 2.1) – мультипарадигменна мова програмування. Підтримує та використовує ООП за всіма принципами та правилами програмування специфікації ECMAScript (стандарт ECMA-262).

Рисунок 2.1 – Інтерфейс прикладу використання JavaScript

JavaScript використовується для різноманітних цілей у веб-розробці та програмуванні загалом. Основні використання JavaScript включають:

Веб-розробка: JavaScript використовується для створення динамічних веб-сторінок, взаємодії з користувачем та реалізації різноманітних функціональностей на стороні клієнта. Він дозволяє змінювати вміст сторінок, обробляти події, валідувати дані, створювати анімації та взаємодіяти з веб-сервером через AJAX.

Розробка мобільних додатків: JavaScript використовується в різних фреймворках та платформах, таких як React Native та Ionic, для створення кросплатформових мобільних додатків. За допомогою JavaScript розробники можуть створювати додатки, які працюють на різних операційних системах, таких як iOS та Android.

Розробка ігор: JavaScript використовується для створення веб-ігор, які запускаються безпосередньо в браузері. Він дозволяє контролювати графіку, анімацію, фізику, штучний інтелект та інші аспекти гри.

Розробка серверної частини: JavaScript також може бути використаний для розробки серверної частини додатків за допомогою платформи Node.js. Він надає можливість створювати серверні програми, оброблювати запити, працювати з базами даних та реалізовувати різноманітні бізнес-логіку.

Розробка розширень браузерів: JavaScript дозволяє створювати розширення для різних веб-браузерів, що розширюють їх функціональність та надають нові можливості.

Робота з даними та взаємодія з API: JavaScript дозволяє отримувати дані з серверів за допомогою AJAX-запитів та взаємодіяти з різними API, такими як соціальні мережі, геолокаційні сервіси, платіжні системи тощо.

Це лише кілька з великої кількості прикладів використання JavaScript. Завдяки своїй широкій підтримці, JavaScript став однією з найпопулярніших мов програмування для розробки веб-додатків та програмного забезпечення загалом.

Найперша реалізація JavaScript була створена Бренданом Ейхом в компанії Netscape, і з тих пір оновлюється, щоб відповідати ECMA-262 Edition 5 і пізніших версій. Цей движок називається SpiderMonkey і реалізований на мові C/C++ . Движок Rhino створений Норрісом Бойдом і реалізований на мові Java. Як і SpiderMonkey, Rhino відповідає (ir.nmu.org.ua) ECMA-262 Edition 5.

JavaScript – це є об'єктно-орієнтована мова програмування («кодингу зі скриптами» іншомовний вираз в народі), яка має ряд властивостей, властивих функціональним мовам:

1. Функції – об'єкти першого класу;
2. Функції – списки;
3. Функції – каррінг;
4. Так звані анонімні функції;
5. Замикання – додаткова гнучкість.

МП JavaScript використовується в web-додатках клієнтської частини: клієнт-серверних програм, в якому клієнтом є браузер, а сервером – (dbpedia.org) web-сервер, що мають розподілену між сервером і клієнтом логіку. Обмін інформацією в web-додатках відбувається по мережі. (eir.zntu.edu.ua) Одним з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому (dbpedia.org) web-додатки є кросплатформним сервісами.

2.1.2 Мова програмування TypeScript

TypeScript (рис. 2.2) – мова програмування, представлена Microsoft в 2012 році і позиціонується як засіб розробки web-додатків, що розширює можливості JavaScript.

Розробником мови програмування «TypeScript» є Андерс Хейлсберг, який створив раніше наступні мови програмування Turbo Pascal, Delphi та C#, саме ці мови дали великий початок в основу програмування в усьому світі.

Рисунок 2.2 – Вид логотипу мови програмування TyPe Script

TypeScript – це розширення мови ECMAScript 5. До неї додані наступні поліпшені опції:

1. Анотації типів і перевірка їх узгодження на етапі компіляції;
2. Висновок типів;
3. Клас;
4. Інтерфейс;
5. Перелічувані типи;
6. Домішка;
7. Узагальнене програмування;
8. Модулі[12];
9. Скорочений синтаксис "стрілок" для анонімних функцій;
10. Додаткові параметри та параметри за замовчуванням;
11. Кортеж.

Синтаксично, TypeScript дуже схожий на JScript.Net, чергову реалізацію Microsoft мовного стандарту ECMA-262, що забезпечує підтримку статичної типізації і класичних об’єктно-орієнтованих можливостей мови, таких як класи, спадкування, інтерфейси і простору найменуванням.

2.1.3 Node.js і npm

Node.js (рис. 2.3) є асинхронним середовищем виконання JavaScript, призначеним для створення масштабованих мережевих додатків. Нижче наведений приклад "hello world" демонструє можливість обробки багатьох з'єднань одночасно. Кожне з'єднання активує функцію зворотного виклику, але якщо немає активних з'єднань, Node.js переходить у сплячий режим.

Лістинг коду 2.2 - Приклад коду node.js

```
const http = require('http');
const hostname = '127.0.0.1';
const port = 3000;
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});
server.listen(port, hostname, () => {
console.log(`Server running at http://${hostname}:${port}/`);
(www.admecindia.co.in)
});
```

Рисунок 2.3 – Вид логотипу Node.js

Node Package Manager – це менеджер пакетів для МП JavaScrip, логотип показано на рисунку 2.4, npm.Inc є дочірнею компанією GitHub (дочірня компанія Microsoft), яка надає хостинг для розробки програмного забезпечення та контролю версій з використанням Git.

Рисунок 2.4 – Вид логотипу npm.Inc

Npm - це менеджер пакетів, який входить до складу Node.js. Протягом багатьох років Node широко використовувався розробниками JavaScript для обміну інструментами, установки різних модулів і управління їх залежностями.

Він має дві основні функції:

Він є широко використовуваним репозиторієм для публікації проєктів Node.js з відкритим вихідним кодом. Ця онлайн-платформа дозволяє будь-кому публікувати та ділитися інструментами, які були написані з використанням JavaScript і Node.js.

npm є інструментом командного рядка, який сприяє взаємодії з онлайн-платформами, такими як браузери та сервери. Він надає засоби для установки та видалення пакетів, управління версіями та управління залежностями, необхідними для запуску проекту.

Для використання npm потрібно спочатку встановити Node.js, оскільки вони мають взаємозв'язок. Командний рядок npm відповідає за правильну роботу Node.js. Щоб використовувати пакети, проєкт повинен містити файл з ім'ям package.json. Усередині цього файлу знаходяться метадані, пов'язані з проектом:

1. Назва проекту;
2. Первісна версія;
3. Опис;
4. Точка входу;
5. Тестові команди;

6. Репозиторій Git;
7. Ключові слова;
8. Ліцензія;
9. Залежності;
10. DevDependencies.

2.1.4 Фреймворк «Angular»

"Angular" є фреймворком, розробленим компанією Google для створення клієнтських додатків. Основна спрямованість цього фреймворку полягає в розробці SPA-додатків (односторінкових додатків). Angular є нащадком фреймворку AngularJS, але водночас він є новим фреймворком з принципово новим підходом. Angular надає ряд функціональностей, таких як двостороннє зв'язування, що дозволяє динамічно змінювати дані в інтерфейсі, коли дані моделі змінюються в іншому місці, шаблони, маршрутизацію і багато іншого.

Рисунок 2.5 – Вид логотипу фреймворку «Angular»

Популярність Angular обумовлена кількома факторами:

Корпоративна підтримка: Angular розробляється компанією Google, що (shorturl.at) надає йому вагомий статус та надійність. Компанії часто віддають

перевагу використанню фреймворку, який має сильну корпоративну підтримку.

Широкі можливості: Angular надає потужні інструменти для розробки клієнтських додатків, включаючи двостороннє зв'язування даних, шаблони, маршрутизацію, обробку подій і багато іншого. Це дозволяє розробникам ефективно створювати складні додатки з багатим функціоналом.

Спільнота розробників: Angular має велику та активну спільноту розробників, (shorturl.at) яка допомагає одне одному, надає підтримку і ділиться знаннями. Це створює сприятливу атмосферу для взаємодопомоги та швидкого вирішення проблем.

Масштабованість: Angular добре підходить для розробки великих проектів, оскільки він має вбудовану підтримку модульності і компонентної архітектури. Це дозволяє ефективно організовувати код і підтримувати його при подальшому розширенні.

Активне оновлення: Команда розробників Angular постійно вдосконалює фреймворк, випускаючи нові версії з поліпшеннями та новими функціями. Це дозволяє розробникам залишатися на хвилі останніх технологій і використовувати найновіші можливості.

Ці фактори разом сприяють популярності Angular серед розробників, а також забезпечують йому широке застосування у проектах різного розміру та складності.

2.1.5 Angular Material

Angular Material (рис. 2.6) – це бібліотека компонентів інтерфейсу користувача для розробників Angular. Компоненти Angular Material допомагають створювати привабливі, покрокові та функціональні web-сторінки та web-додатки, дотримуючись при цьому сучасних принципів web-дизайну, таких як переносимість браузера, незалежність від пристроїв і витончений мінімалізм.

Рисунок 2.6 – Вид логотипу Angular Material

Бібліотека дає розробнику широкий вибір стилізованих компоненти для розробки, що прискорює саму розробку та уніфікує додатки. Частина бібліотеки наведена нижче.

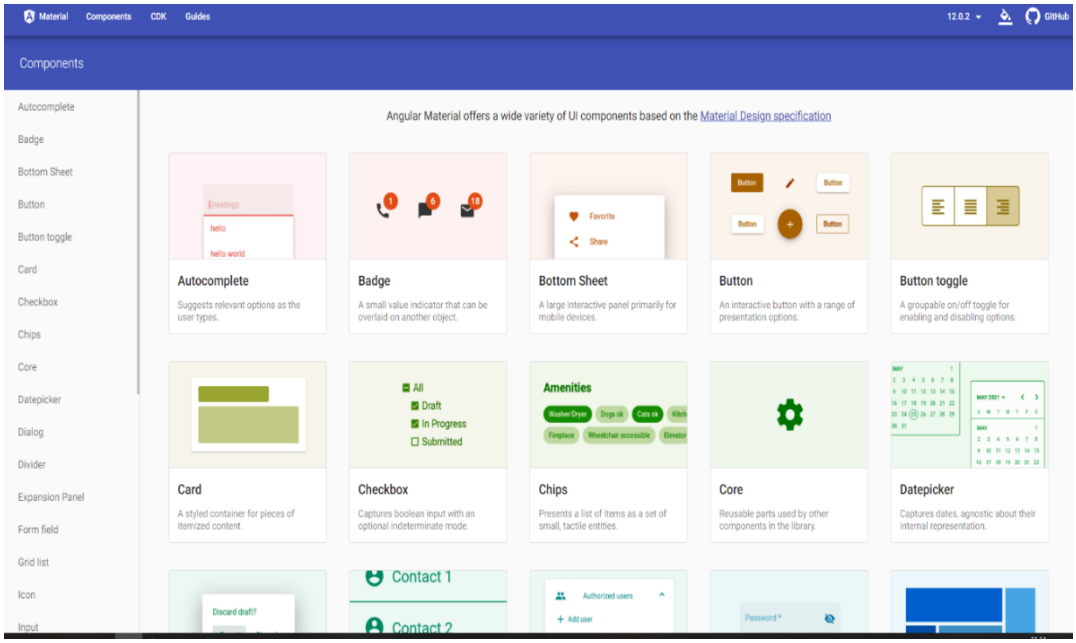

Рисунок 2.7 – Вид частини компонетів «Materials»

Material Design (Матеріальний дизайн) є стилем графічного оформлення інтерфейсів програмного забезпечення і додатків, розробленим компанією Google. Вперше представлений на конференції Google I/O 25 червня 2014 року,

(uk.wikipedia.org) цей стиль розширює концепцію «карток», яка з'явилася у Google Now, і застосовується для створення суворих макетів, анімацій, переходів, відступів та ефектів глибини (світла і тіні). Графічні дизайнери Google пропонують, щоб додатки мали заокруглені кути, а перемикання між картками було плавним та майже непомітним.

2.1.6 Sass та SCSS

Sass – syntactically awesome style sheets (синтаксично приголомшливих таблиць стилів) – це сценарії препроцесора, яка інтерпретується або компілюється в каскадні таблиці стилів. (ela.kpi.ua) Логотип графічного представлення зображено на рисунку 2.8.

Рисунок 2.8 – Вид логотипу з назвою «Sass»

Sassy CSS в будь-якому своєму прояві є «розширенням» мови CSS, а значить, все що працює в CSS, працює і в Sass / SCSS.

У Sass/SCSS є змінні, і вони відрізняються від тих що в CSS - вони починаються з двох тире (--color: #9c27b0). У SCSS змінна позначається знаком долара ($color: #9c27b0).

2.1.7 AngularFire

AngularFire згладжує нерівні краї, з якими може зіткнутися розробник Angular при впровадженні агностичного Firebase JS SDK і прагне забезпечити

більш (openarchive.nure.ua) природний досвід для розробників, відповідаючи умовам Angular.

AngularFire підтримує:

1. Залежності;
2. Обгортки Zone.js;
3. RxJS;
4. API, зручний для NgR;
5. Пасивне завантаження – AngularFire динамічно імпортує більшу частину Firebase, скорочуючи час на завантаження програми, Deploy schematics, Google Analytics, Router Guards.

2.2 Інструменті та бібліотеки серверної частини

Серверна частина представлена Node.js додатком розробленому на фреймворку Nest.js за допомогою пакетного менеджера npm.

2.2.1 Nest.js

Nest – це платформа для створення ефективних, масштабованих додатків Node.js на стороні сервера (library.econom.zp.ua) (рис. 2.9).

Рисунок 2.9 – Лого Nest.js

По синтаксису та підходу до організації коду Nest схожий та навіть ідентичний Angular – широке використання декораторів, максимальна типізація, розбивання додатку на відокремлені модулі-компоненти та використання обгорток для базових API мови JS.

Приклад сервісу з модуля роутера продуктів:

```
import { HttpException, HttpStatus, Injectable } from
'@nestjs/common';
import { FirebaseAdmin, InjectFirebaseAdmin } from 'nestjs-
firebase';
import { Product } from './product.interface';
@Injectable()
export class ProductsService {
  constructor(
    @InjectFirebaseAdmin() private readonly firebase:
FirebaseAdmin,
  ) {}
  public async getAllProducts(): Promise<Array<Product>> {
    const querySnapshot = await
this.firebase.db.collection('products').get();
    const collection = [];
    querySnapshot.forEach((doc) => {
      collection.push({ id: doc.id, ...doc.data() });
    });
    return collection;
  }
  public async getProductById(id: string) {
    const documentRef = await this.firebase.db
      .collection('rates')
      .doc(id)
      .get();
    const product = documentRef.data();
    if (product) {
      return documentRef.data();
    } else {
      throw new HttpException(
        {
          status: HttpStatus.FORBIDDEN,
          error: 'Product not found',
        },
        HttpStatus.FORBIDDEN,
      );
    }
  }
  public async createProduct(product: Partial<Product>) {
```

```
    return await
this.firebase.db.collection('products').add(product);
  }
  public async updateProduct(product: Partial<Product>) {
    const docRef =
this.firebase.db.collection('products').doc(product.id);
    delete product.id;
    return await docRef.set(product, { merge: true });
  }
  public async deleteProduct(id: string) {
    console.log(id);
    return await
this.firebase.db.collection('products').doc(id).delete();
  }
}
```

2.3 База даних Firebase

База даних (Baa сервіс) Firebase, – це хмарна база даних, яка дозволяє користувачам зберігати і отримувати інформацію, а також має зручні засоби і методи взаємодії з нею. (ela.kpi.ua)

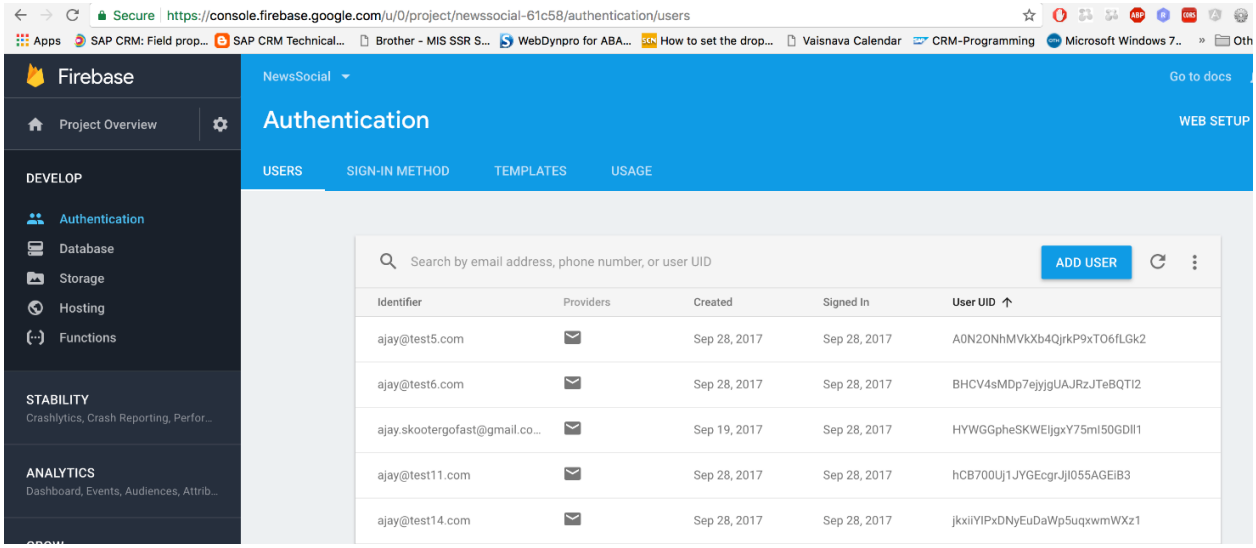

Рисунок 2.9 – Логотип Firebase

База даних реального часу Firebase – БД Firebase_Realtime дозволяє зберігати та запитувати дані користувачів, а також робить їх доступними для користувачів у режимі реального часу. БД Cloud Firestore - це гнучка,

масштабована база даних для розробки під мобільні пристроїв, ресурсу Інтернет та серверів від БД Firebase та Google_Cloud_Platform. БД Firebase Storage – дозволяє завантажувати та зберігати створений користувачем вміст, наприкладі файлу та зображення (картинка).

БД Firebase Cloud Messaging – це рішення для обміну повідомленнями між платформами, яке дозволяє надійно доставляти повідомлення безкоштовно. Аутентифікація Firebase, допомагає аутентифікувати та керувати користувачами, які отримують доступ до програми.

2.4 Платіжне API

Для оплати товару використовується інтеграція через сервіс «Google Pay». Основною причиною такого вибору стала орієнтація всього додатку на використання сервісів та продуктів Google, тому для підтримання екосистеми вибір і пав на сервіс Google Pay. Логотип даного сервісу зображено на рисунку 2.10. нашого другого розділу, він і є головною родзинкою в проєкті до магістерської роботи.

Рисунок 2.9 – Вид логотипу Google Pay

Взаємодія з API сервісу Google Pay сувого регламентована. Google навіть дає рекомендації що до зовнішнього вигляду кнопки інтеграції.

2.5 Висновок за другим розділом

Розглянуто повний інструментарій програмної розробки яка представляє практичну частину розробленого проєкту магістерської роботи.

Всі технології передбачені з прямим доступом до відкритої мережі Інтернет. IDE для розробки Front-end частини та Back-end частини програмного продукту. Технології для розробки Front-end частини програмного продукту. Красиво подано інформацію СУБД для серверу бази даних розробленого програмного продукту. Мови програмування, які будуть використані у розробці автоматизованої системи під приватну лікарню на прикладі багатопрофільної клініки «ДіаСервіс». Клініка за великі роки напрацювала дуже велику базу клієнтів та має позитивну динаміку з належного обслуговування. Тож вона повинна використовувати належні програмні продукти для швидкого інформаційного та цифрового документообігу.

За звичай кажуть так не треба економити на препаратах та клієнтах які звернулися саме до вас, а треба вразити та провести повний супровід для майбутнього повернення його знову і знов.

РОЗДІЛ 3
ОПИС РОЗРОБКИ З БОКУ АДМІНІСТРАТОРА ТА КЕРІВНИЦТВО КОРИСТУВАЧА ПРОГРАМНОГО ПРОДУКТУ ПРИВАТНОЇ ЛІКАРНІ «ДІАСЕРВІС»

3.1 Старт розробки фронт частини проекту

Кожен WEB-додаток має в собі ключовий програмний файл під назвою «package.json» він є головним. Саме тут знаходяться ключові параметри розробленого проєкту, програмний код даного файлу зображено на лістингу 3.1 даного розділу .

Лістинг коду сервісу 3.1 - Файл package.json

```
{
     "name": "it-planet",
     "version": "1.0.1",
     "scripts": {
      "ng": "ng",
     "start": "ng serve",
     "build": "ng build",
      "lint": "ng lint"},
 "private": true,
 "dependencies":{ (stackoverflow.com)
       "@angular/animations": "~10.1.5",
      "@angular/cdk": "^10.2.5",
     "@angular/common": "~10.1.5",
      "@angular/compiler": "~10.1.5",
     "@angular/core": "~10.1.5",
      "@angular/forms": "~10.1.5",
     "@angular/material": "^10.2.5",
     "@angular/platform-browser": "~10.1.5",
     "@angular/platform-browser-dynamic": "~10.1.5",
       "@angular/router": "~10.1.5",
     "@coreui/angular": "^2.9.4",
     "@coreui/coreui": "^3.3.0",
```

```
"@fortawesome/angular-fontawesome": "^0.7.0",
"@fortawesome/fontawesome-free-brands": "^5.0.13",
 "@fortawesome/fontawesome-svg-core": "^1.2.32",
"@fortawesome/free-solid-svg-icons": "^5.15.1",
"@ngrx/store": "^10.0.1",
      "firebase": "^7.24.0",
"font-awesome": "^4.7.0",
 "ngx-letter-image-avatar": "^0.1.6",
"ngx-material-file-input": "^2.1.1",
 "ngx-swiper-wrapper": "^10.0.0",
"primeicons": "^4.0.0",
 "rxjs": "~6.6.0",
"swiper": "^6.3.4",
"tslib": "^2.0.0",
 "zone.js": "~0.10.2"
},
"devDependencies": {
 "@angular-devkit/build-angular": "~0.1001.6",
"@angular/cli": "~10.1.6",
"@angular/compiler-cli": "~10.1.5",
"@angular/fire": "^6.0.3",
"@types/jasmine": "~3.5.0",
"@types/jasminewd2": "~2.0.3",
"@types/node": "^12.11.1",
"codelyzer": "^6.0.0",
"jasmine-core": "~3.6.0",
"jasmine-spec-reporter": "~5.0.0",
 "karma": "~5.0.0",
"karma-chrome-launcher": "~3.1.0",
"karma-coverage-istanbul-reporter": "~3.0.2",
"karma-jasmine": "~4.0.0",
 "karma-jasmine-html-reporter": "^1.5.0",
 "protractor": "~7.0.0",
 "ts-node": "~8.3.0",
"tslint": "~6.1.0",
 "typescript": "~4.0.2"
}}
```

Основний розділ файлу містить важливі поля, включаючи:

name – назва проекту;

version – версія проекту;

scripts – команди для виконання в консолі npm;

private – тип репозиторію;

dependencies – основні бібліотеки проекту;

devDependencies – бібліотеки для розробки.

Для початку розробки системи, що стосується теми дипломної роботи, необхідно встановити залежності за допомогою команди npm і запустити команду "ng serve" для неперервної збірки проекту.

3.2 Старт розробки серверної частини проекту

Серверна частина написана в програмному оточенні Node.js, тому тут також є пакетний файл і інсталяція залежності проходить тим самим способом. Нижче представлено вміст файлу package.json серверної частини додатку:

```
{
  "name": "nest-back",
  "version": "0.0.1",
  "description": "",
  "author": "",
  "private": true,
  "license": "UNLICENSED",
  "scripts": {
    "prebuild": "rimraf dist",
    "build": "nest build",
    "format": "prettier --write \"src/**/*.ts\" \"test/**/*.ts\"",
    "start": "nest start",
    "start:dev": "nest start --watch",
    "start:debug": "nest start --debug --watch",
    "start:prod": "node dist/main",
    "lint": "eslint \"{src,apps,libs,test}/**/*.ts\" --fix",
    "test": "jest",
    "test:watch": "jest --watch",
    "test:cov": "jest --coverage",
    "test:debug": "node --inspect-brk -r tsconfig-paths/register -r ts-node/register (github.com) node_modules/.bin/runInBand",
    "test:e2e": "jest --config ./test/jest-e2e.json"},
  "dependencies": {
    "@nestjs/common": "^8.0.0",
    "@nestjs/core": "^8.0.0",
    "@nestjs/platform-express": "^8.0.0",
    "firebase-admin": "^10.0.0",
    "firebase-nest": "^0.7.1",
    "nestjs-firebase": "^8.1.0",
    "reflect-metadata": "^0.1.13",
    "rimraf": "^3.0.2",
    "rxjs": "^7.2.0"
  },
  "devDependencies": {
```

```
    "@nestjs/cli": "^8.0.0",
    "@nestjs/schematics": "^8.0.0",
    "@nestjs/testing": "^8.0.0",
    "@types/express": "^4.17.13",
    "@types/jest": "^27.0.1",
    "@types/node": "^16.0.0",
    "@types/supertest": "^2.0.11",
    "@typescript-eslint/eslint-plugin": "^4.28.2",
    "@typescript-eslint/parser": "^4.28.2",
    "eslint": "^7.30.0",
    "eslint-config-prettier": "^8.3.0",
    "eslint-plugin-prettier": "^3.4.0",
    "jest": "^27.0.6",
    "prettier": "^2.3.2",
    "supertest": "^6.1.3",
    "ts-jest": "^27.0.3",
    "ts-loader": "^9.2.3",
    "ts-node": "^10.0.0",
    "tsconfig-paths": "^3.10.1",
    "typescript": "^4.3.5"
  },
"jest": {
"moduleFileExtensions": [
"js",
"json",
"ts"
],
"rootDir": "src",
"testRegex": ".*\\.spec\\.ts$",
"transform": {
"^.+\\.(t|j)s$": "ts-jest"
},
"collectCoverageFrom": [
"**/*.(t|j)s"
],
"coverageDirectory": "../coverage",
"testEnvironment": "node"}} (medium.com)
```

Для запуску проекту в режимі з параметром «наглядати», тобто перезбирати проект в разі зміні виконуючих файлів, треба ввести npm run start:debug.

3.3 Кабінет БД Firebase

Так, як для зберігання постійних та особистих даних про меню та користувачів додаток використовує в БД BaaS-рішення БД Firebase, то

потрібно створити та налаштувати його. Подібний кабінет зображено на рисунку 3.1.

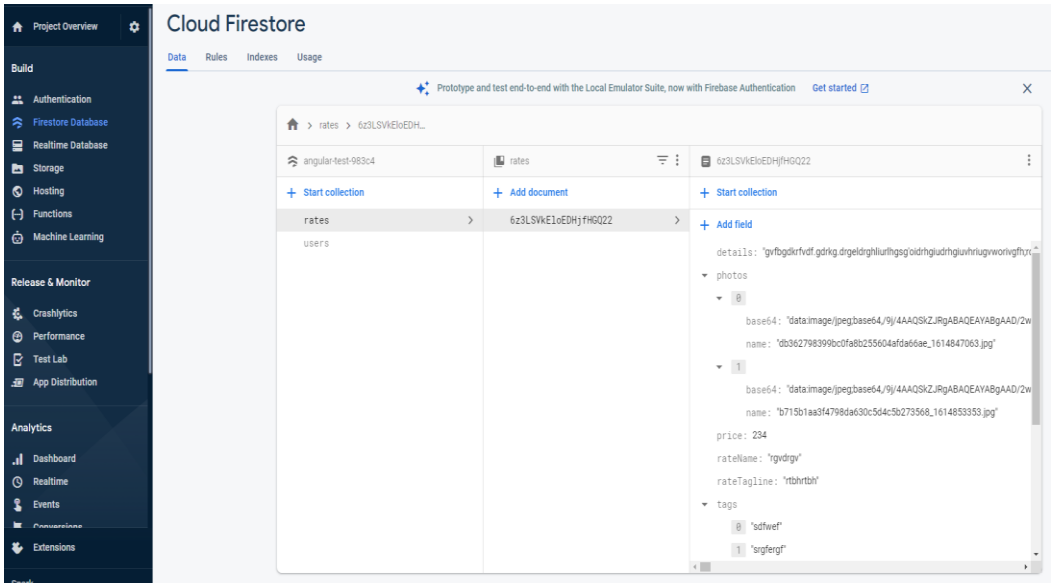

Рисунок 3.1 – Вид особистого кабінету розробника Firebase

БД Firebase - це NoSQL СУБД, яка дає широкий функціонал для розробника. Щоб підвищити безпеку бази, потрібно прописати правила бази, в яких буде описано які колекції доступні для публічного редагування, а які ні.

Лістинг коду 3.2 – Вид коду правил використання даних в БД

```
rules_version = '2';
service cloud.firestore {
match /databases/{database}/documents {
match /{document=**} {
allow read, write: if
request.time < timestamp.date(2022, 1, 1);}
match /users/{userId} {
allow write, read: if isOwner(userId);}
// Reusable function to determine document ownership
function isOwner(userId) {
return request.auth.uid == userId
}
}}
```

Rules_version – це поле, яке показує яка саме ітерація правил перед нами. Станом на листопад 2022 року доступна версія 2 правил безпеки Cloud Firestore. Версія 2 правил змінює поведінку рекурсивних символів узагальнення {name=**}.

Всі Firestore – правила безпеки складаються з match заяв, які ідентифікують документи в базі даних, і allow виразів, які керують доступом до цих документів. Кожен запит до бази даних Cloud Firestore з web-клієнта перевіряється на відповідність правилам безпеки перед читанням або записом будь-яких даних. Якщо правила забороняють доступ до будь-якого із зазначених шляхів документа, весь запит не виконується.

3.4 Структура складових проєкту

3.4.1 Структура та код (library.econom.zp.ua) фронт частини

Додатки Angular схожі по своїй структурі. Цей проєкт підтримує більшість best practices. Структура розробленого проєкту зображена на рисунку (library.econom.zp.ua) 3.2.

Додаток складається з модулів, моделей, сервісів та компонентів. Модулі вказують складальнику які файли в які бандли потрібно складати. Це дозволяє раціоналізувати структуру та не друкувати в основному модулі всі компоненти. На лістингу 3.3 зображено модуль для компонентів Materials, який дозволяє використовувати компоненти mat.

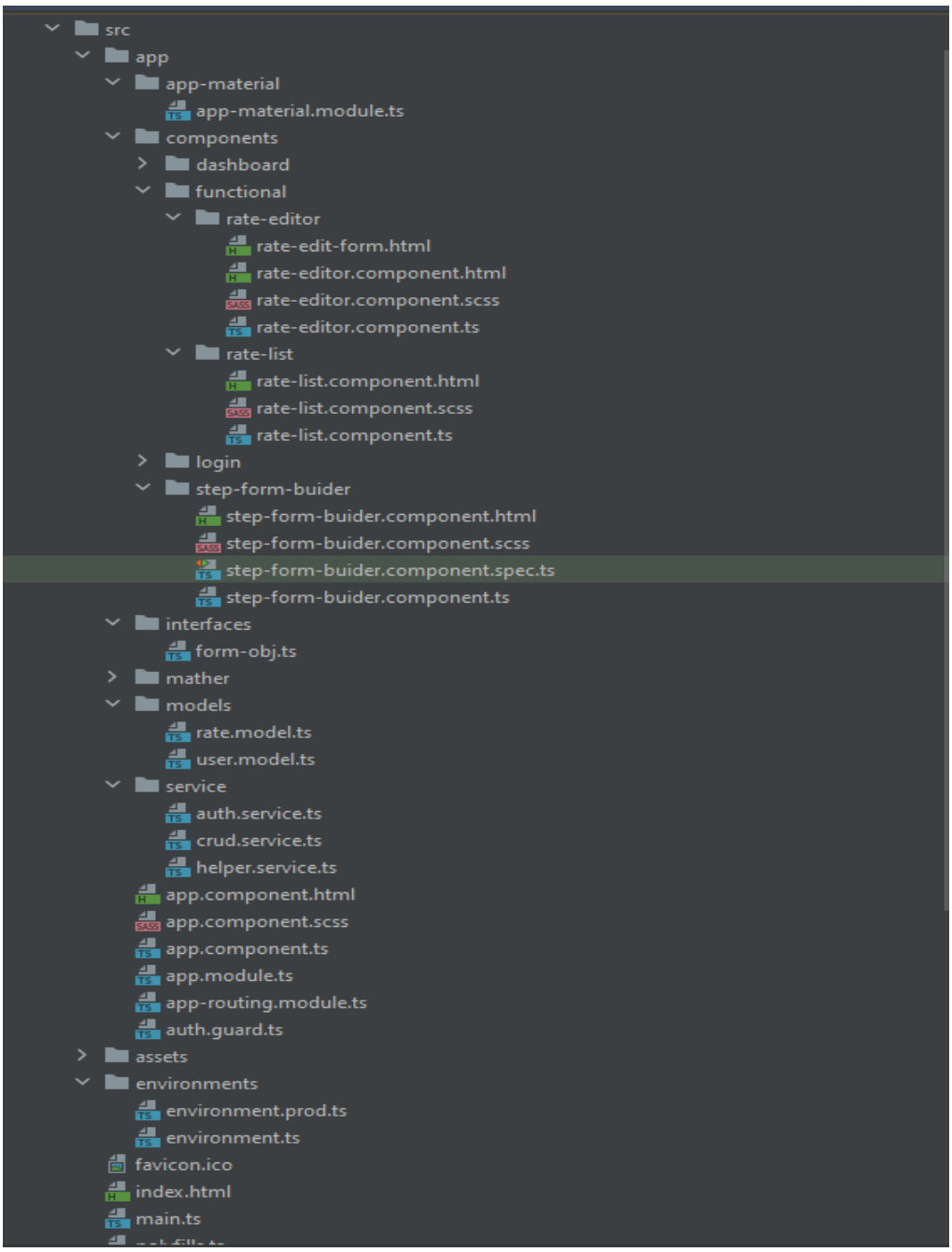

Рисунок 3.2 – Вид структури (library.econom.zp.ua) розробленого проєкту

Лістинг коду 3.3 – Вид модуля Materials

```
import { NgModule } from '@angular/core';
import {MatDialogModule} from '@angular/material/dialog';
import {MatFormFieldModule} from '@angular/material/form-field';
import {MatInputModule} from '@angular/material/input';
import {MatStepperModule} from '@angular/material/stepper';
import {MatButtonModule} from '@angular/material/button';
import {MatSnackBarModule} from '@angular/material/snack-bar';
```

```
     import {MatChipsModule} from '@angular/material/chips';
     import {MatIconModule} from '@angular/material/icon';
     import {MatCardModule} from '@angular/material/card';
     import { FontAwesomeModule } from '@fortawesome/angular-
fontawesome';
     import {MatToolbarModule} from '@angular/material/toolbar';
     import { NgxLetterImageAvatarModule } from 'ngx-letter-image-
avatar';
     import {MatSidenavModule} from '@angular/material/sidenav';
     import {MaterialFileInputModule} from 'ngx-material-file-
input';
     import {MatSelectModule} from '@angular/material/select';
     @NgModule({
     declarations: [],
     exports: [
   MatDialogModule,
   MatFormFieldModule,
   MatInputModule,
   MatStepperModule,
   MatButtonModule,
   MatSnackBarModule,
   MatChipsModule,
   MatIconModule,
   MatCardModule,
   FontAwesomeModule,
   MatToolbarModule,
   NgxLetterImageAvatarModule,
   MatSidenavModule,
   MaterialFileInputModule,
   MatSelectModule ]
}) export class AppMaterialModule { }
```

Моделі – це інтерфейси TypeScript, які допомагають типізувати об'єкти в додатку, тим самим полегшуючи розробку. На лістингу 3.4 зображено код інтерфейсу поля меню, який описує об'єкт меню - поля назви, структуру фото.

Лістинг коду 3.4 – Вид модуля графічного інтерфейсу

```
export interface Item {
 rateName: string;
 rateTagline: string;
 tags: Array<string>;
 price: number;
 photos: Array<{name: string, src: string}>;
 details: string;
```

}

Сервіси в Angular представляють досить широкий спектр класів, які виконують деякі специфічні завдання, наприклад, логування, роботу з даними і т.д. На відміну від компонентів і директив сервіси не працюють з (ela.kpi.ua) уявленнями, тобто з розміткою html, не роблять на неї прямого впливу. Вони виконують строго певне і досить вузьке завдання. На лістингу 3.5 зображено сервіс аутентифікації, який реалізує можливості логування, реєстрації, відновлення паролю та аутентифікації через Google.

Лістинг коду 3.5 – Вид розробленого сервіс програмної аутентифікації

```
import {Injectable, NgZone} from '@angular/core';
import {User} from '../models/user.model';
import {auth} from 'firebase/app';
import {AngularFireAuth} from '@angular/fire/auth';
import {AngularFirestore, AngularFirestoreDocument} from
'@angular/fire/firestore';
import {Router} from '@angular/router';
@Injectable({
    providedIn: 'root'})
export class AuthService {
    userData: any;
    constructor(
  public afs: AngularFirestore,
  public afAuth: AngularFireAuth,
  public router: Router,
  public ngZone: NgZone)
{
  this.afAuth.authState.subscribe(user => {
    if (user) {
      this.userData = user;
      localStorage.setItem('user',
JSON.stringify(this.userData));
      JSON.parse(localStorage.getItem('user'));
    } else {
      localStorage.setItem('user', null);
      JSON.parse(localStorage.getItem('user'
(stackoverflow.com)); (stackoverflow.com)     }
  });
    email: user.email,
    displayName: user.displayName,
    photoURL: user.photoURL,
    emailVerified: user.emailVerified
```

```
      };
      return userRef.set(userData, {
        merge: true
      });
    }
    // Sign out
    SignOut(){ (stackoverflow.com)
      return this.afAuth.signOut().then(() => {
        localStorage.removeItem('user');
        setTimeout(() => {
          this.router.navigate(['sign-in']);
        }, 100);
      });
    }}
```

Одним з ключових елементів програми є компоненти. Компонент управляє відображенням уявлення на екрані.

Лістинг коду 3.6 – Вид коду dashboard.component.html

```
    <ng-container>
     <mat-toolbar class="header-bar">
     <div       *ngIf="authService.userData       as       user"
(stackoverflow.com)                              class="user-container"
(click)="sidenav.toggle()">
         <img                                    class="app-avatar"
ngxLetterImageAvatar="{{(user.displayName)  ?  user.displayName  :
'User'}}"
         src="{{(user.photoURL)      ?      user.photoURL      :
'/assets/dummy-user.png'}}"
         alt="{{user.displayName}}">
     <span>
       {{(user.displayName) ? user.displayName : 'User'}}
     </span>
   </div>
 </mat-toolbar>
<mat-sidenav-container class="page-container">
   <mat-sidenav #sidenav mode="side" class="user-sidenav">
width: 18%;
 justify-content: space-between;}
.header-bar {
 position: fixed;
 top: 0;
 left: 0;
 right: (er.nau.edu.ua)
 z-index: 2;}
.footer-bar {
 position: fixed;
```

```
 bottom: 0;
 left: 0;
 right: 0;
}
.page-container {
 padding-top: 30px;
 position: absolute; (er.nau.edu.ua)
 top: 60px;
 bottom: 60px;
 left: 0;
 right: 0;}
.user-sidenav {
 padding-top: 3’0px;
 overflow-x: hidden;
 display: flex;
 align-items: center;
 justify-content: center;
 width: (css-tricks.com) 200px;}
```

3.4.2 CRUD-сервіс

Даний сервіс використовується для роботи в серверною частиною додатку через AJAX-запити. Його код представлено нижче:

```
import {Injectable} from ‘@angular/core’;
import {Product, ProductData} from ‘../models/product.model’;
import {Observable} from ‘rxjs’;
import {HttpClient} from ‘@angular/common/http’;
import {environment} from ‘../../environments/environment’;
@Injectable({
  providedIn: ‘root’
})
export class CrudProductService {
  private readonly pathUrl = environment.backHost + ‘/products/’;
```

```
  constructor(private http: HttpClient) (ela.kpi.ua) {
  }
  public createDocument(data: ProductData): Observable<any> {
    console.log(data);
    return this.http.put(this.pathUrl + 'set-product', data);
  }
  public getDocuments(): Observable<any> {
    return this.http.get(this.pathUrl);
  }
  public updateDocument(data: Product): Observable<any> {
    console.log(data);
    return this.http.put(this.pathUrl + 'set-product', data);
  }
  public deleteDocument(data: Product): Observable<any> {
    console.log(data);
    return                this.http.delete(`${this.pathUrl}delete-
product/${data.id}`);
  }
}
```

3.4.3 Система сервісу користувачів

В лістингу коду 3.5 вже було представлено сервіс аутентифіувції, який реалізує можливості логування, реєстрації, відновлення паролю та аутентифікації через Google. Всі маніпуляції з користувачами записуються в базу, але завдяки правилам сторонній користувач не може модифікувати записи інших користувачів. А вже для навігації в додатку (основна частина функціоналу недоступна незареєстрованому користувачу) використовується auth.guard.ts.

Лістинг коду 3.9 guards зображено нижче.

Лістинг коду 3.9 – Вид розробленого файлу «Auth.guard.ts»

```
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, Router } from '@angular/router';
import { AuthService} from './service/auth.service';
import { Observable } from 'rxjs';
@Injectable({
providedIn: 'root'
})
export class AuthGuard implements CanActivate {
constructor(
public authService: AuthService,
public router: Router
){ }
canActivate(
next: ActivatedRouteSnapshot, state: RouterStateSnapshot): (stackoverflow.com) Observable<boolean> | Promise<boolean> | boolean {
if (this.authService.isLoggedIn !== true) {
this.router.navigate(['sign-in']);
}
return true; }}
```

Так званий «Захисник» – використовується в основному модулі додатку. Показано лістинг 3.10 використання наведено нижче.

Лістинг коду 3.10 – Вид коду маршрути додатку

```
const routes: Routes = [
{path: 'sign-in', component: SignInComponent},
{path: 'register-user', component: SignUpComponent}, (www.freewebsolution.it) {
path: '', component: DashboardComponent, canActivate: [AuthGuard],
children: [
{
path: 'rate-list',
```

```
component: RateListComponent
}
]
},
{path: 'forgot-password', component: ForgotPasswordComponent},
{path: 'verify-email-address', component: VerifyEmailComponent},
(www.freewebsolution.it)
{path: '**', redirectTo: '', pathMatch: 'full'}
];
```

3.4.5 Реєстрація в додатку.

«Password» – графічне представлення вбачається на рисунку 3.3. це поля імені користувача та паролю. Якщо користувач хоче зайти за допомогою свого аккаунту Google, то це можна зробити через окрему форму в звичному форматі з введенням електронної пошти з паролем у відповідних полях, показано принцип користування формою акаунта на рисунку 3.4.

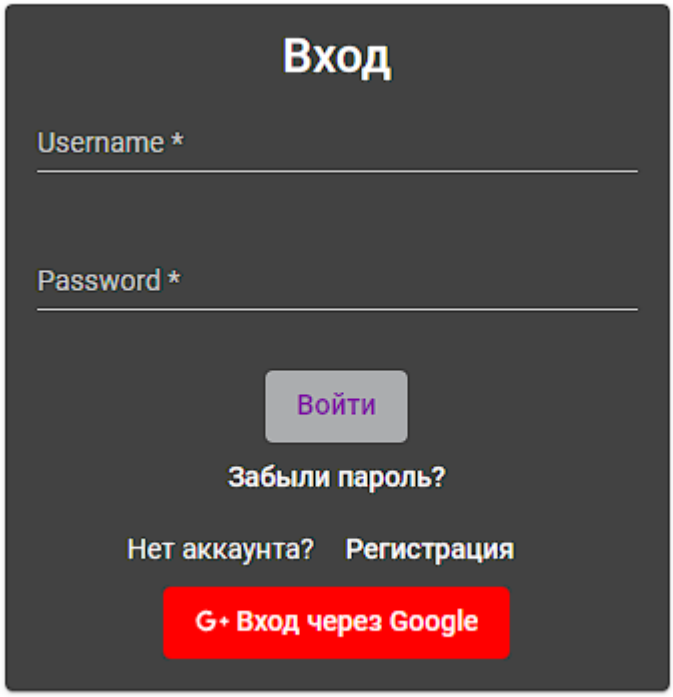

Рисунок 3.3 – Вид форми “Login” для авторизації

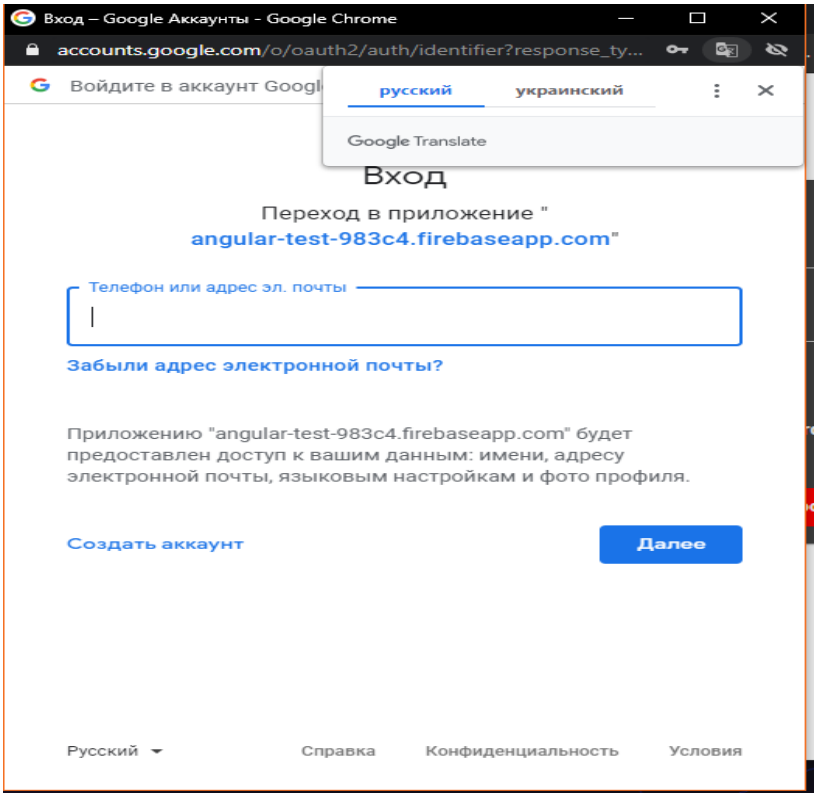

Рисунок 3.4 – Вид форми Google

На формі під назвою «Регистрация»– реєстрації користувач може створити аккаунт. Форма наведена нижче, на рисунку 3.5. зробивши покликання на кнопку «Регистрация».

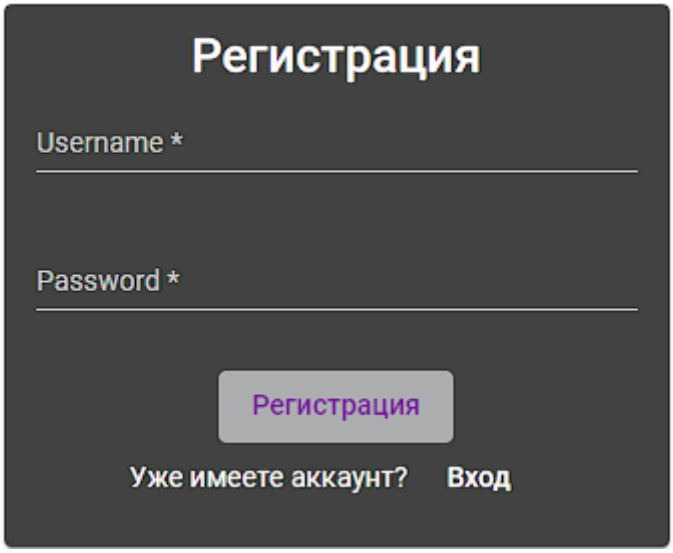

Рисунок 3.5 – Вид форми реєстрації «Registration»

В даній розробці передбачено можливість відновити пароль. «Восстановить пароль» - форма відновлення наведена нижче на рисунку 3.6. Де також треба буде в поле Email Address - (*поле обов’язкового введення), як для всіх відомий запит введення електронної пошти.

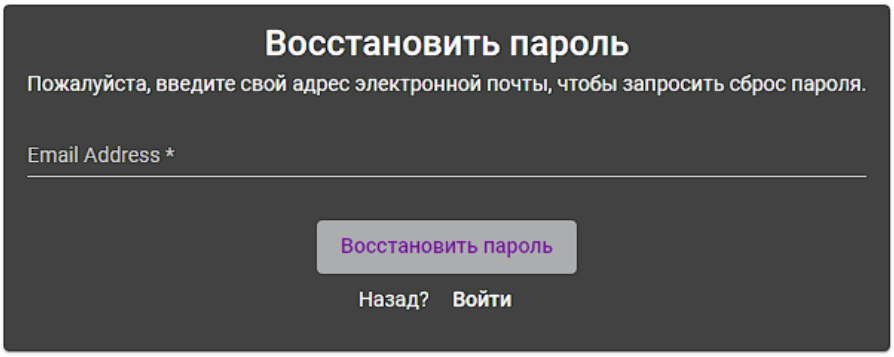

Рисунок 3.6 – Вид форми «Восстановить пароль»

3.4.6 Будівельник форм

Однією з особливостей даного додатку є будівельник динамічних форм розробки. Будівельник форм представляє собою компонент - Angular, який приймає конфігурацію форми в вигляді js-об’єкту та будує по цій конфігурації форму. При відправки форми компонент повертає об’єкт з полями згідно конфігурації. Готова форма зображена на рисунку 3.7.

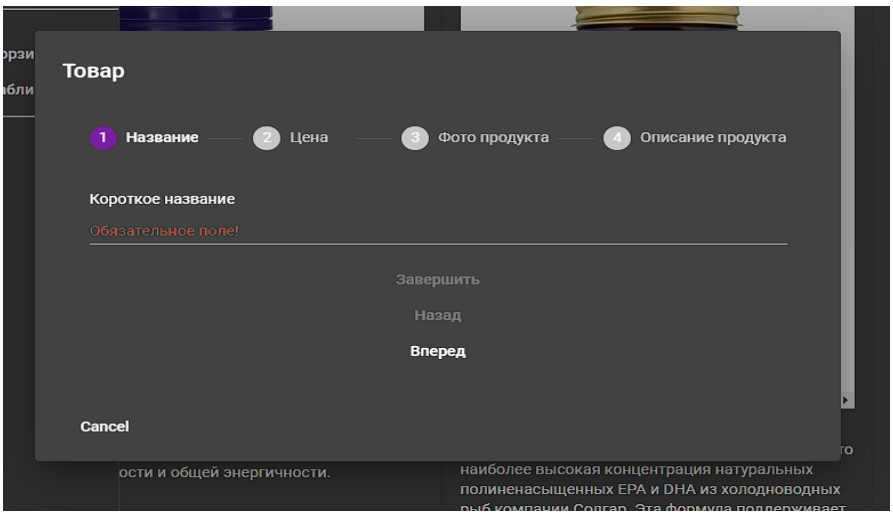

Рисунок 3.7 – Вид форма продутку

3.4.7 Конфігурація форми

Об’єкт конфігурації описується розробленим інтерфейсом, який наведено нижче.

Лістинг коду 3.11 – Вид інтерфейс конфігурації

```
interface Field {
 name: string;
 label: string;
 fieldType: string;
 placeholder: string;
 validators: Array<‘email’ | ‘required’ | ‘minLength’ | ‘maxLength’ | ‘mobile’ | ‘price’ | ‘numeric’>;
}
export interface StringField extends Field{
 type: string;
 preparingValue?: string;
}
export interface ChipListField extends Field{
 minTagsQuantity: number;
 maxTagsQuantity: number;
```

```
 preparingValue?: Array<string>;
}
export interface  Textarea extends Field{
 minSymbolsQuality: number;
 maxSymbolsQuality: number;
 minRowsQuality: number;
 maxRowsQuality: number;
 preparingValue?: string;
}
export interface ImagesField extends Field {
     minImagesQuality: number;
     maxImagesQuality: number;
 multiple: boolean;
 preparingValue?: Array<{name: string, src: string}>;
}
export interface FormObj {
 name: string;
 steps: Array<{
   stepName: string;
       stepLabel: string;
   fields: Array<StringField | ChipListField | Textarea |
ImagesField>;
 }>;
}
```

Згідно інтерфейсу і створюється об’єкт. В ньому є обов’язкові поля та не обов’язкові.

3.4.8 Компонент форми

Компонент форми представляє собою доволі громіздкий шаблон, в якому знаходяться можливі поля.

В файлі ts описана логіка взаємодії з батьківськими компонентами та хелперні функції для полів.

Лістинг коду 3.13 – Вид шаблону компонента форми

```
form action="" [formGroup]="globalFormGroup">
     <mat-horizontal-stepper #stepper linear>
   <mat-step *ngFor="let step of formObject.steps"

[stepControl]="globalFormGroup.controls[step.stepName]"
             formGroupName="{{step.stepName}}"
             errorMessage="Заполните обязательные поля.">
     <ng-template matStepLabel>{{step.stepLabel}}</ng-template>
     <mat-form-field *ngFor="let field of step.fields">
       <ng-template [ngIf]="field.fieldType === 'text-input'">
         <mat-label>{{field.label}}</mat-label>
         <input matInput
                [formControlName]="field.name"
                [placeholder]="field.placeholder"
                [(ngModel)]="preparingObj[field.name]">
         <mat-error
*ngIf="!!globalFormGroup.controls[step.stepName].controls[field.
name].invalid">Обязательное поле!</mat-error>
       </ng-template>
       <ng-template [ngIf]="field.fieldType === 'chip-list'">
         <mat-chip-list #chipList aria-label="Fruit selection">
           <mat-chip *ngFor="let tag of preparingObj[field.name]"
[selectable]="true"
                     [removable]="true"
                     (removed)="removeIntoChipList(tag,
field.name,  step.stepName,  {min:  field.minTagsQuantity,  max:
field.maxTagsQuantity})">
             {{tag}}
             <mat-icon matChipRemove>Отмена</mat-icon>
           </mat-chip>
           <input [placeholder]="field.placeholder"
                  [matChipInputFor]="chipList"

[matChipInputSeparatorKeyCodes]="separatorKeysCodes"
```

```
                [matChipInputAddOnBlur]="true"
                (matChipInputTokenEnd)="addToChipList($event,
field.name,  step.stepName,  {min:  field.minTagsQuantity,  max:
field.maxTagsQuantity})">
        </mat-chip-list>
        <mat-error   *ngIf="preparingObj[field.name].length   <
field.minTagsQuantity">Нужно больше тэгов!</mat-error>
<mat-error        *ngIf="preparingObj[field.name].length        >
field.maxTagsQuantity">Нужно меньше тэгов.</mat-error>
<mat-hint>От             {{field.minTagsQuantity}}             до
{{field.maxTagsQuantity}} тэгов</mat-hint>
      </ng-template>
      <ng-template [ngIf]="field.fieldType === 'textarea'" >
        <mat-label>{{field.label}}</mat-label>
        <textarea matInput
 cdkTextareaAutosize
 placeholder="{{field.placeholder}}"
cdkAutosizeMinRows="{{field.minRowsQuality}}"
                 cdkAutosizeMaxRows="{{field.maxRowsQuality}}"
```

Також є стилі. Завдяки глобальним стилям та використанню Material Design, файл стилів дуже малий та описує лише стилі галереї фото.

Лістинг коду 3.15 – Стилі компоненту

```
.image-list-container {
 width: 100%;
 display: flex;
 flex-direction: row;
 > img {
   max-width: 100%;
   display: flex;
   margin-left: 10px;
   margin-right: 10px; }}
```

3.4.9 Використання будівельника форм

В компонент будівельника форм потрібно передати конфігурацію форми та callback-функцію, яка буде опрацьовувати вихідні дані та реагувати на них.

Лістинг коду 3.16 – Вид коду використання компоненту форми

```
<app-step-form-buider
(finishEvent)="finish($event)"
[formObject]="testFormObj"></app-step-form-buider>
```

3.4.10 Кошик та оплата

В додатку є сторінка кошика. Вона зображена на рисунку 3.8. Тут користувач може редагувати своє замовлення та оплатити його.

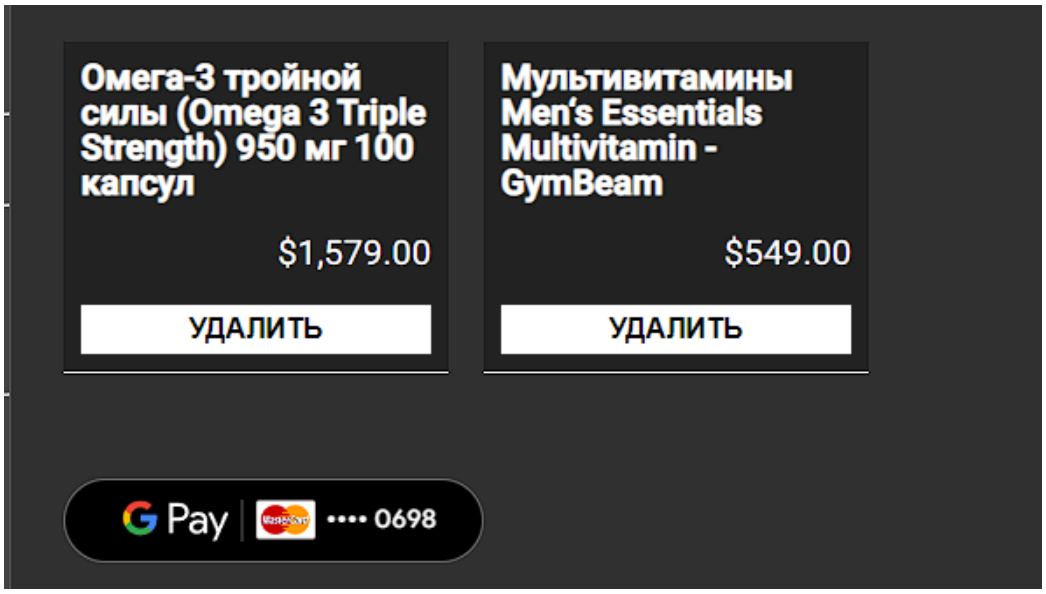

Рисунок 3.8 – Сторінка кошику

Оплата проходить через сервіс гугла, а саме Google Pay. Форма оплати зображена на рисунку 3.9.

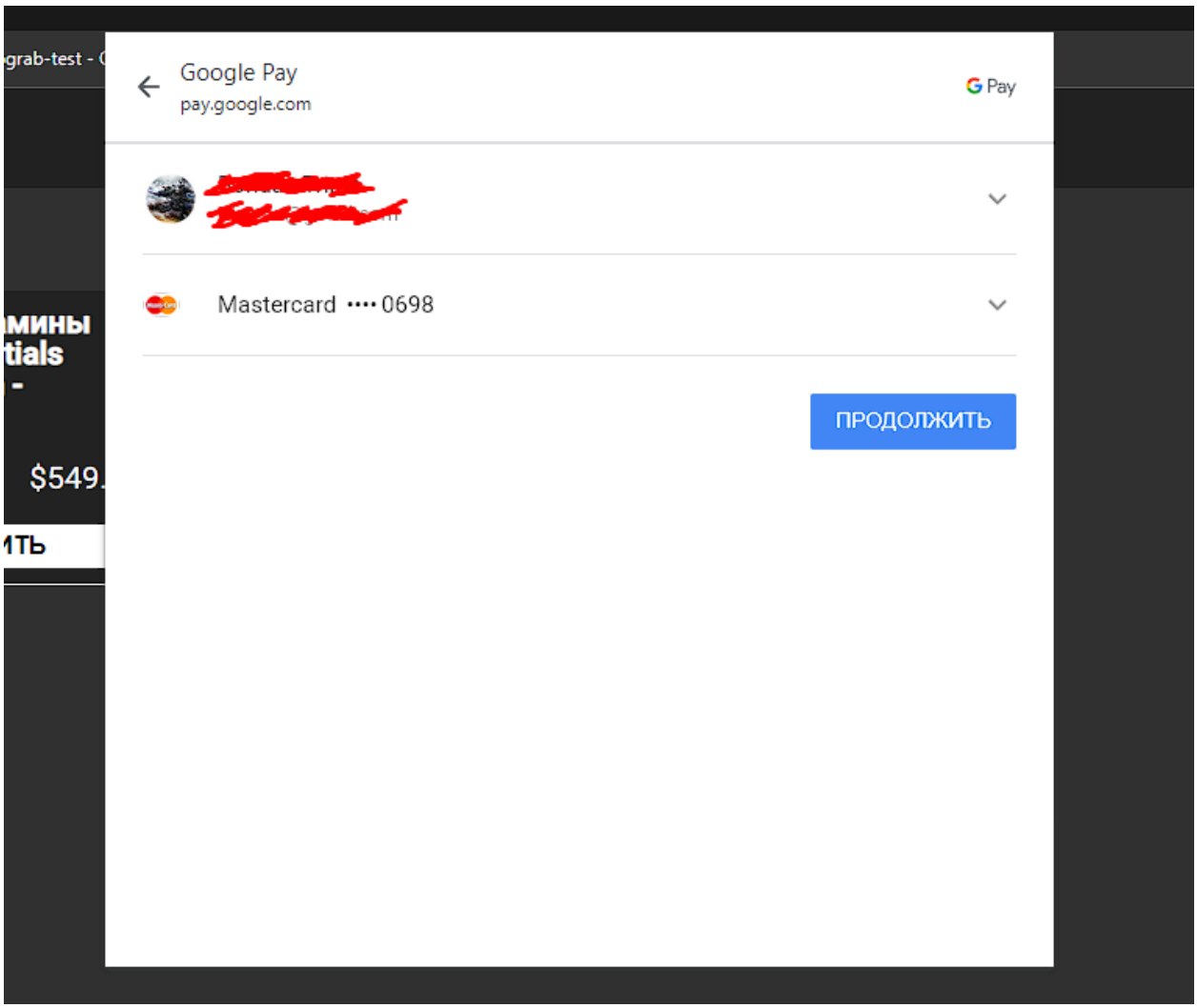

Рисунок 3.9 – Форма оплати

3.4.11 Структура та код серверної частини проекту

Серверна частина слідує структури, подібної до фронт-частини. Скріншот зі структурою представлено нижче.

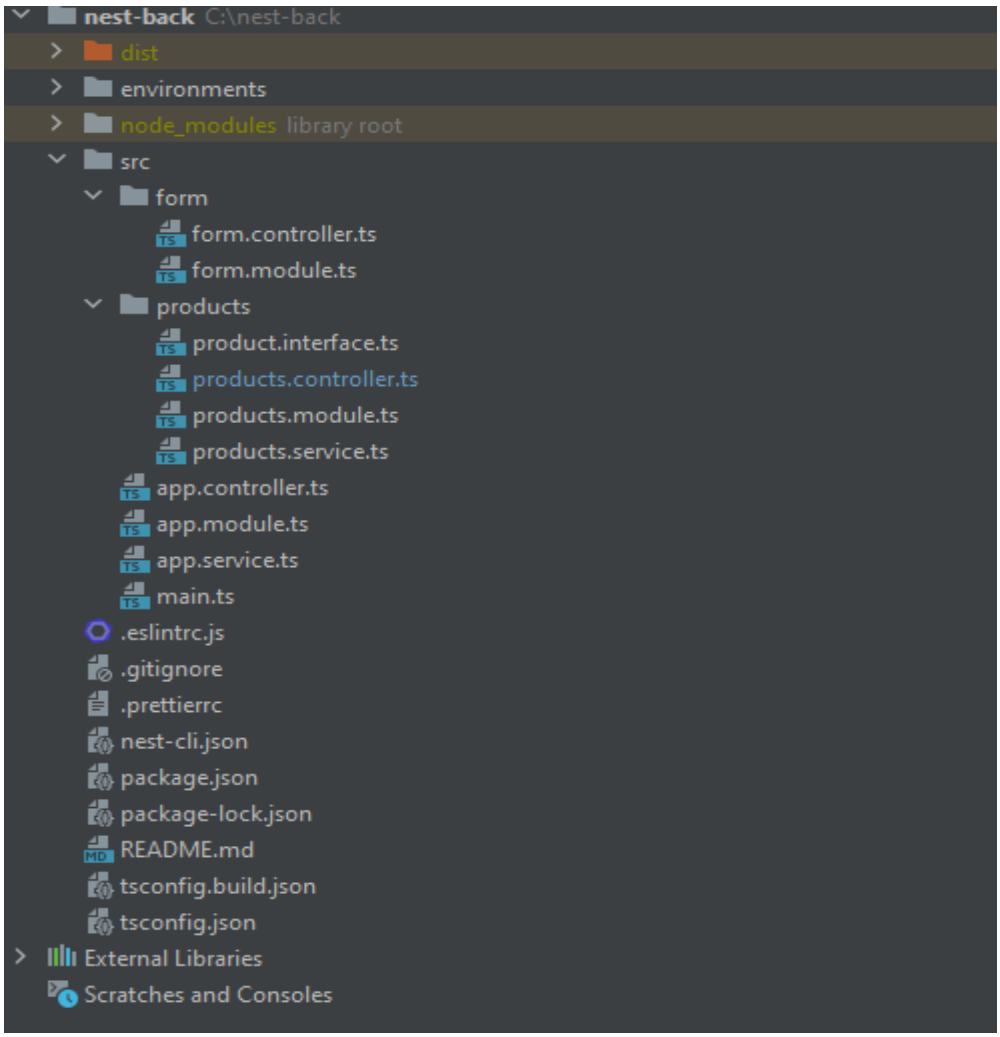

Рисунок 3.10 – Структура серверної частини

Додаток складається з двох модулів – модуль форм та модуль продуктів. Перший з них відповідає за видачу шаблону форм з бази даних. Нижче представлено код контролера:

```
import { Controller, Get } from '@nestjs/common';
import { FirebaseAdmin, InjectFirebaseAdmin } from 'nestjs-firebase';
@Controller('form')
export class FormController {
  constructor(
    @InjectFirebaseAdmin() private readonly firebase: FirebaseAdmin,
  ) {}

  @Get('product-template')
  async getProductTemplate() {
    try {
      const docRef = await this.firebase.db
```

```
        .collection('forms')
        .doc('Product')
        .get();
      return docRef.data();
    } catch (e) {
      throw e;
    }
  }

  @Get('service-template')
  async getServiceTemplate() {
    try {
      const docRef = await this.firebase.db
        .collection('forms')
        .doc('Service')
        .get();
      return docRef.data();
    } catch (e) {
      throw e;
    }
  }
}
```

В базі даних зберігаються статичні шаблони форм. Нижче представлено скріншот з таким шаблоном:

Другий контролер відповідає за продукти. Так як він не тільки читає дані, але і модифікує їх, для зручності було створено окремий сервіс, де описано дані операції.

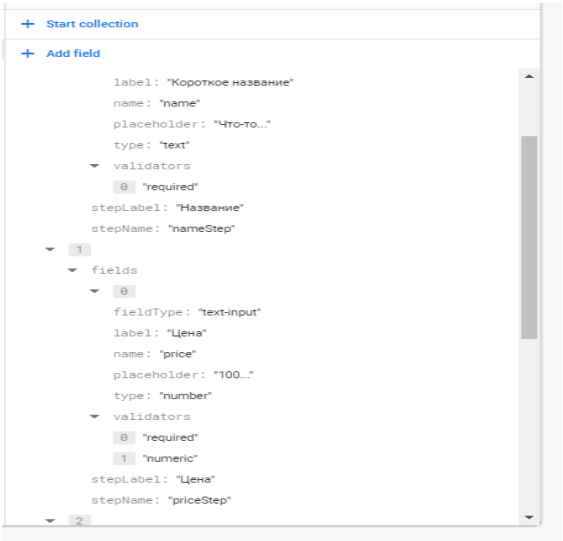

Рисунок 3.11 – Шаблон форми продукту

Нижче представлено код сервісу продуктів:

```
import { HttpException, HttpStatus, Injectable } from '@nestjs/common';
import { FirebaseAdmin, InjectFirebaseAdmin } from 'nestjs-firebase';
import { Product } from './product.interface';


@Injectable()
export class ProductsService {
constructor(
@InjectFirebaseAdmin() private readonly firebase: FirebaseAdmin,
) {}

public async getAllProducts(): Promise<Array<Product>> {
const querySnapshot = await this.firebase.db.collection('products').get();
const collection = [];
querySnapshot.forEach((doc) => {
collection.push({ id: doc.id, ...doc.data() });
});
return collection;
}

public async getProductById(id: string) {
const documentRef = await this.firebase.db
.collection('rates')
.doc(id)
.get();
const product = documentRef.data();
if (product) {
return documentRef.data();
} else {
throw new HttpException(
{
status: HttpStatus.FORBIDDEN,
error: 'Product not found',
},
HttpStatus.FORBIDDEN,
);
}
}
public async createProduct(product: Partial<Product>) {
return await this.firebase.db.collection('products').add(product);
}
public async updateProduct(product: Partial<Product>) {
```

```
    const                          docRef                          =
this.firebase.db.collection('products').doc(product.id);
    delete product.id;
    return await docRef.set(product, { merge: true });
    }
    public async deleteProduct(id: string) {
    console.log(id);
    return                                                      await
this.firebase.db.collection('products').doc(id).delete();
    }
    }
```

3.5 Висновок до третього розділу

В розділі висвітлено реалізацію практичної частини магістерської роботи, а саме програмний веб-сервіс та серверний додаток для приватної лікарні «Діасервіс». У вигляді рисунків показано графічні елементи програмного продукту показано різноманітні лістинги програмного коду. До кожного скріншота програмного продукту подано повний опис та викладено в розділі. Також в розділи показана робота сервісу як збоку адміністратора так і користувача. Всі дані зберігаються в базі даних та яка має відкритий кодинг з боку доступу системного адміністратора або інженера програмного забезпечення.

ВИСНОВОК

Під час виконання магістерської дипломної роботи було зібрано та опрацьовано багато матеріалу, як друкованої літератури так і в мережі Інтернет. Розглянуто методи розробки та інструментарій веб-додатків та роботи з BaaS-сервісом.

Завдяки широкому функціоналу фреймворку Angular, розроблена проєкт надає досвід використання приблизний до нативних систем. Використовуючи технологію веб-сокетів, готовий проєкт забезпечує повну синхронність роботи.

Отже розроблений сервіс має наступні розроблені властивості:

1. Меню веб-додатку;
2. Сторінка кошику «Діасервіс»;
3. Форма оплати «Діасервіс»;
4. Розвинену систему користувачів, яка підтримує синхронізацію з аккаунтом Google;
5. Досягнена повна синхронність всіх актуальних сесій.

Практична робота представлена у вигляді програмної розробки на мові програмування TS на основні фреймворку Angular, а теоретична частина описана в тестовому форматі з винесенням різнопланових пояснень та зображень (скріншоти програмної частини).

В роботі досягнуто основну мету та новизна: розроблено форму оплати та синхронізацію всіх активних сесій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Корисна інформація про HTML. URL : http://htmlbook.ru/html/
2. Мейер, Е.А. CSS-каскадні таблиці стилі: докладний посібник. Москва : Символ-Плюс, 2008. 573 с.
3. Документація по VueJS. URL : https://vuejs.org/v2/guide/.
4. Основні етапи створення сайту у web-студії. URL : https://impulse-design.com.ua/etapy-razrabotki-sajta.html.
5. Статичні і динамічні сайти сьогодні. URL :https://jino.ru/journal/articles/staticheskie-dinamicheskie-sayty/.
6. Сучасний підручник JavaScript: URL : https://learn.javascript.ru/.
7. Документація по БД Firebase та хостингу Firebase Hosting. URL : https://firebase.google.com/docs/.
8. Документація по NodeJs. URL : https://nodejs.org/uk/docs/.
9. Документація по Vuetify.js. URL : https://vuetifyjs.com/ru/components/api-explorer/.
10. Медична інформаційна система, що це? URL : https://blog.h24.ua/uk/shho-take-mis/.
11. Інформаційні технології для медицини. URL : https://www.infomed.ck.ua/.
12. Медична інформаційна система nHealth (Здоров’я Нації). URL : https://vikisoft.kiev.ua/nhealth/.