МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки», _____

освітньо-професійної програми «Інформаційні технології проектування» _____

на тему: **«Інформаційна система підтримки діяльності конструктора одягу»**

Здобувачки групи ІТмз.22с Жиленко Тетяни Іванівни

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Тетяна ЖИЛЕНКО
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник к.т.н., доц. Федотова Н.А. _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2024

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри ІТ

Світлана ВАЩЕНКО

«_____» _____ 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові
Жиленко Тетяні Іванівні

1 Тема кваліфікаційної роботи «Інформаційна система підтримки діяльності конструктора одягу»

затверджена наказом по університету від «08» листопада 2023 р. № 1249-VI
2 Термін здачі студентом кваліфікаційної роботи «___» ___грудня___ 2023 р.
3 Вхідні дані до кваліфікаційної роботи технічне завдання для розробки інформаційної системи, запит на розробку інформаційної системи

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз обраної предметної області, постановка задачі й методи дослідження, проектування та моделювання, практична реалізація інформаційної системи
5 Перелік графічного матеріалу (з точним зазначенням обов'язкових слайдів презентації) актуальність, постановка задачі, дослідження аналогів, порівняльна характеристика аналогів, контекстна діаграма процесу розробки, діаграма декомпозиції процесу розробки, діаграма варіантів використання користувача, діаграма варіантів використання адміністратора, реалізація проекту, база даних, архітектура додатку, реалізація інформаційної системи, посилання на інформаційну систему, публікації проекту, висновки.

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	**Консультант**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Підготовка специфікації	20.09.2023	
2	Вибір програми для розробки	20.11.2023	
3	Створення програмного продукту	16.11.2023	
4	Впровадження в дію	01.12.2023	
5	Налаштування та тестування	03.01.2024	

Магістрант _____ Тетяна ЖИЛЕНКО

Керівник роботи _____ к.т.н., доц. Наталія ФЕДОТОВА

АНОТАЦІЯ

Тема кваліфікаційної роботи є «Інформаційна система підтримки діяльності конструктора одягу».

Пояснювальна записка складається зі вступу, 4 розділів, висновку, списку використаних джерел із 39 найменувань, 2 додатків. Загальний обсяг роботи – 120 сторінок, у тому числі 74 сторінки основного тексту, 6 сторінок списку використаних джерел, 40 сторінок додатків.

Кваліфікаційна робота присвячена розробці інформаційної системи по організації допомоги конструктору одягу, що буде відповідати усім встановленим параметрам. Замовники отримають можливість взаємодіяти із ательє та отримати гарантію в забезпеченні якісними послугами.

Під час роботи було проведено детальний аналіз існуючих сервісів, що забезпечують галузь легкої промисловості. Також проведено їх порівняльний аналіз із наближеними функціональними можливостями. Сформовано мету, задачі та методи дослідження згідно теми розробки інформаційної системи, обрано засоби, за допомогою яких буде реалізована системи. Робота спроектована послідовно поетапно з огляду менеджера / адміністратора та замовника. Виконана програмна реалізація бази даних.

Результатом проведеної роботи є виконана інформаційна система підтримки діяльності конструктора одягу. Використання якої може сприяти поліпшенню оцифровізації процесу створення продукту швейного виробництва.

Використання такої інформаційної системи сприятиме вдалому та ефективному розподілу часу роботи працівників ательє та підприємств легкої промисловості.

Ключові слова: INFORMATION SYSTEM, OUTERWEAR, SERVICE, CATALOG, CALCULATOR, PATTERNS, COATS, ATELIER EMPLOYEE, USER-CUSTOMER, ADMINISTRATOR, PROPOSAL, FEEDBACK.

ЗМІСТ

ВСТУП

Актуальність. ІТ легкої промисловості на даний момент тримають два фронти: економічний і військовий. Уряд активізує співпрацю з вітчизняними підприємствами легкої промисловості, які забезпечують товарами народного споживання не тільки мирне населення, а і потреби ЗСУ та інші оборонні відомства [1-2].

В плані заходів з підтримки легкої промисловості України зі сторони держави на 2022-2024 роки є такі пункти [3]:

- підвищення рівня конкурентоспроможності українських виробників шляхом капітальних інвестицій у модернізацію основних засобів;
- розширення існуючих та створення нових виробництв за рахунок імпорту високоякісного сучасного обладнання з низькими виробничими витратами та високою якістю готової продукції для підвищення рівня конкурентоспроможності підприємств текстильної промисловості України;
- ефективне використання коштів державного бюджету у сфері підготовки та перепідготовки кадрів для легкої промисловості.

Одяг, що виготовляється в Україні користується значним попитом на міжнародному ринкуна. Україна має 146 країн партнерів. Понад 80% продукції експортується до країн Європи [4]. Найбільшими імпортерами текстильних виробів на сьогодні вважаються: Німеччина, Польща, Італія, Данія, Франція.

Тому важливу роль у розвитку української легкої промисловості відіграє застосування інноваційних технологій, особливо в період карантинів і воєнного стану. Адже саме використаття новітніх методів забезпечує збільшення продуктивності, поліпшенню якості, підвищеннння фінансового благополуччя підприємств і ательє, це в свою чергу сприяє підвищенню конкурентоспроможності на ринку, що безсумніву примножую економічні і соціальні показниках розвитку нашої країни [4-5].

Отже, була сформульована тема роботи, визначено об’єкт та предмет дослідження тощо. Розглянемо більш детально кожен з цих пунктів окремо.

Тема дослідження. Інформаційна система підтримки діяльності конструктора одягу.

Об’єкт дослідження. Процес взаємодії із замовниками за допомогою інформаційної системи.

Предмет дослідження. Методи оптимізації взаємодії ательє, підприємств легкої промисловості з клієнтами за для швидкого та висококваліфікованого виконання замовлення з урахуванням фізичних особливостей побудови тіла українського споживача.

Мета. Створити інформаційну систему у вигляді відкритої платформи для налаштування взаємодії замовників із ательє, підприємствами легкої промисловості задля масового і продуктивного виконання замовлення.

Щоб досягти мети проекту, необхідно виконати наступні задачі:

– виконати аналіз проблемної області, дослідити найсучасніші публікацій, запевнитися в актуальності теми та визначити аудиторію, на яку спрямоване використання даної технології;

– перевірити можливості аналогів інформаційних систем;

– проаналізувати та вказати технологій розробки даної інформаційної системи;

– створити моделі та структури інформаційної системи;

– довершити структуру інформаційної технології;

– впровадити функціонал;

– завершити впровадження шляхом тестування системи.

Практична цінність. Поліпшення якості виготовлення виробів і/або використання розроблених проектів в бізнес-планах підприємств легкої промисловості. Створення альбому варіантів плечового одягу. Вперше представлена українська інформаційна система для широкого загалу з

можливістю замовлення одягу, виконаного в традиціях української побудови лекал з врахуванням фізичних особливостей будови тіла пересічного українця.

Гіпотеза дослідження. Використання представленої інформаційної системи шляхом візуалізації та якісного підбору лекал пошиву одягу дозволить надавати якісні послуги.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз останніх досліджень та публікацій

Розвиток індустрії моди останнім часом все більше базується на використанні інноваційних технологій. Орлова Н. [6] вважає, щоб якість виробів стала вищою, потрібно автоматизувати процеси виготовлення виробів, а отже збільшити їх виробництво, до того ж зменшити собівартість. Це стає можливим завдяки активному впровадженню інформаційних та нанотехнологій в легку промисловість. Метою дослідження є саме визначення перспективних напрямів проектування одягу, нових технологій у легкій промисловості для навчання майбутніх фахівців.

Першу лінію займають наукові напрямки, які зосереджені на проектуванні одягу в 3D- просторі, це допомагає легкому сприйняттю клієнтами зображуваних схожих силуетів тіл людей і вибору різних об'ємних моделей одягу. Також відбувається зменшення залишків виробництва, що сприяє сталому його розвитку, йдеться у статтях Пашкевич К [7-8].

Автори [6-8] підсумували, що студенти, яких навчили використовувати 3D-технологій проектування одягу, є більш конкурентоспроможними на ринку легкої промисловості і здатні рухати і поглиблювати її розвиток.

Ідеєю даної магістерської роботи якраз і є спрощення процесу підготовки і виготовлення одягу завдяки сучасним технологіям.

1.2 Аналіз програмних продуктів

Серед програм із можливістю тривимірного проектування і візуалізації є Browzwear, Clo 3D, Marvelous designer, Tailornova, VidyaViewer.

Browzwear – найперша запровадила і представила свої результати на ринку моди. Вона складається з 5 середовищ, які виконують певні дії, що разом сприяють формуваню колекції [9] (рис.1.1). Дана програма може виконувати калькуляцію розмірних ознак, обирати тип тканини, кольорів, фасонів. Програма оптимальна для розробки виробничого процесу: створення та корекція моделі, у тому числі при зміні контурів наявних зразків.

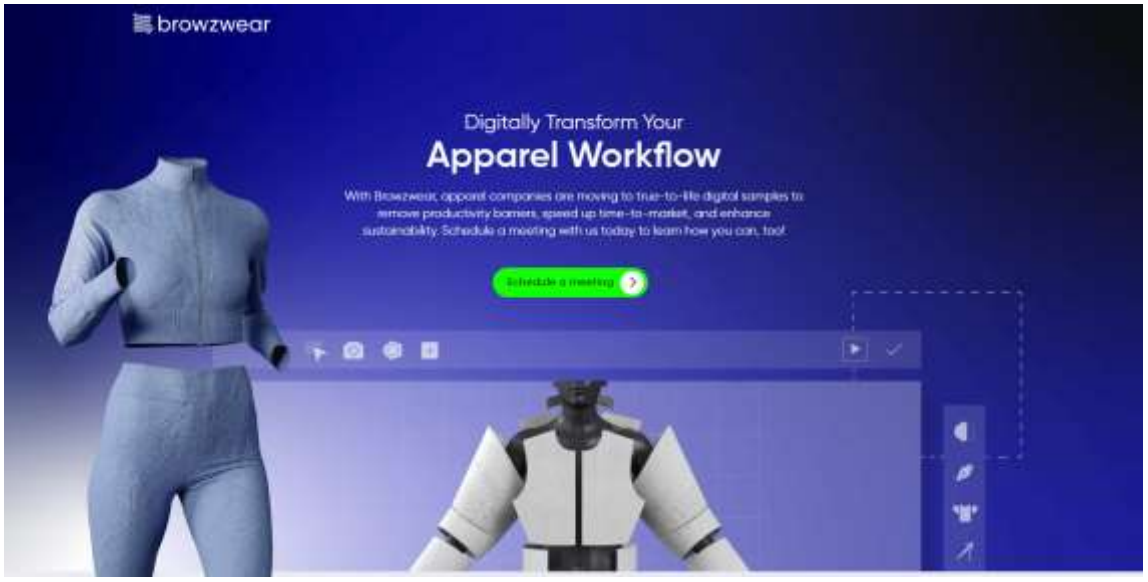


Рисунок 1.1 – Головна сторінка сайту.

Джерело [9]

Clo 3D – дуже затребувана програма, за рахунок реалістичного рендеру створення реалістичних образів. Вона має готові шаблони, які можна змінювати і перетворювати на власні 3D моделі одягу у форматі 360 градусів з можливістю

запису відео. Аватар, на який примірятиметься одяг, можна редагувати, щоб максимально наблизити його до індивідуальної фігури [10-15] (рис.1.2).

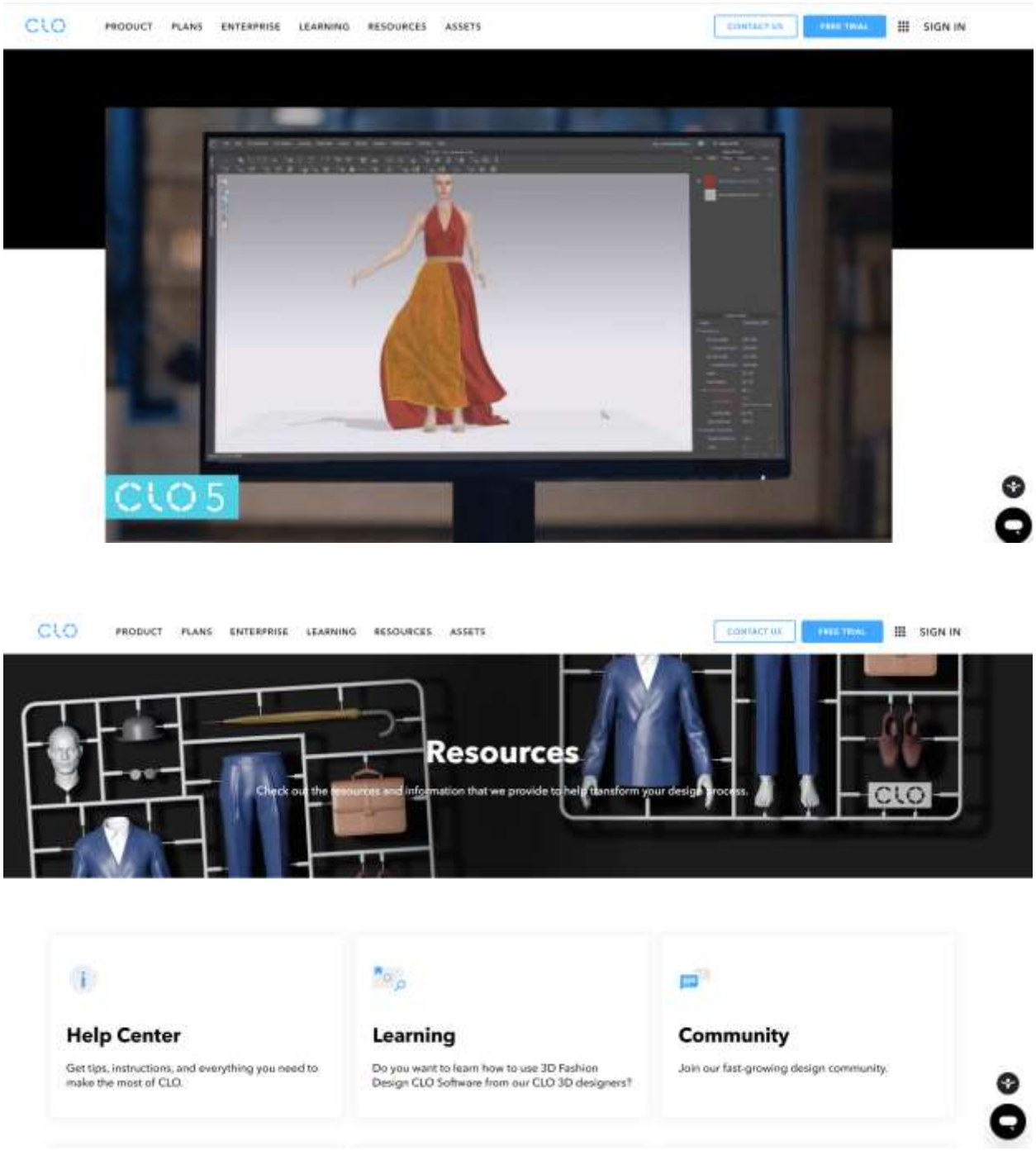


Рисунок 1.2 – Сторінка з варіантами пропозицій.

Джерело [11]

Marvelous designer – програма для створення тривимірних елементів одягу та інших виробів з тканини та шкіри: штор, покривал, оббивки меблів [7, 16].

Програма працює з площинної та тривимірної побудовою лекал та ескізів одягу. Таке проектування дає можливість робити навіть промислові зразки одягу, де є можливість вносити зміни в реальному часі. Всі вироби можна відразу проектувати та приміряти на віртуальну модель людини.

Рисунок 1.3 – Головна сторінка сайту Marvelous designer.

Джерело [7]

Tailornova – доступна лише онлайн [9]. Програма має готові викройки певних моделей, які можна спроектувати за власними розмірами. Алгоритм 3D Fit Model — допомагає відобразити вигляд моделі. Готові лекала можна роздрукувати або зберегти у форматі PDF (рис.1.4).

Vidya Viewer – виконує найскладніші завдання візуалізації виробів [17], що мінімізує витрати на прототипи моделей до 60%. Програму можна використовувати, як мережу для конструкторів, дизайнерів, керівників проектів та менеджерів для обміну інформацією новими ідеями та пошиття зразків (рис.1.5).

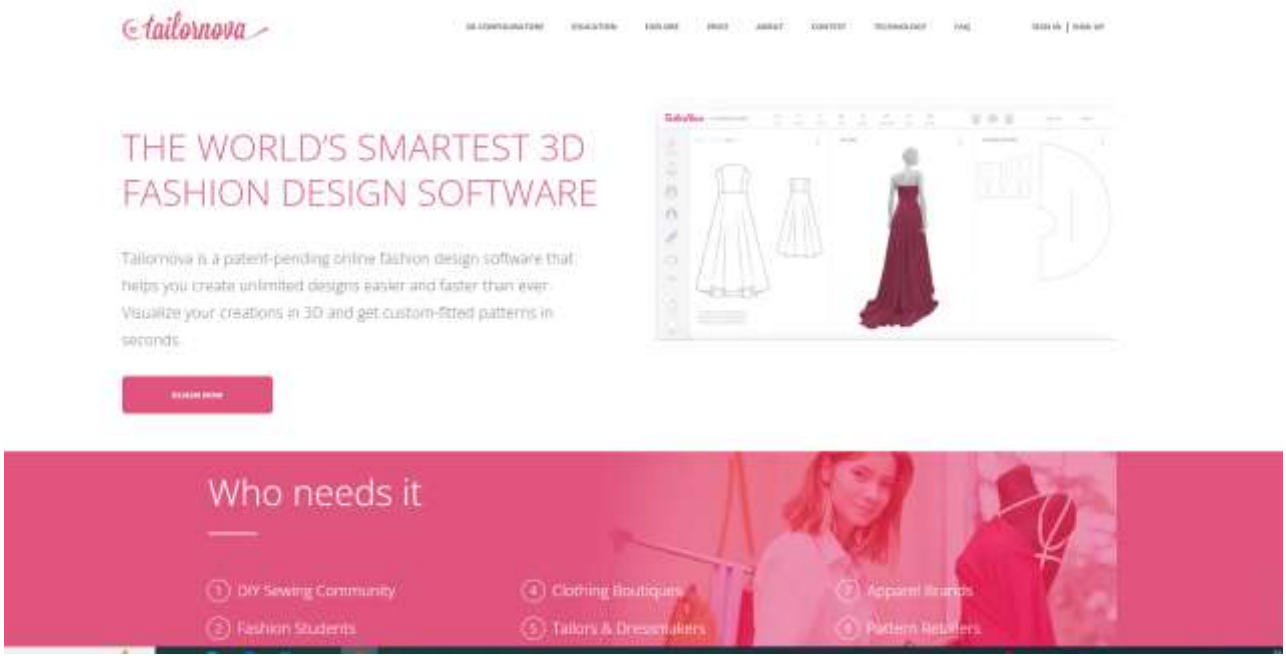


Рисунок 1.4 – Сторінка з лекалами.

Джерело [9]

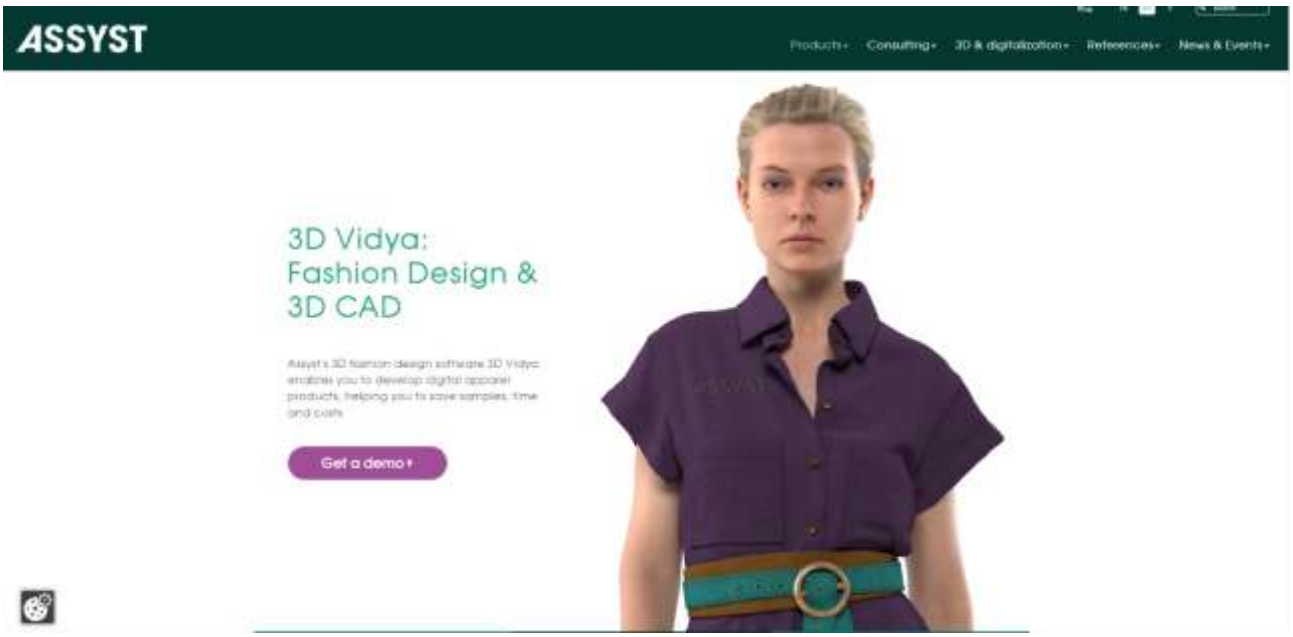


Рисунок 1.5 – Головна сторінка з візуалізації одягу.

Джерело [9]

Аналізуючи ці програми автори публікацій [9 – 19] найбільше зупинялися на Clo 3D, Marvelous designer і вважають їх найбільш точними і придатними для використовування в промисловості. Автори вказували на деякі недосконалості програм, адже тривимірний простір це все таки не реальний світ. Вони описали переваги в можливості швидкої реалістичної розробки моделей одягу із найменшими затратами. І це є неоціненний крок в віртуальному проектуванні.

Віртуальний одяг – це такий одяг який створений в цифровому тривимірному вигляді в спеціальних програмах. Значним стрибком у використання такого одягу, став час пандемії , аналізує Борщевська Н [20]. Весь digital-одяг можна розділити на три напрямки [21]:

-3D-моделювання для подальшого виробництва;

-3D-моделювання для pr-контенту ;

-3D-моделювання для метавсесвітів.

Так наприклад, Dress-x – проект, [22] заснований в 2019 році в Лос-Анджелесі двома українками уже має на платформі (рис.6) понад 2000 товарів, серед яких не лише вбрання, а й аксесуари, тобто можна підібрати повний комплект одягу до $2000 за базову комбінацію товарів, та купити окремо кожну модель, починаючи від 75$.

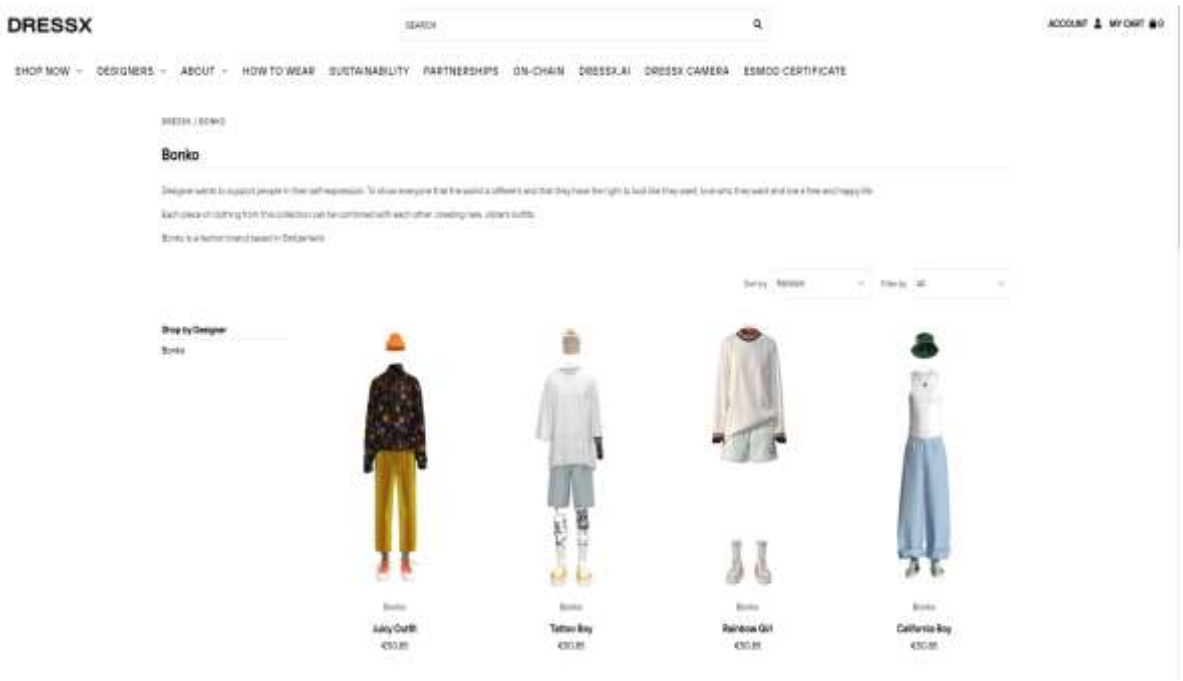


Рисунок 1.6 – Підбір пасуючого одягу.

Джерело [22]

Купуючи одяг у звичайних магазинах, люди обов´язково його приміряють. Нажаль в інтернет-магазинах такої можливості немає і нерідко покупці залишаються незадоволеними своїм придбанням. Віртуальна можливість створення світу моди уже досягла значних успіхів у віртуальній можливості примірки одягу, пишеться у статті [23]. Зараз вже можна побачити на теренах інтернету «віртуальні примірочні» виробів з оцінкою якості посадки на фігуру виробу. Застосування технологій «віртуальної примірки» і «3D-віртуального магазину» можуть збільшити попит і задоволеність покупців, підтвердив в своїй роботі Парк Х [24].

У роботі [25] автори пропонують Advanced Virtual Apparel Try use Augmented Reality (AVATAR) - мультисенсорний сканер тіла в поєднанні з технологіями соціальних мереж. Система має веб-камеру та TensorFlow, який визначає розмірні ознаки тіла в реальному часі. Бібліотека OpenCV збільшує 3D-модель. Система реагує на жести рук, вона також може визначати колір обличчя та рекомендувати клієнтам відповідні кольори одягу.

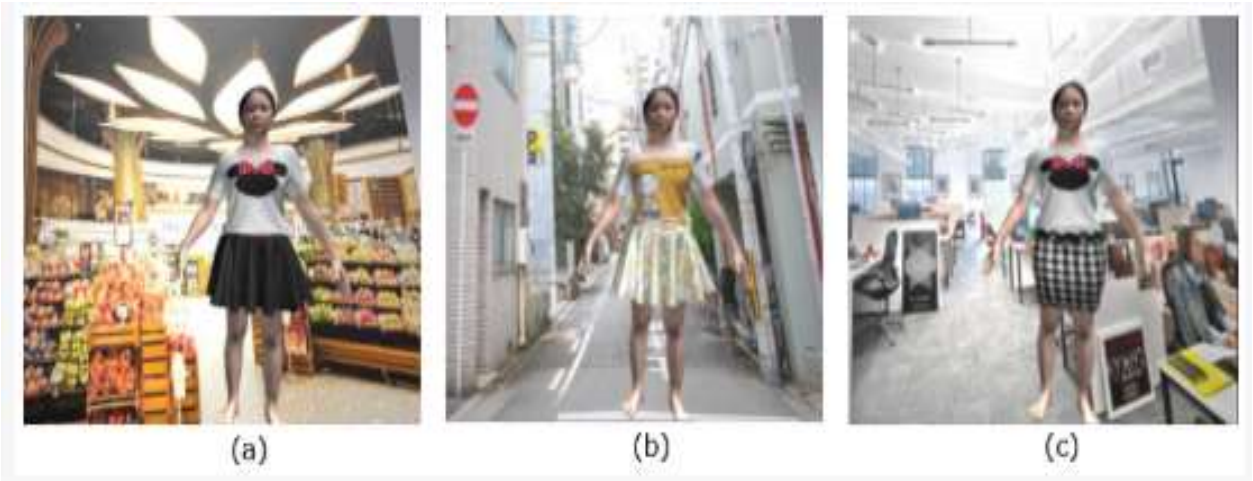


Рисунок 1.7 –3D підбір одягу.

Джерело [25]

Автори статті [26] пишуть про модний дисплей із доповненою реальністю, що намагається спростити процес покупок. Продукт здатний ідентифікувати особу, відображати на ній одяг і аксесуари у режимі реального часу. Застосування цієї технології дозволяє модельєрам створювати 3D моделі і тестувати їх віртуально, перш ніж витрачати матеріали та ресурси для їх відшивання в живу.

У роботах [27-29] досліджували якість виконання системи віртуальної примірки в залежності від пози споживача.

Автори [27] пропонують багатопозиційну керовану віртуальну пробну мережу – MG-VTON, що розбиває процес примірки на три етапи. А саме: аналіз фігури тіла, синтез зображення, враховуючи пози людини і потім продовжує візуалізувати текстури для найбільш реального вигляду. Мережу продовжують доповнювати різними позами для реалістичності програми.

Автори [28] візуалізували тканину завдяки CP- VTON. Ця концепція 3D-реконструкції змінює одяг відповідно до людських параметрів там, де існуючі методи VTON не працюють.

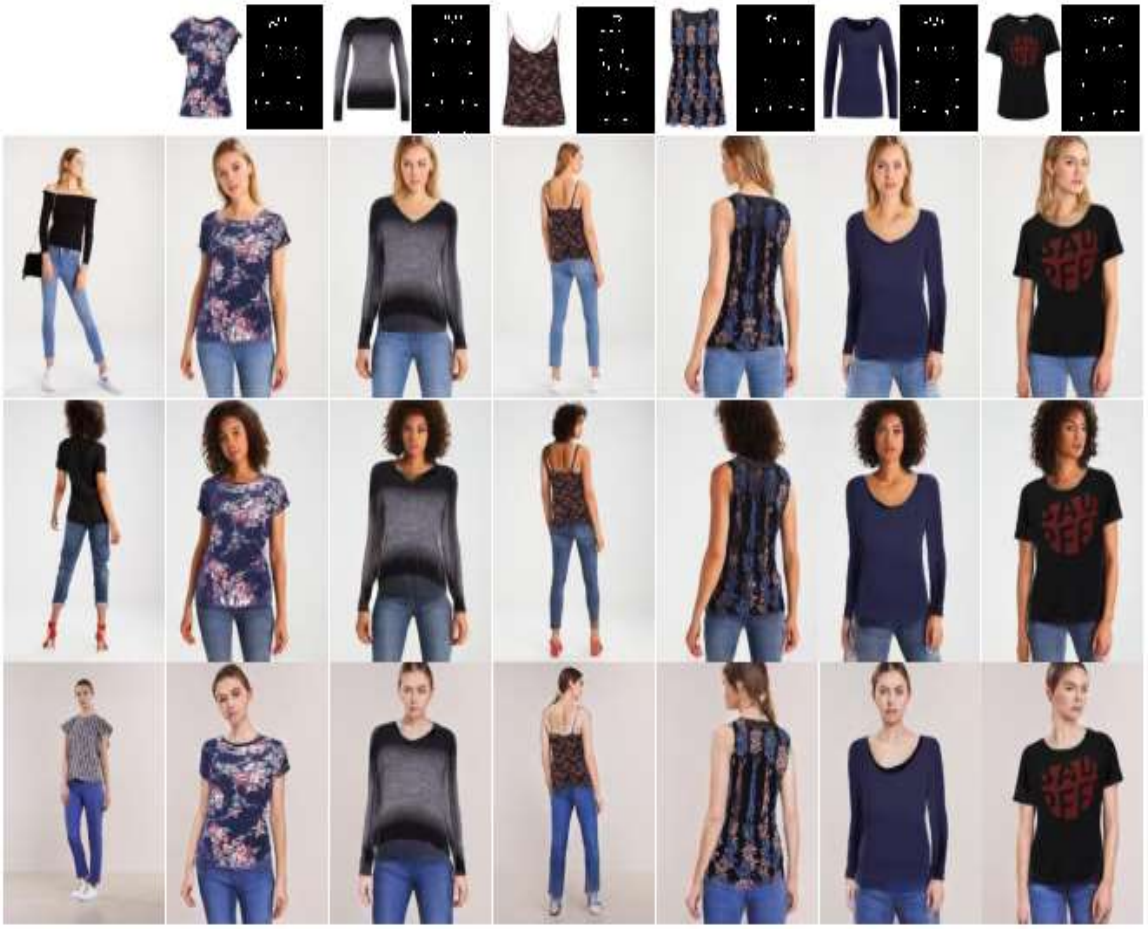


Рисунок 1.8 – Підбір тканини блузи.

Джерело [28]

У статті [30] йдеться про те, що деякі методи не розрізняють одяг і не одяг, що заважає правильності віртуальної примірки. Тому автори пропонують пробну мережу роз’єднаного циклу (DCTON). Ця програма дає можливість визначити особливості фігури, підібрати зміни моделі під неї, визначити колір шкіри і виконати повну композицію. Результати впровадження свідчать про високі результати роботи даної програми.

Рисунок 1.9 – Підбір тканини футболки.

Джерело [30]

Також існують програми , що полегшують підібрати одяг у статті [31], за допомогою датчика Microsoft Kinect V2 для отримання параметрів тіла за допомогою Ог, От, Ос, щоб створити унікальну персональну модель. Можна віртуально накласти 3D-одяг на користувача в режимі реального часу. Програма вміє розпізнавати стать людини. А також композиція одягу доповнюється фізичними рухами, подібними до реальних рухів користувача у цьому вбранні.

Рисунок 1.10 – Головна сторінка Kinect V2.

Джерело [31]

Оскільки все навколо розвивається, то віртуальні простори стають популярними. Завдяки такому розвитку наше життя стає комфортнішим і цікавішим, так Ласіцина Д. [32] пояснює появу віртуального простору для модних показів колекцій, що є дуже зручним і неординарним способом представлення дизайнера.

Підсумовуючи всі переглянуті дослідження можна сказати що прогрес у віртуальних технологіях є дуже високим і те, що маємо зараз є дуже якісними і достатньо точними програмами і системами. Але на даний момент вони тільки починають активний розвиток і ще не все ідеально вдається врахувати. Найбільше науковці приділяли увагу віртуальним вимірам тіла людини. І це логічно, адже для тривимірного проектування одягу, як і для будь якого іншого методу, важливі точні антропометричні дані споживача.

Аналізуючи досліджених авторів, можна сказати, що ніхто з них докладно не зупинявся на аналізі програм тривимірного проектування, а тільки оглядово. Не було згадок про розроблення одягу відштовхуючись від одної конкретної моделі і застосування різних функцій до них, тому тема магістерської роботи буде актуальною і можливо, буде основою чи поштовхом для створення ІТ осередку легкої промисловості в Сумському регіоні.

1.3 Результат дослідження аналогів

Не зважаючи на якісний дизайн та можливість створювати віртуальні моделі дані інформаційні системи не забезпечують як виробника швидким створенням лекал і пошивом індивідуального одягу з врахуванням особливостей фігури, так і вимог споживача в отримані реального продукту.

Провівши детальний аналіз існуючих ІТ в галузі легкої промисловості, нами було зроблено висновок, що існує недостатньо мобільних додатків для швидкої побудови лекал. Тому в роботі [33] описано створений нами мобільний додаток для конструювання лекал жіночого одягу. Наступним кроком стало створення мобільного додатку з вивчення техніки шиття для виготовлення хутряного та шкіряного одягу, який пройшов тестування серед дизайнерів одягу України і має відчутну конкурентоспроможність [34].

Робота [35] присвячена методиці вибору оптимального обладнання, класу фахівця, типу тканини і особливостей пошиву плечового одягу за допомогою нелінійного графу. Далі у роботі [36] наведено короткий опис всіх створених нами мобільних додатків. Але оскільки кожен з них функціонує окремо, то виникла необхідність у створенні інформаційної системи, яка буде об’єднувати їх воєдино.

Задля візуального представлення та виявлення необхідних параметрів створимо порівняльну таблицю продуктів, які були розглянути в попередніх підрозділах.

Таблиця 1.1 – Порівняльна характеристика продуктів - аналогів

Критерії	Dress-x	Tailornova	Clo 3D	Marvelous designer
Зручність у використанні	+	+	+	+
Якість дизайну	+	+	+	-
Повнота інформації про послуги	+	+	-	-
Модуль для подання запиту	+	+	+	+
Модуль для написання повідомлення ательє	+	+	-	+
Спроможність набуття товару, чи послуги	+	+	-	-
Фактичність особистого кабінету користувача	+	+	+	+
Можливість обрати послуги	+	+	-	-
Можливість введення розмірних ознак	+	+	-	+
Можливість отримання реального продукту	-	+	-	-
Адаптивність	+	+	+	+

Джерело: побудовано автором

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Головною ідеєю написання магістерської роботи була реалізація інформаційної системи для спрощення роботи ательє та підприємств легкої промисловості по виготовленню одягу масового використання за допомогою оцифрування, а також можливості клієнта самостійно обирати бажану модель за рахунок аналізу її 3D зображення.

Інформаційна система повинна бути представлена як відкрита платформа-сайт, який повинен бути розміщений в мережі Інтернет.

Кінцева реалізація інформаційної системи повинна відповідати наступним вимогам: швидка побудова лекал одягу майстром за розмірними ознаками фігури клієнта, візуалізація бажаної моделі одягу, збереження даних заказу клієнта, наповнення портфоліо.

Система повинна бути доступною в мережі Інтернет та мати чітку структуру з необхідними функціональними можливостями.

Для можливості використання інформаційної системи необхідно знати найважливіші складові функціонування інформаційної системи та мати достатні навички користування персональним комп'ютером та web-браузерами.

Стилізація сайту повинна відповідати вимогам сьогодення та ресурсній оптимізації, що забезпечує потреби як промисловців текстильної галузі, так і пересічного споживача.

Інформаційна система буде мати широкий спектр фото високої якості.

Розділи інформаційної системи:

Головна сторінка –інформація про сервіс;

Товари – інформація про товари;

Контакти –інформація про контакти ательє;

Мій аккаунт – інформація про розмірні ознаки фігури та виконані замовлення з фото звітом;

Кошик –інформація про товар замовлення.

Для користування сайтом у клієнта повинна бути можливість використовувати Web-браузери Chrome10 і вище, Firefox 3.5 і вище або Safari 3.2.

Функціональні вимоги інформаційної системи:

- можливість проглянути інформацію про ательє;
- можливість отримання інформації про його послуги;
- можливість перегляду фото та відео пальто;
- можливість виконати замовлення;
- можливість перегляду контактів ательє;
- можливість калькуляції лекала замовлення за власними розмірними ознаками;
- фідбек від клієнтів;
- можливість формування портфоліо виконаних робіт.

Даний сервіс полегшить можливість виконати правильний вибір клієнта, та підвищить кількість замовлень для ательє або підприємства масового виробництва одягу. Важливим є те, що всі виконані роботи будуть зберігатися на сайті і спрощуватимуть можливість обрання бажаного товару наступними клієнтами, а також швидка калькуляція лекал для майстрів підприємств для створення лекал.

2.2 Розробка візуальної частини інформаційної системи

Невід’ємною складовою створення інформаційної системи є - процес візуалізації. Для його реалізації було використано HTML, Java Script, SCSS. Кожна з яких відповідає за певний етап і не може функціонцвати окремо. Наприклад, фундаментом створення кожної сторінки є HTML. SCSS відповідає за стиль та структуру. JavaScript – це мова програмування для обчислення, маніпулюляцій та перевірки даних. А спільна робота і взаємодія HTML, Java Script, SCSS забезпечує повноцінну життєдіяльність інформаційної системи. Для обрання фреймворку було враховано найпопулярніші у 2023-2023 роках: React, Angular, Vue, Svelte, Spring [37] тощо.

Проведемо порівняння сильних сторін (табл. 2.1)

Таблиця 2.1 – Порівняння сильних сторін Angular та Vue.js.

№	**Angular**	**Vue.js**
1	Модульність.	Код JavaScript, шаблони та код CSS не змішуються.
2	Компоненти з модулями.	Можливість використання ES5, ES6 і TypeScript.
3	Ін'єкція залежності.	Дуже добре взаємодіє з іншими бібліотеками та проектами.
4	Значний каталог готових компонентів.	Прогресивна розробка програм.
5	Високий рівень контролю.	Документація проста та добре структурована.

Продовження табл.2.1

№	**Angular**	**Vue.js**
6	Дозволяє нормально використовувати HTML і CSS з усіма їхніми функціями та перевагами.	Одночасно складна наскрізна структура та рівень перегляду, який забезпечує керування станом.
7	Найкращий інструмент командного рядка в екосистемі JavaScript.	Підтримує як JavaScript, так і TypeScript.
8	Придатна для великої кодової бази корпоративних програм.	Бібліотека завантажується та встановлюється надзвичайно швидко, що позитивно впливає на швидкість веб-сайту, а отже, на UX та SEO.
9	Вбудовані функції, як очищення DOM, сприяють найбезпечнішим інтерфейсним фреймворком.	
10	Покращений час запуску та час відгуку та зменшений розмір пакета.	

Джерело: побудовано автором

Далі наведемо слабкі сторони (табл. 2.2.)

Таблиця 2.2 – Порівняння слабких сторін Angular та Vue.js.

№	**Angular**	**Vue.js**
1	Початок роботи може бути складним, особливо без глибоких знань про Typescript.	Не має підтримки та фінансової підтримки жодного з основних технічних гравців.
2	Нові ітерації історично значно відрізнялися від попередніх версій, що ускладнює передбачувану міграцію додатків до останньої версії Angular у минулому.	Велика частина вмісту, описів плагінів, посібників щодо фреймворку китайською мовою.
3	Введення залежностей займає багато часу, а створення залежностей для компонентів може бути складним.	Складність реактивності. Система реактивності відтворює лише пакети даних, ініційовані активацією компонента користувачем. Під час читання цих даних часто допускаються помилки, які вимагають скорочення даних.
4		Менша екосистема плагінів та інструментів.

Джерело: побудовано автором

Перерахуємо переваги та недоліки Spring Framework, який обрано для створення інформаційної системи (табл. 2.3.).

Таблиця 2.3. – Аналіз Spring Framework.

№	**Переваги**	**Недоліки**
1	Гнучкість і модульність.	Трудомістке налаштування.
2	Аспектно-орієнтоване програмування.	Складне керування за великої кількості бібліотек.
3	Уніфікований спосіб доступу до різних джерел даних.	Проблеми інтеграції.
4	Уніфікований спосіб доступу до різних джерел даних.	
5	Безпека та аутентифікація.	
6	Невелика вага та архітектурна гнучкість.	
7	Сильна підтримка громади.	
8	Може використовуватися в різних областях.	
9	Зменшення повторюваного коду.	
10	Спрощене тестування.	

Джерело: побудовано автором

За переважною більшістю позитивних сторін були обрані фреймворки для web-додатків Angular та Spring.

Для управління базою даних обрано PostgreSQL.

PostgreSQL дає можливість безпечно зберігати різноманітні типи даних, підтримує стандарти SQL92 і SQL99 і має низку власних розширень, що дозволяє налаштувати його на будь-якій UNIX-сумісній платформі.

3. МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

3.1. Структура інформаційної системи

Вимоги забезпечення працездатності інформаційної системи структуровані у наступні категорії (табл3.1): забезпечення візуалізації, контенту та структури.

Таблиця 3.1 – Вимоги забезпечення працездатності інформаційної системи

№	Категорія			
1	Структура			
	Зрозуміле основне наповнення	Навігація	Вкладеність каталогів	
2	Візуалізація			
	Чіткість тексту	Кольорова гама	Адаптивність	Стилістика
3	Контент			
	Інформативність	Релевантність	Оригінальність	Цікавість

Джерело: побудовано автором

Інформаційна система має відмінні функціональні модулі для зареєстрованих і незареєстрованих користувачів. Порівняльний аналіз подано нижче у табл. 3.2.

Таблиця 3.2 – Порівняння доступних модулів зареєстрованих і незареєстрованих користувачів.

№	До авторизації		Після авторизації	
1	Головна сторінка	Містить основну інформацію про інформаційну систему.	Особисті розмірні ознаки	Можливість вносити дані про особисті розмірні ознаки
2	Авторизація	Авторизація вже зареєстрованих користувачів	Мої замовлення	Перегляд виконаних замовлень
3	Реєстрація	Реєстрація нових користувачів.	Калькулятор лекал	Калькуляція лекал за розмірними ознаками клієнта.
4			Редагування особистої інформації	Можливість вносити зміни, які виникають з часом.

Джерело: побудовано автором

3.2. Структурно-функціональне моделювання процесу

Для структурно-функціонального моделювання інформаційної системи були створені діаграми IDEF0.

Основна мета IDEF0 є заздалегідь обдумане ділення загального процесу інформаційної системи на окремі його підпроцеси [37]. На діаграмі показано враховування всіх можливих зв’язків між усіма підпроцесами, щоб унеможливити появу помилок роботи інформаційної системи та виправити їх [38]. Як відомо IDEF0 характеризується наявністю розміщення та використання стрілок, як звичайних (входів/виходів), так і стрілок-елементів управління.

Розроблена діаграма IDEF0 та її декомпозиція наведені нижче (рис. 3.1-3.2).

Рисунок 3.1 – IDEF0 інформаційної системи

Джерело: побудовано автором

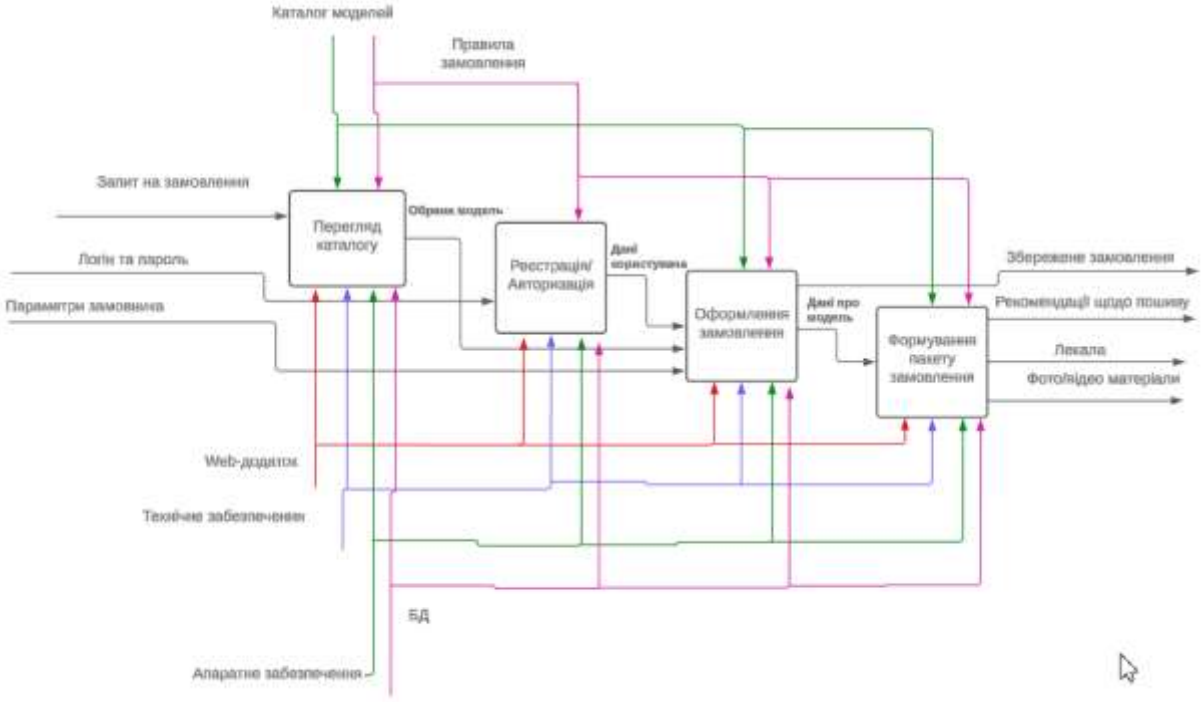


Рисунок 3.2 – Декомпозиція діаграми IDEF0

Джерело: побудовано автором

3.3. Моделювання варіантів використання

Для унаочнення функціоналу інформаційної системи втілено діаграму варіантів використання [16, 18].

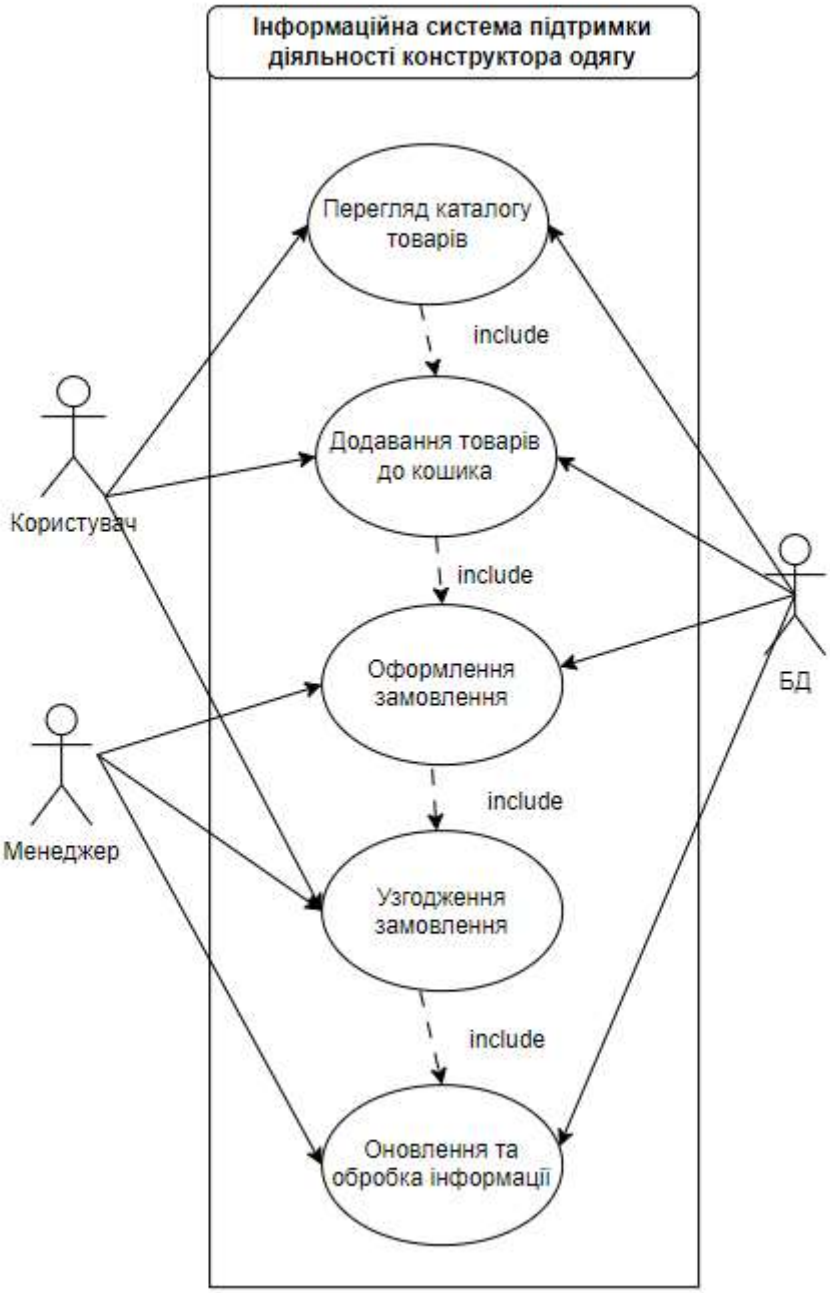


Рисунок 3.3 – Діаграма варіантів використання

Джерело: побудовано автором

Деталізація варіантів використання інформаційної системи представлена в табл.3.3 та табл.3.4.

Таблиця 3.3. - Опис акторів

№	Іменування актора	Детальний опис
1	Представник ательє/Менеджер	Формує пропозиції для оформлення договорів. Редагує та видаляє дані на сайті, відповідає на запитання.
2	Користувач-замовник	Відправляє запити на сформований пост представником ательє для підписання договору.

Джерело: побудовано автором

Таблиця 3.4. - Варіанти використання

№	Варіант використання	Детальний опис	Іменування актора
1	Авторизація	Дає можливість зареєстрованому користувачеві авторизуватися в системі.	Всі користувачі
2	Реєстрація	Дає можливість зареєструватися новому користувачеві.	Всі користувачі
3	Редагування інформації на сайті	Дає можливість адміністратору редагувати та змінювати загальну інформацію на сторінках інформаційної системи.	Представник ательє
4	Перегляд пропозицій	Дає можливість переглядати інформацію про товари.	Всі користувачі
5	Редагування створеної пропозиції	Дає можливість представнику ательє редагувати власні пропозиції.	Представник ательє

Продовження табл. 3.

№	Варіант використання	Детальний опис	Іменування актора
6	Відповідь на запит від користувача-замовника	Дає можливість представнику ательє відповідати на запит користувача до певної пропозиції, формує запит для обговорення умов договору для подальшої співпраці із користувачем-замовником.	Представник ательє
7	Запит на створену пропозицію від представника ательє	Дає можливість користувачу- замовнику відправляти запит на створену пропозицію від представника ательє.	Користувач-замовник
8	Пошук та фільтрації пропозицій	Дає можливість виконувати фільтрацію пропозицій тематики за тегами.	Всі користувачі
9	Блокування користувачів	Дає можливість адміністратору заблокувати користувачу.	Представник ательє
10	Редагування особистих даних	Дає можливість редагувати особисту інформацію в інформаційній системі, наприклад: змінювати контактну інформацію, розмірні ознаки, оновлювати фото пропозицій та виконаних робіт, редагувати інформацію про ательє.	Всі користувачі
11	Формування запитання адміністрації	Дає можливість користувачеві інформаційної системи відправляти адміністраторам запитання.	Представник ательє, користувач-замовник

Джерело: побудовано автором

3.4. Проектування моделі бази даних інформаційної системи

На етапі проектування інформаційної системи була розроблена модель бази даних, розробленої з використанням СУБД Postgre SQL.

Для бази даних були виділені наступні сутності:

category – містить інформацію про види товарів;

product – запам`ятовує дані про товари;

order – містить дані про замовлення;

client – запам`ятовує дані про клієнтів;

У табл.3.5 наведена інформація про атрибути до відповідних сутностей.

Таблиця 3.5 – Інформація про атрибути та сутності

Назва сутності	Назва атрибуту	Тип атрибуту	Ключі	Опис атрибуту
category	id_category	INT	PK	Унікальний ідентифікатор категорії
	name	VARCHAR(50)	-	Назва категорії
	description	VARCHAR(250)	-	Опис категорії
product	id_product	INT	PK	Унікальний ідентифікатор товару
	id_category	INT	FK1	Унікальний ідентифікатор категорії
	name	VARCHAR(250)	-	Назва товару
	description	VARCHAR(250)	-	Опис товару
	photo	VARCHAR(250)	-	Фото товару
service	id_ service	INT	PK	Унікальний ідентифікатор послуги
	id_category	INT	FK1	Унікальний ідентифікатор для фото

Продовження таблиці 3.5 – Інформація про атрибути та сутності

Назва сутності	Назва атрибуту	Тип атрибуту	Ключі	Опис атрибуту
order	id_order	INT	PK	Унікальний ідентифікатор замовлення
	id_list_p	INT	FK1	Унікальний ідентифікатор списку товарів
	id_client	INT	FK3	Унікальний ідентифікатор клієнта
	status	ENUM («прийнято», «в роботі», «відправелно», «завершено»)	-	Статус замовлення
client	id_client	INT	PK	Унікальний ідентифікатор клієнта
	name	VARCHAR(250)	-	Ім'я клієнта
	phone	VARCHAR(12)	-	Номер телефону
	login	VARCHAR(50)	-	Логін
	password	VARCHAR(50)	-	Пароль
the list of products	id_list_p	INT	PK	Унікальний ідентифікатор списку товарів
	id_product	INT	FK1	Унікальний ідентифікатор товару
	count	INT	-	Кількість товарів

Продовження таблиці 3.5 – Інформація про атрибути та сутності

Назва сутності	Назва атрибуту	Тип атрибуту	Ключі	Опис атрибуту
the list of services	id_ list_s	INT	PK	Унікальний ідентифікатор списку послуг
	id_ service	INT	FK1	Унікальний ідентифікатор послуги
	count	INT	-	Кількість послуг

Джерело: побудовано автором

За допомогою сформованих сутностей та атрибутів було побудовано ER-діаграма, яка зображена на рис. 3.4. Дана діаграма дозволяє наочно побачити , які саме звя`зки встановлені між сутностями та що вони з себе являють [38].

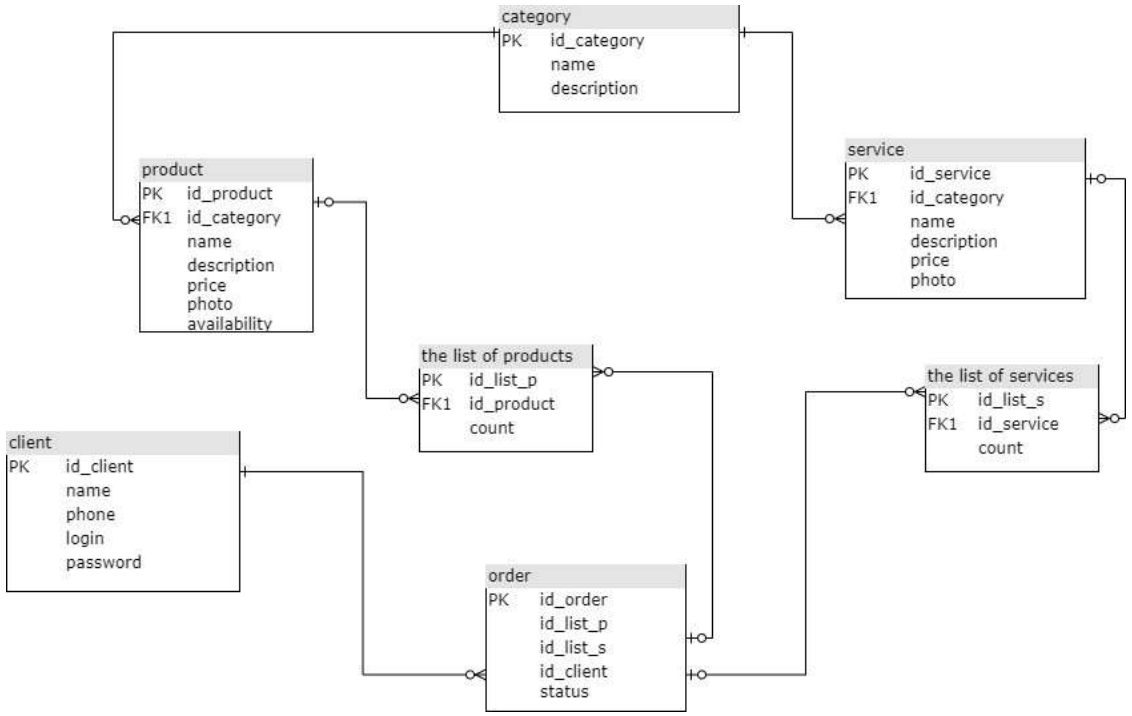


Рисунок 3.4 – ER діаграма

Джерело: побудовано автором

4. ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1. Програмна реалізація

Програма інформаційної системи обслуговування роботи ательє складається з двох частин:

- Клієнт (візуальна частина) – створено за допомогою технології Angular (HTML / TypeScript / SCSS)
- Сервер (функціональна частина) – створено за допомогою мови Java, а саме технології Spring Boot

Структура проєкту в Angular:

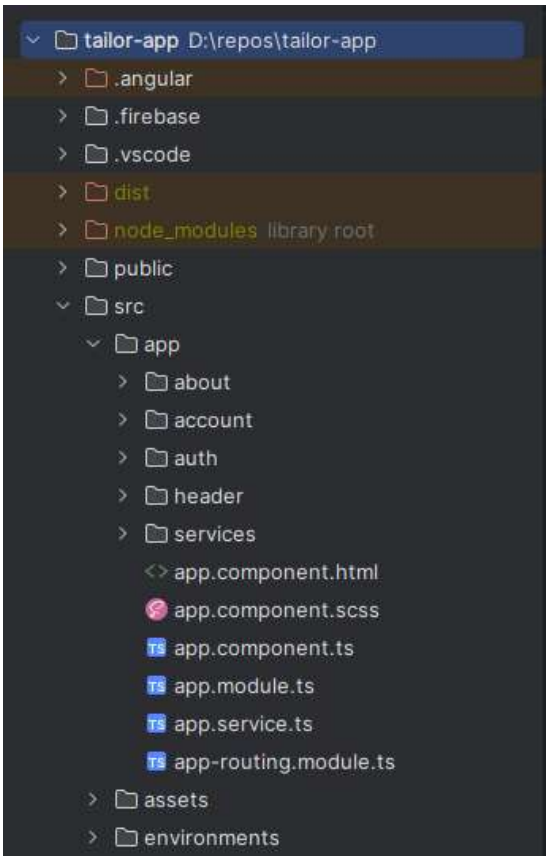


Рисунок 4.1 - Структура проєкту в Angular

Джерело: побудовано автором

Проєкт складається з наступних пакетів:

Таблиці 4.1 – Пакети в Angular

Назва	Опис
app	Містить всі компоненти аплікації
app.about	Компонент “Про нас”
app.account	Відповідає за рівні доступу
app.auth	Відповідає за аутентифікацію, активацію та реєстрацію акаунту та діалогові вікна
app.header	Шапка сторінки
app.services	Каталог та калькулятор,
assets	Всі картинки

Джерело: побудовано автором

Структура проєкту в Spring Boot:

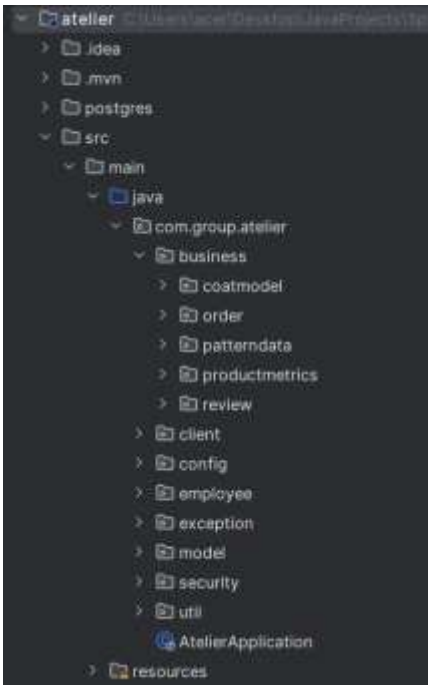


Рисунок 4.2 - Структура проєкту в Spring Boot

Джерело: побудовано автором

Проект складається з наступних пакетів:

Таблиці 4.2 – Пакети в Spring Boot

Назва	Опис
business	Комплекс пакетів, який відповідає за бізнес логіку в додатку
business.coatmodel	Моделі пальто
business.order	Замовлення
business.patterndata	Калькулятор параметрів лекал
business.productmetrics	Вхідні параметри виробів
business.review	Відгуки
client	Реєстрація, особові дані і параметри клієнта
config	Технічна конфігурація додатку
employee	Обслуговування даних працівників ательє
exception	Відповідає за обслуговування винятків у додатку
model	Комплекс класів, який відображає структуру бази даних
security	Механізми аутентифікації і авторизації в додатку
util	Пакет допоміжних класів
resources	Статичні ресурси, такі як конфігурація, картинки і текстові файли

Джерело: побудовано автором

Таким чином ми підготували структуру проекту із всіма потрібними шаблонами, файлами та налаштуваннями.

4.2. Структура користування інформаційною системою користувачем

Для використання інформаційної системи необхідно скористатися сторінкою a-telie.web.app. На головній сторінці, на яку потрапляє користувач (рис.4.3), розміщена інформація про послуги, які надає ательє, його адреса та контакти. Ця сторінка є головною як для зареєстрованих, так і незареєстрованих користувачів.

Рисунок 4.3 – Головна сторінка.

Джерело: побудовано автором

Також у верхній шпальті доступною є можливість переходу на сторінку перегляду каталогу моделей ательє рисунок 4.4. та введення власних розмірних ознак у калькулятор рисунок 4.5.

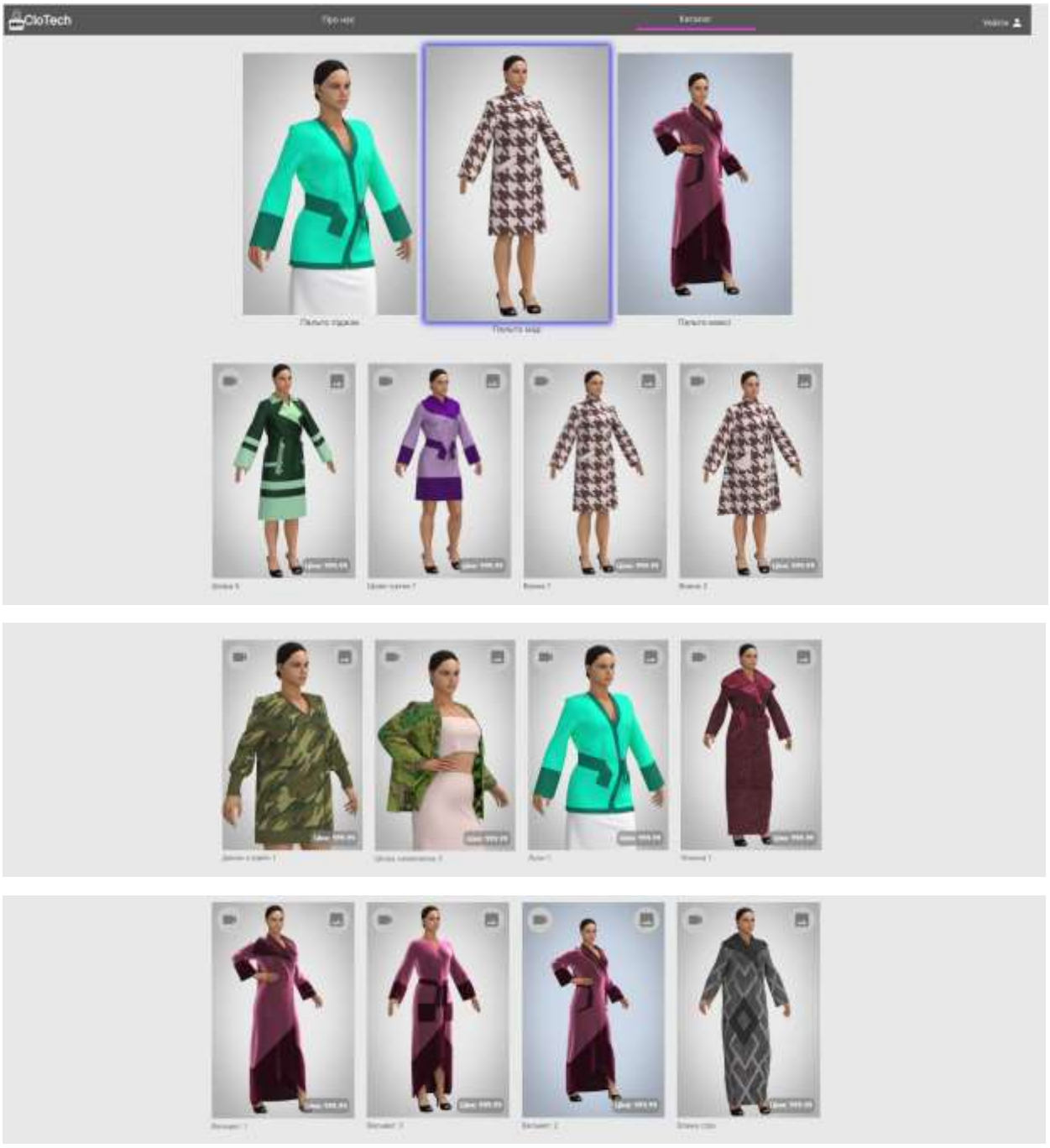


Рисунок 4.4. – Каталог з підкаталогами моделей.

Джерело: побудовано автором

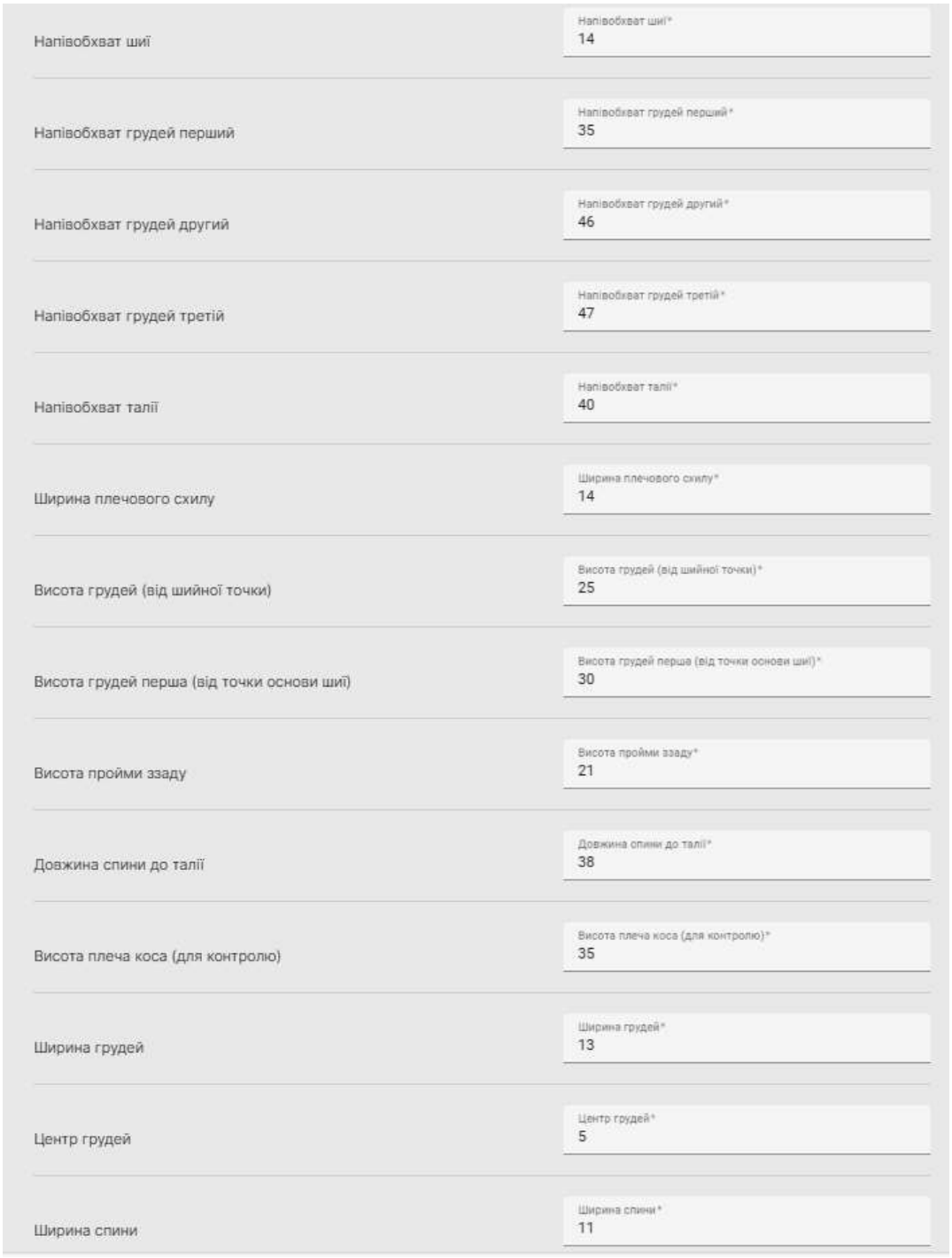


Рисунок 4.5. – Вхідні дані для роботи калькулятора.

Джерело: побудовано автором

Після цього можливості незареєстрованих користувачів завершуються, а зареєстровані можуть формувати далі замовлення.

З даної сторінки можна виконати перехід на сторінки реєстрації та авторизації у верхній шпальті сторінки.

4.3. Процес реєстрації та створення кабінету користувача

На сторінці реєстрації користувач повинен заповнити такі поля як:

– Ім’я;
– Прізвище;
– Електронна пошта;
– Пароль.

Заповнивши всі поля без виключення користувач повинен натиснути кнопку «Зареєструватися» (рис. 4.6).

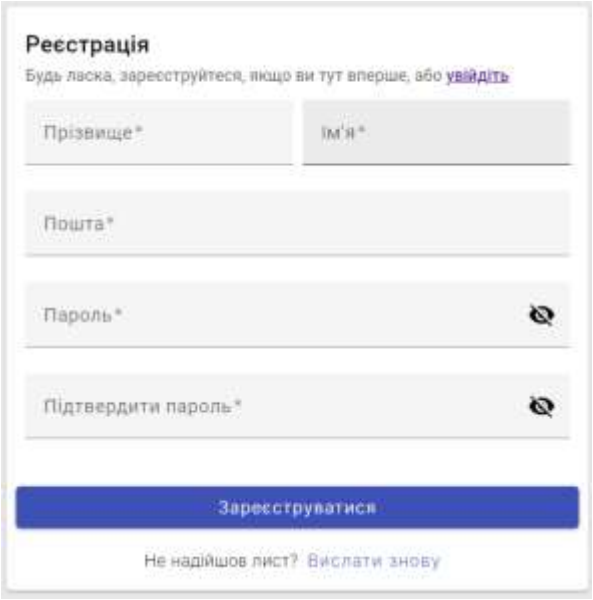


Рисунок 4.6 – Сторінка реєстрації

Джерело: побудовано автором

Після реєстрації користувач отримує листа на електронну адресу про необхідність завершення реєстрації та створення акаунта (рис.4.7)

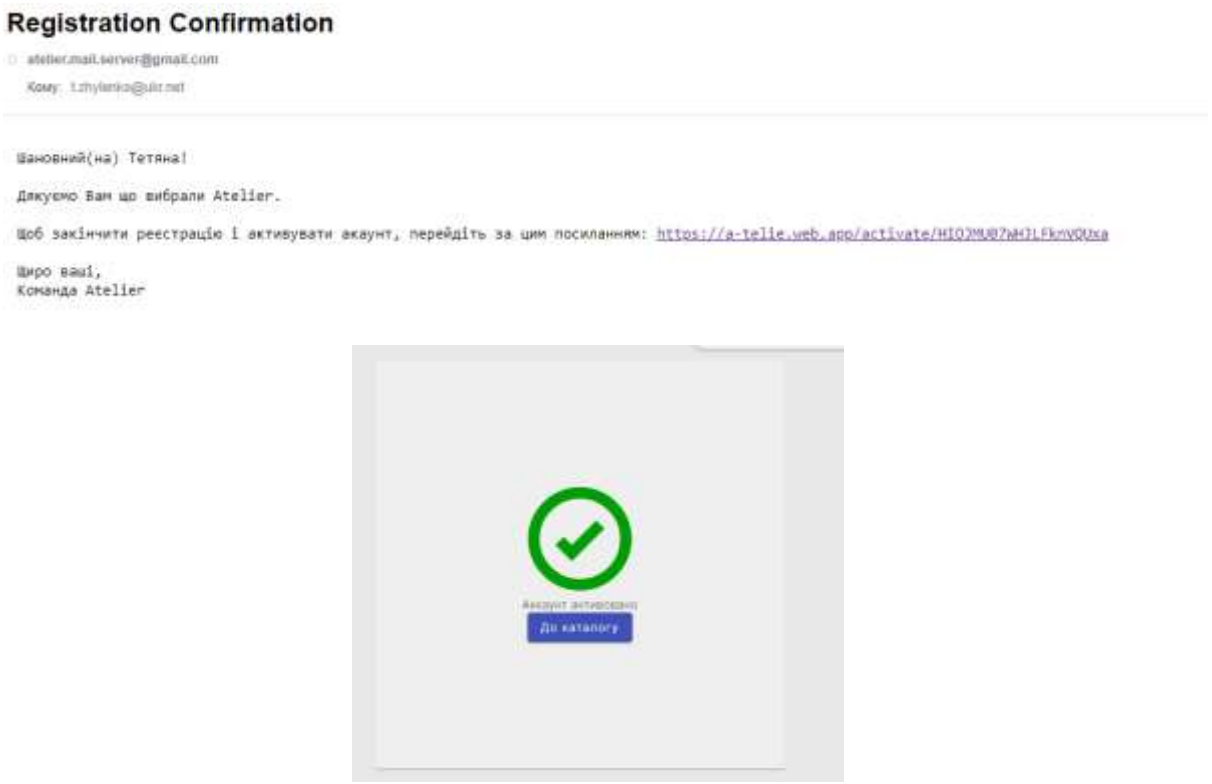


Рисунок 4.7 – Підтвердження реєстрації

Джерело: побудовано автором

Далі користувач перенаправляється на сторінки авторизації (рис.4.7), де він повинен ввести електрону пошту та пароль, що були введені під час реєстрації (рисунок 4. 8).

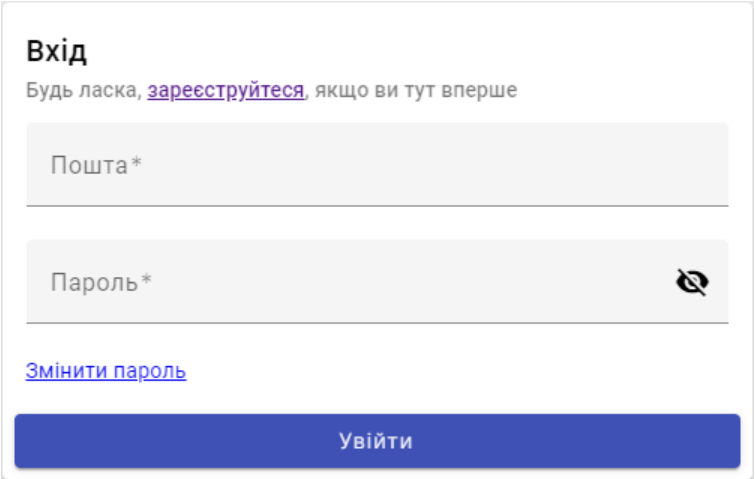


Рисунок 4.8 – Сторінка авторизації

Джерело: побудовано автором

На верхньому шпальті з’являється додаткове віконце Акаунт, натискаючи на який клієнт переглядає моделі у каталозі (рис. 4.9).

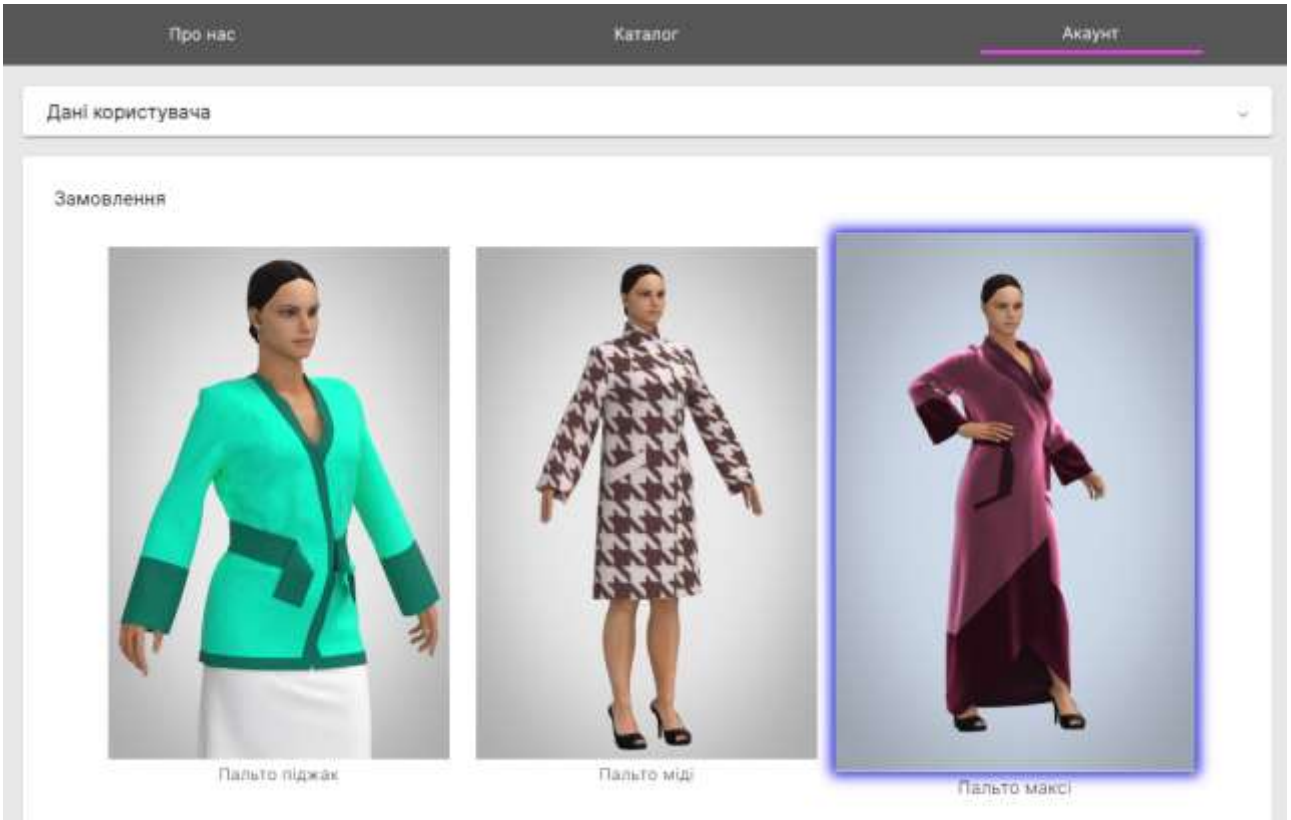


Рисунок 4.9 Моделі каталогу.

Джерело: побудовано автором

Далі користувачеві відкриваються певні можливості:

Вводити особисті дані (рис.4.10).

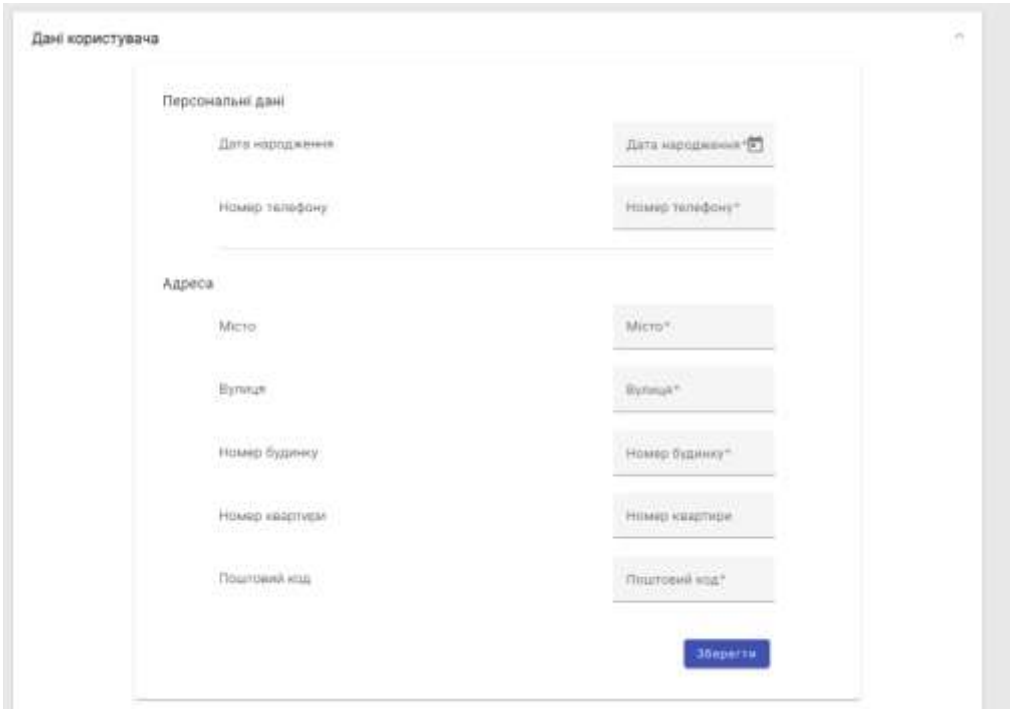


Рисунок 4.10 - Дані користувача.

Джерело: побудовано автором

– зберігати свої розмірні ознаки (рис.4.11),

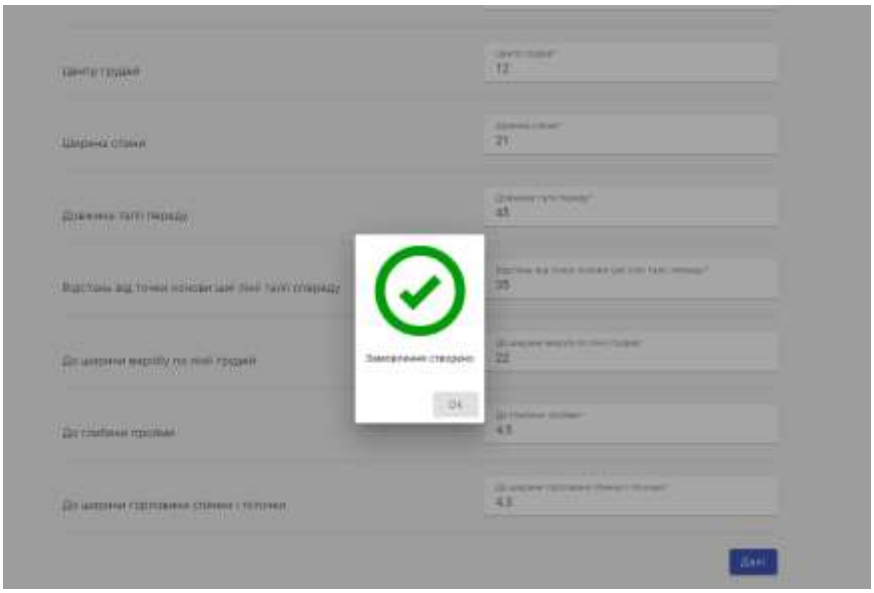


Рисунок 4.11 – Збереження розмірних ознак.

Джерело: побудовано автором

– додавати модель до списку замовлень (рис. 4. 12),

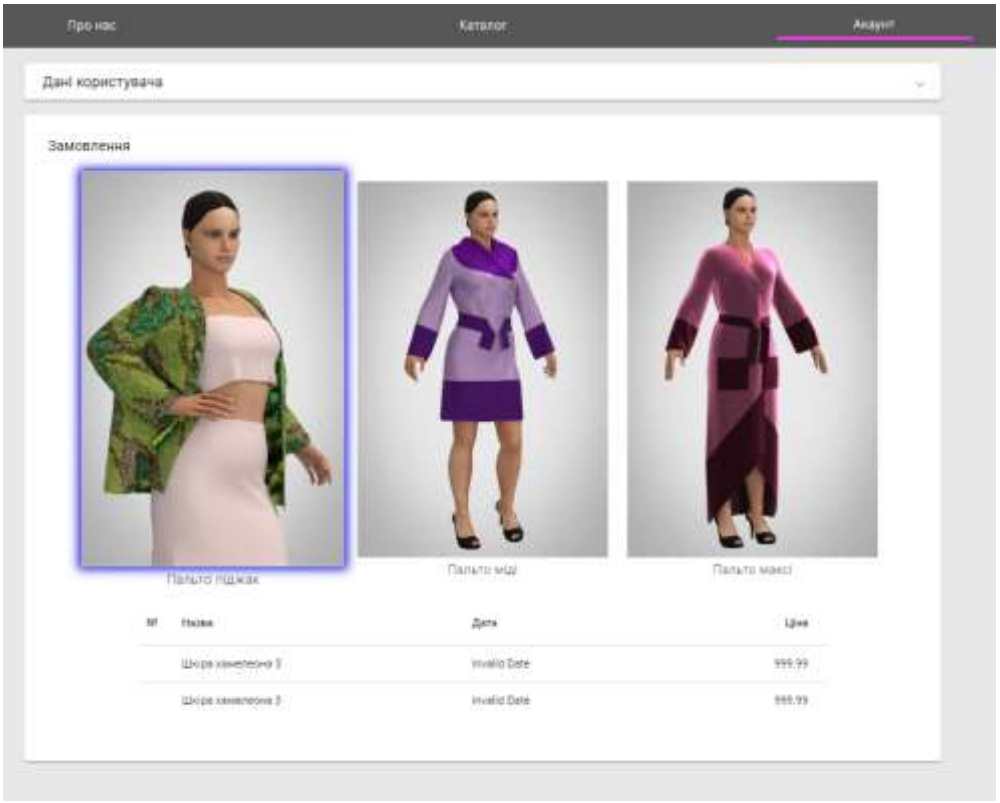


Рисунок 4.12 – Додавання нової моделі.

Джерело: побудовано автором

– відстежувати стан замовлення,

При натисканні на модель розкривається збільшене її фото, вхідні та вихідні дані для створення лекала (рис. 4. 13).

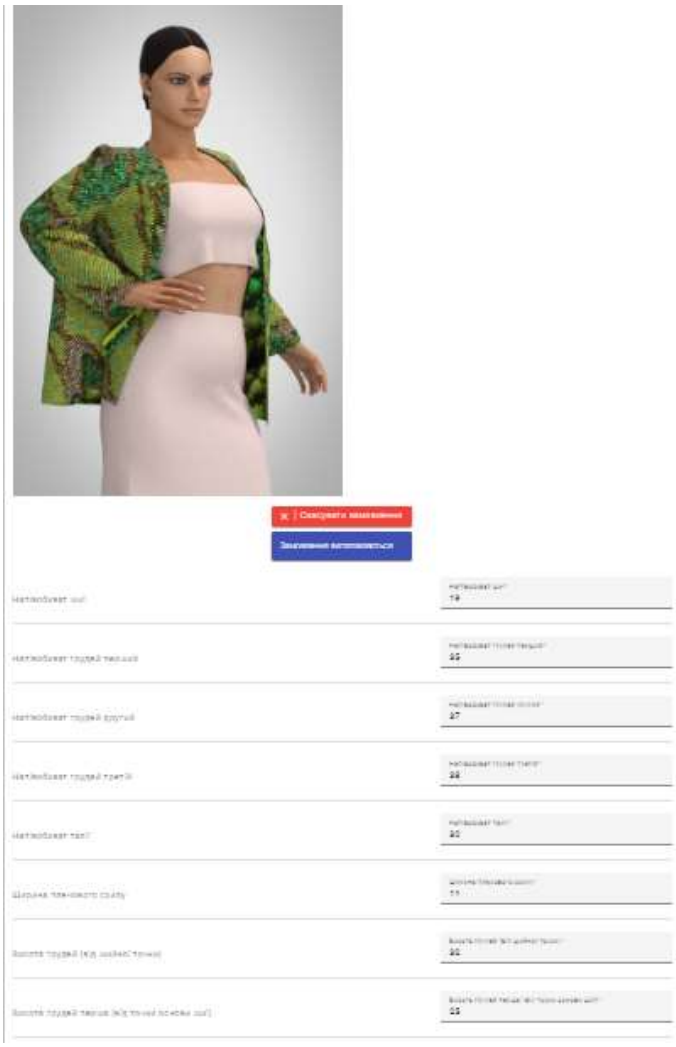


Рисунок 4.13 – Вигляд моделі замовлення.

Джерело: побудовано автором

– переглядати історію своїх попередніх замовлень (рис. 4. 14).

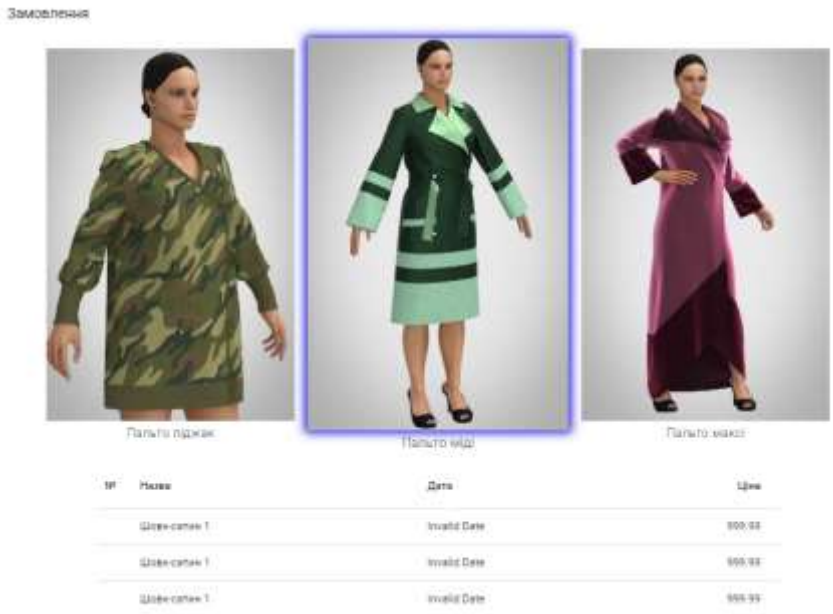


Рисунок 4.12 – Таблиця замовлень.

Джерело: побудовано автором

Виконаємо порівняльний аналіз переваг можливостей зареєстрованих і незареєстрованих користувачів (Таблиця 4.3)

Таблиця 4.3 - Можливості зареєстрованих і незареєстрованих користувачів

№	Незареєстрований користувач	Зареєстрований користувач
1	Перегляд каталогу моделей	
2	Використання калькулятора розмірних ознак	
3		зберігати свої розмірні ознаки
4		додавати модель до списку замовлень
5		відстежувати стан замовлення
6		переглядати історію своїх попередніх замовлень
		Зв`язок з працівниками ательє

Джерело: побудовано автором

4.4. Підтвердження замовлення або його відміна

Після того, як замовлення було створене, у клієнта з`являється можливість відслідкувати його стан. У випадку, коли клієнт захоче скасувати замовлення, у нього є така можливість, адміністратор ательє цього зробити не може.

Натискаючи клавішу «Скасувати замовлення», відразу модель зникає з кабінету.

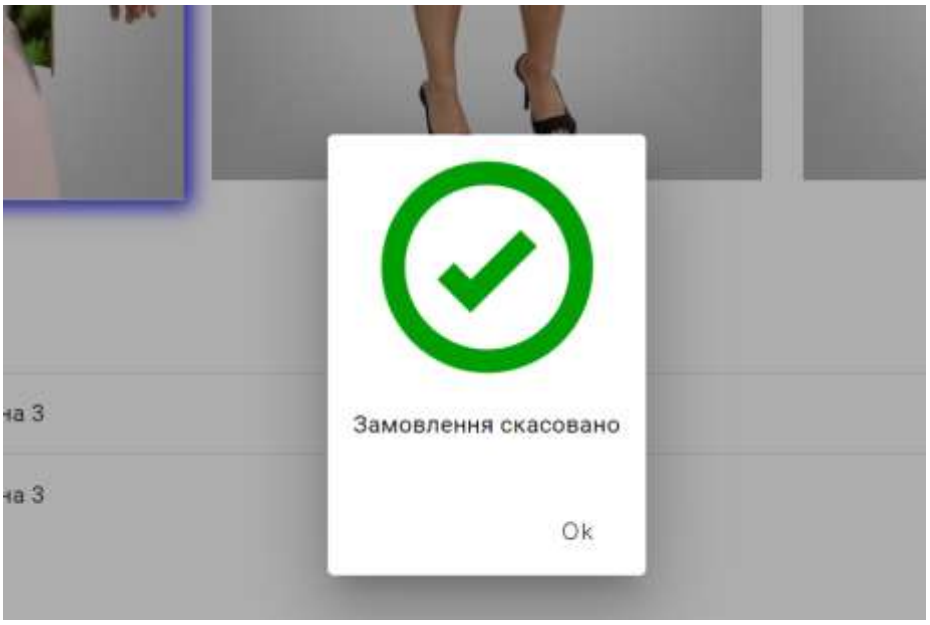


Рисунок 4.15 – Скасування замовлення

Джерело: побудовано автором

4.5. Робота з калькулятором розмірних ознак

Перш ніж замовити пальто, необхідно ввести розмірні ознаки клієнта. Кожна ознака має обмежений діапазон. Якщо введена одиниця знаходиться поза

межами інтервалу допустимих значень, рядочок підсвічується червоним і надається підказка, яких мінімальних та максимальних значень може набувати дана одиниця.

Допоки не будуть введені всі правильні вхідні значення, калькулятор не буде виконувати обчислення. Приклад обмежень наведено на рисунку 4.16.

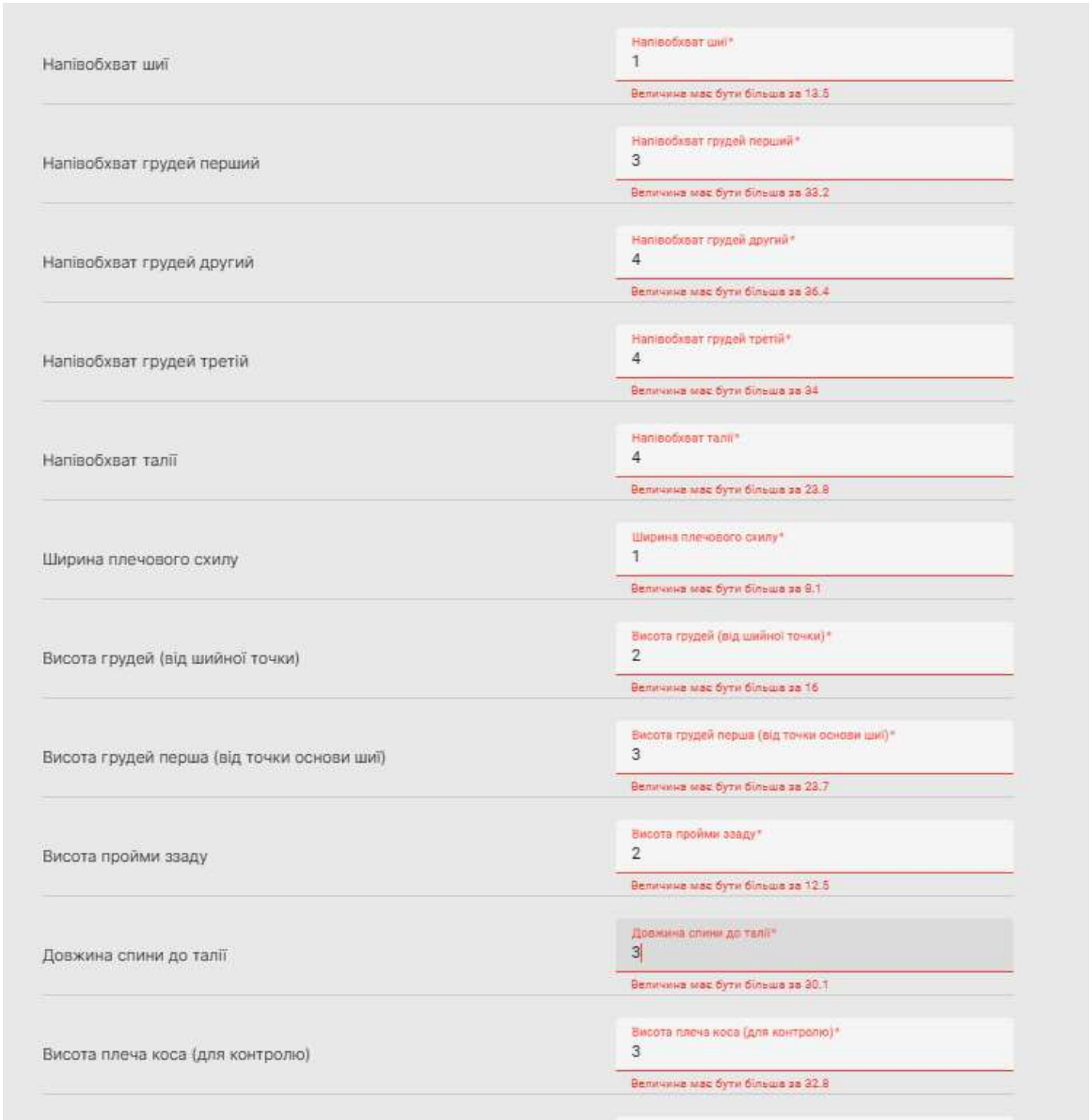


Рис.4.16. – Обмеження вхідних даних розмірних ознак.

Джерело: побудовано автором

4.6. Математична система розрахунку лекал

Наведемо покрокові операції роботи калькулятора розмірних ознак на прикладі моделі з розмірами 170-96-104.

Таблиця 4.4 –Вхідні данні для роботи калькулятора з необхідними позначеннями.

№ п/п	Назва розмірної ознаки	Умовне позначення	Величина, см
1	Напівобхват шиї	$С_ш$	18,6
2	Напівобхват грудей перший	$С_{гI}$	46,1
3	Напівобхват грудей другий	$С_{гII}$	50,5
4	Напівобхват талії	$С_т$	37,4
5	Напівобхват стегон	$С_б$	52,0
6	Висота грудей	$В_г$	35,6
7	Ширина грудей	$Ш_г$	17,5
8	Відстань між сосковими точками	$Ц_г$	10,2
9	Ширина спини	$Ш_с$	18,3
10	Відстань між точками лопаток	$Ц_л$	10,1
11	Довжина талії спини	$Д_{тс}$	41,4
12	Довжина талії спини 1	$Д_{тс1}$	44,5
13	Висота плеча коса	$В_{шк}$	44,5
14	Висота плеча коса(спереду)	$В_{пкп}$	26,0
15	Ширина плечового скату	$Ш_п$	13,6
16	Обхват плеча	$О_п$	30,1
17	Відстань від шийної точки до точки основи шиї збоку(по лінії виміру обхвату шиї)	$Д_{шош}$	9,0
18	Довжина виробу	По моделі	65,0

Джерело: побудовано автором

Таблиця 4.5 – Розрахункові формули для роботи калькулятора конструювання жіночого демісезонного пальто.

№п/п	Конструктивна ділянка на кресленні	Умовне позначення	Розрахунковаформула	Розрахунок	Вихідні дані для калькулятора
1	2	3	4	5	6
Побудова базисної сітки конструкції пальто					
1	Побудувати прямий кут з вершиною в точці А, що відповідає положенню сьомого шийного хребця хребта.				
2	Ширина виробу	Аа1	СгII+Пг	50,4+6,0	56,5
3	Ширина спини	Аа	Шс+Пшс	18,3+2,05	20,35
4	Ширина пілочки	а1а2	0,9Шг+(СгII- СгI)+Пшг	0,9×17,5+(50,5-46,1)+2,74	22,89
5	Положення лінії талії	АТ	$Д_{тс1}+П_{дтс}$	44,5+1,0	45,5
6	Положення лінії стегон	ТБ	16-20 см	19,0	19,0
7	Довжина виробу	ТН	По моделі	65,0	65,0
8	Із точки а1 провести вниз перпендикуляр до лінії Аа1. Із точок Т, Б, Н провести вправо горизонталі до вертикалі із точки а1 і відповідно проставити точки Т1, Б1, Н1.				
Побудова верхніх контурних ліній спинки					
9	Висота горловини	ТА1	Дтс+Пвр	41,4+1,7	43,1
10	Ширина горловини	АА2	$\frac{С_{ш}}{3}$ + ПшрЗ	$18,6_{} + 1,0^{3}$	7,2

Продовження таблиці 4.5

	2	3	4	5	6
11	Провести лінії із точок А1 і А2 вправо і вниз відповідно, так що вони перетинаються у точці А3. Із точки А3 провести бісектрису кута А1А3А2 і поставити точку А4. Горловину оформити плавною лінією через точки А1А4А2.				
12	Визначення скосу плечаспинки	UA2П	Шп+0,4	13,6+0,4	14,0
		ТТ0	2,5 см	2,5	2,5
		UT0П	Впк+1,0 см	44,5+1,0	45,5
13	На перетині дуг А2П і Т0П поставити точку П. А2 і П з'єднати прямою.				
14	Оформлення плечової виточки	А2в	$\underline{Ш_П}$ З	13,6 З	4,6
		вв1	6-9 см	8,5	8,5
		в1в2	0,5 см	0,5	0,5
		Точки в2 і в з'єднати і продовжити вгору на 0,3 см і поставити точку в3. Із в2 вправо провести дугу радіусом, що дорівнює розхилу виточки - 1,5 см і поставити перетин. Із точки в2 провести пряму вгору через зперетин із точки в2 довжиною в2в3 і поставити точку в4. в2в3 = в2в4 А2 з'єднати прямою із в3.			

Продовження таблиці 4.5

15	Нове положення точки П	UA2П2	А2П+розхил виточки	14,0+1,5	15,5
		UT0П2	Впк+1,0 см	44,5+1,0	45,5
		в4 з'єднати із П2.			
16	Пройма спинки	П1Г2	(0,15СгI)+(0,15Шпр)++ Пспр+5,1+Рсут	(0,15×46,1)+(0,15×13,26) ++ 5,1+5,1+1,3	20,47
		Г2П3	П2Г2 З	20,473	6,8
		Через точку Г2 провести горизонталь. На перетині горизонталі з вертикаллю А і а1 поставити точки Г і Г1 відповідно. З точки а2 опустити вертикаль на лінію ГГ1 і поставити точку Г4. Г2Г4 поділити навпілі поставити точку Г3. Ця точка буде вершиною бічного зрізу пілочки і спинки. Провести бісектрису кута П3Г2Г3 довжиною 0,15Шпр + 1,5 = 3,4 см і поставити точку П5. Оформити пройму спинки через точки: П2П3 – з'єднати прямою, П3П4Г3 з'єднати лекальною лінією.		

Продовження таблиці 4.5

	2	3	4	5	6
Побудова верхніх контурних ліній пілочки					
17	Визначення балансу виробу	Г1А3	Вг – Дшош	35,6-9	26,6
		З точки А3 провести вліво горизонталь. З точки А3 зробити перетин вліво довжиною 0,5см, для кращого прилягання виробу до тіла, на місці перетину поставити точку а3. а3 з'єднати із точкою Г1.			
18	Ширина горловини	а3А4	АА2	7,2	7,2
19	Висота горловини	а3А5	0,45Сш	0,45×18,6	8,3
20	Побудова нагрудної виточки	Г1Г5	Цг+0,5	10,2+0,5	10,7
		З Г5 опустити перпендикуляр довжиною 2-3 см.			
		UА4Г6	Вг - Сш – 0,2 3	35,6 - 18,6 – 0,2 3	29,2
		Де UА4Г6 перетинається із перпендикуляром з точки Г5 поставити точку Г6. А4 і Г6з'єднати прямою – це права сторона виточки.			

Продовження таблиці 4.5

		Г6Г7	0,9Шг + $\underline{\text{Сг}II - \text{Сг}I}$ – 1,52	0,9×17,5 + 50,5−46,1 – 1,52	16,45
		Г7Г8	1,5(СгII- СгI)	1,5×(50,5-46,1)	6,6
		З точки Г6 через точку Г8 провести пряму довжиною А4Г6 – це ліва сторона виточки. Г6А6 = А4Г6 = 29,2 см.			
21	Пройма і плечовий край	UА6П5	Шп+0,5	$\underline{13}$,6+0,5	14,1
		UГ6П5	Впкп+0,5	26,0+0,5	26,5
		На перетині цих двох дуг поставити точку П5.			
		Г4П6	0,3Шпр+1,2	0,3×13,~~26+1,2~~	5,1
22	Оформлення пройми	По бісектрисі кута П6Г4Г3 відкласти точку П7: Г4П7 = 2 – 3 см = 2,6 см. Точки А6 і П5 – з'єднати прямою. Точки П5 і П6 з'єднати прямою, поділити відрізок навпіл поставити точку п. До нього відкласти перпендикуляр 0,8 см і поставити точкуп1. Точки П5, п1, П6 – з'єднати дугою. Через точки П6, П7, Г3 провести лекальну лінію.			
Побудова лінії низу, бічних зрізів, лінії борту і середини спинки ?					

Продовження таблиці 4.5

23	Середня лінія спинки	Оскільки спинка без середнього шва, то для оформлення лінії згину середини спинкиз точки А1 провести штрих-пунктирну лінію до точки Н
24	Лінія низу	Проходить від точки Н до точки Н1
25	Лінія борту пілочки	Проходить від точки Н1 до точки А5
26	Лінія бічних зрізів пілочки іспинки	Із точки Г3 опустити вертикаль на лінію низу. Поставити точки Т2, Б2, Н2 на лінії талії,стегон і низу відповідно.

4.7. Кабінет адміністратор

З інформаційною системою працювати повинен окрім клієнтів ще й адміністратор. У верхньому шпальті додаються вкладки: «Замовлення» і «Працівники» (рис.4.17).

Рисунок 4.17 – Головна сторінка сайту з погляду адміністратора.

Джерело: побудовано автором

Якщо перейти на вкладку «Працівники», можна реєструвати нових працівників, також можна відслідковувати всіх працівників ательє (ПІП), інформацію про те, коли вони були зареєстровані та їх поштову адресу. Щоб зареєструвати нового працівника необхідно натиснути «зареєструватися/увійти» і заповнити необхідні поля, як наводилося вище для користувача, ввести прізвище, ім’я, номер телефону, електронну адресу та пароль (рис.4.18). На пошту прийде лист для активації акаунта.

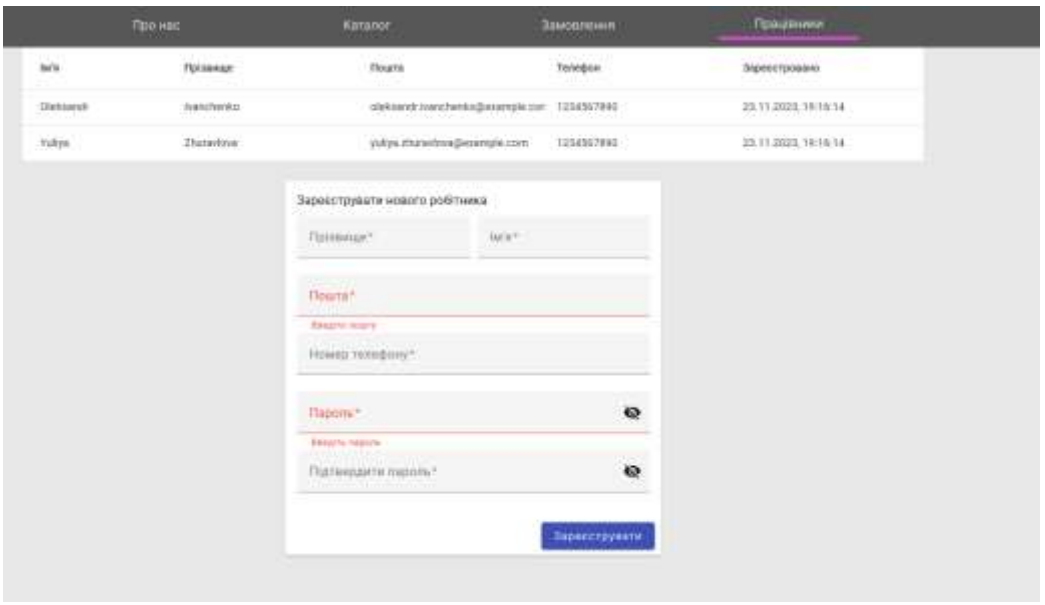


Рисунок 4.18 – Реєстрація нового працівника.

Джерело: побудовано автором

Також адміністратор може видаляти акаунти працівників, які вже не працюють, як показано на рис. 4.19.

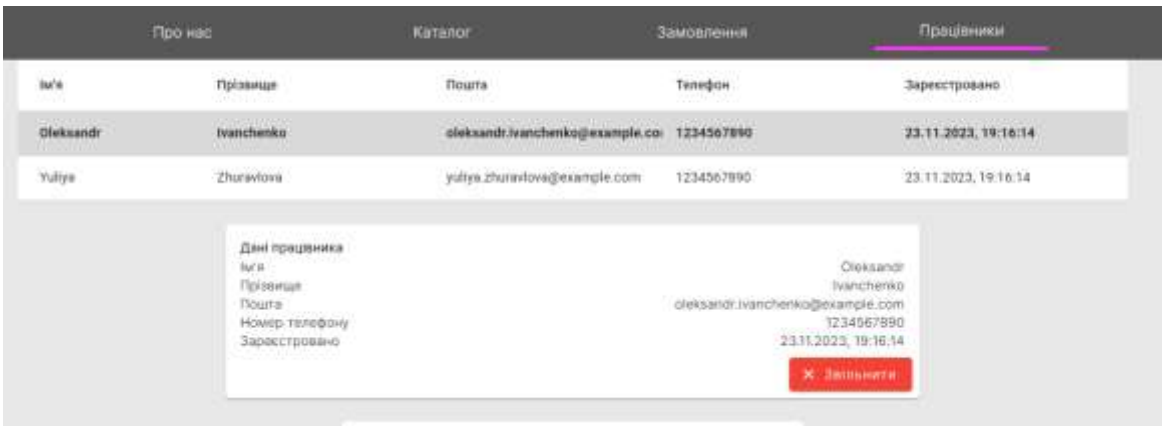


Рисунок 4.19. Видалення акаунта працівника.

Джерело: побудовано автором

Перейшовши на вкладку «Замовлення» можна відсортувати всі замовлення за станом виконання. Таку можливість дає перемикач у лівому верхньому кутку. Можна переглянути всі замовлення, що очікуються, тобто тільки були сформовані користувачами (рис. 4.20).

Адміністратор оперує такою інформацією: коли замовлення було створене, яка його ціна, яка модель та з якого матеріалу буде виготовлятися замовлення.

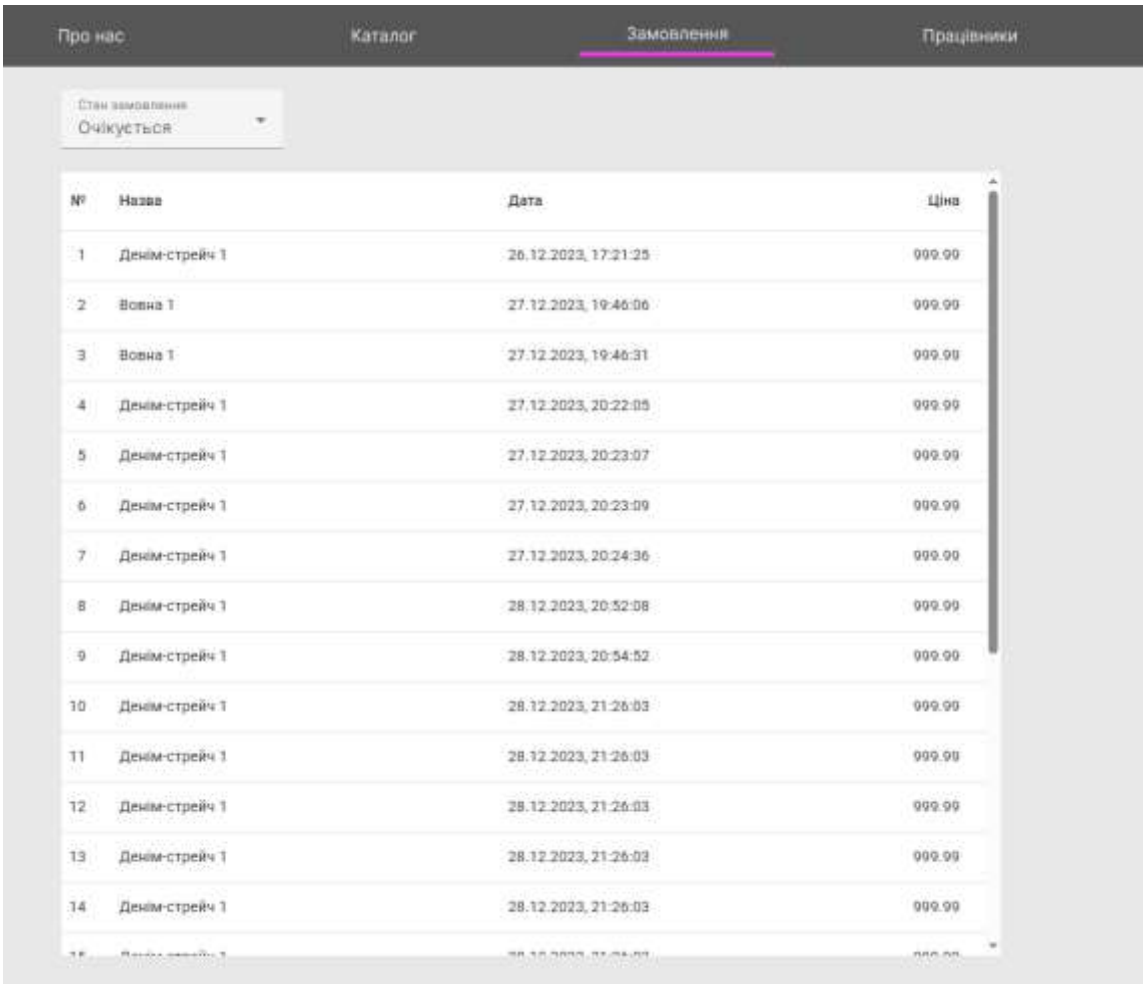


Рисунок 4.20 – Замовлення, які очікуються.

Джерело: побудовано автором

За аналогією можна переглянути замовлення, які знаходяться у процесі виготовлення (рис.4.21).

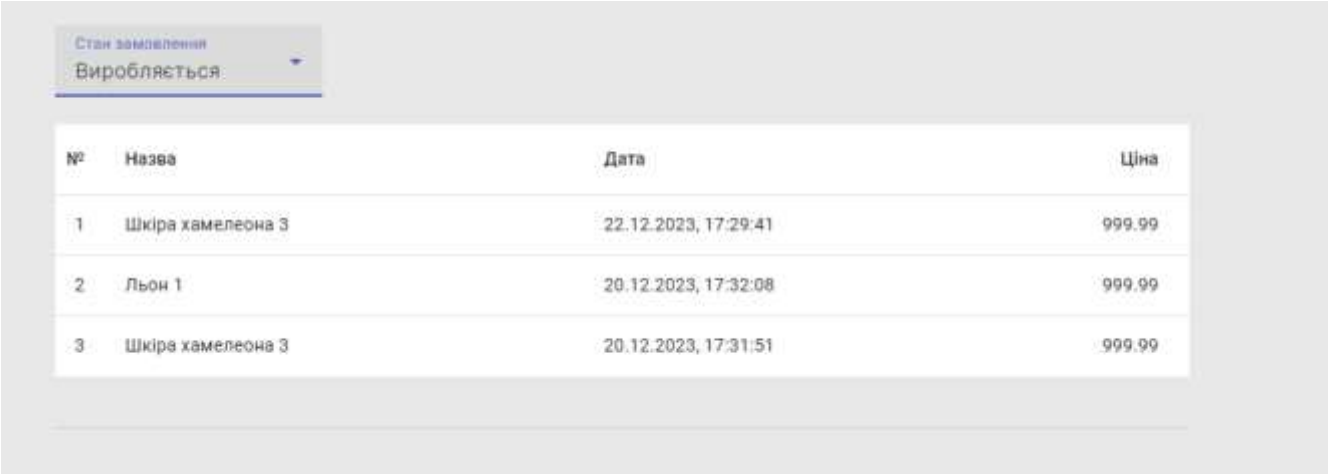


Рисунок 4.21 – Замовлення, які виготовляються.

Джерело: побудовано автором

Також є можливість перегляду історії замовлень, які вже були виготовлені раніше (рис.4.22).

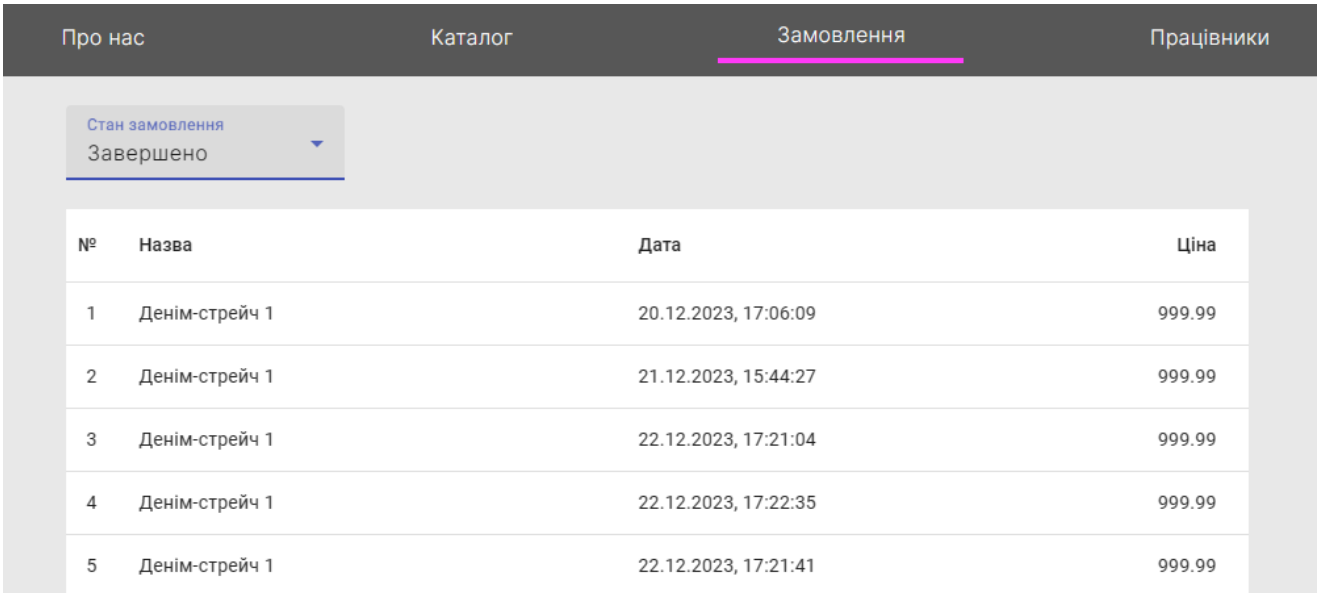


Рисунок 4.22 – Замовлення, виконання яких завершено.

Джерело: побудовано автором

Перейшовши до вкладки «Каталог», адміністратор має додаткову функцію додавання нової моделі (рис.4.23).

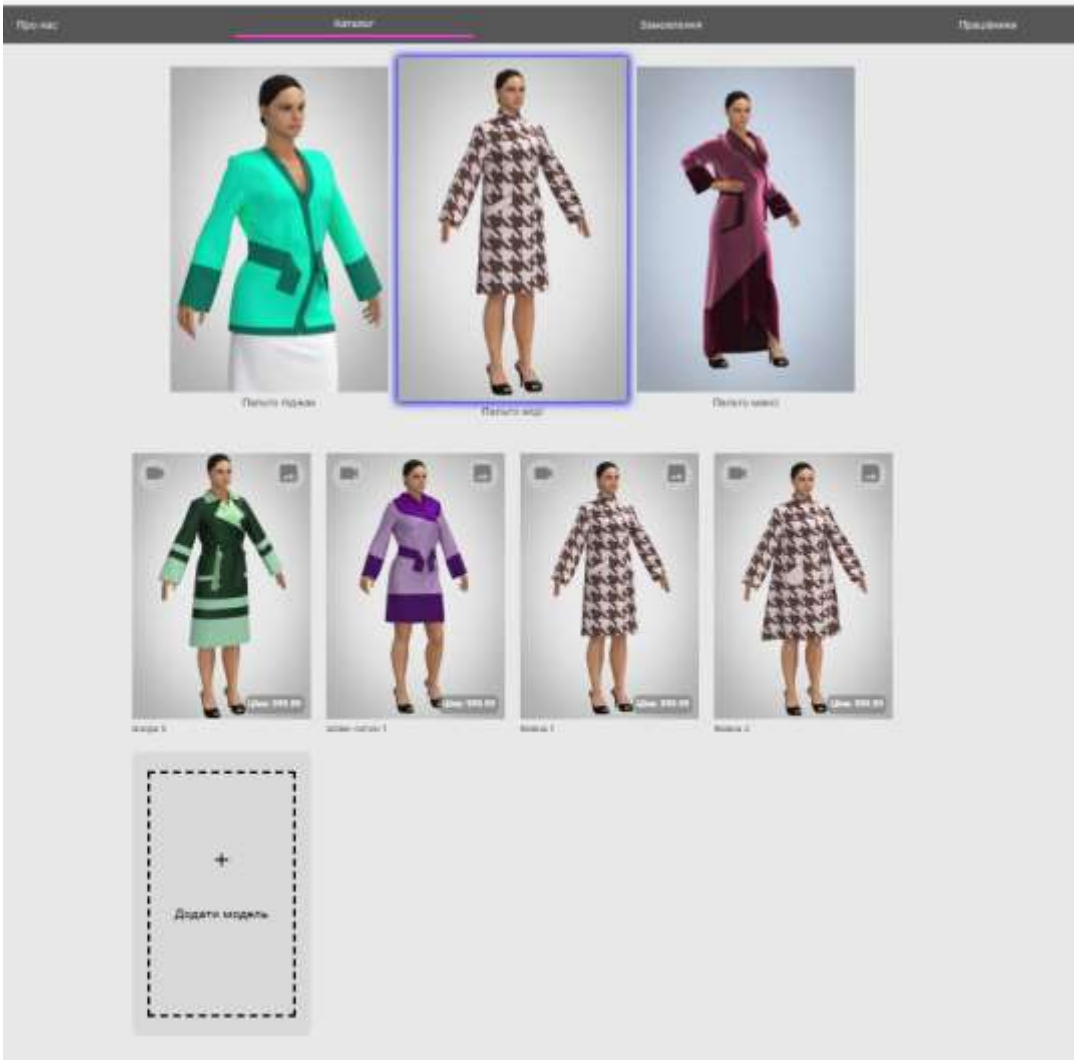


Рисунок 4.23 – Створення нової моделі

Джерело: побудовано автором

Перейшовши на вкладку «додати модель», необхідно заповнити всі поля для повної характеристики нової моделі (рис.4.24).

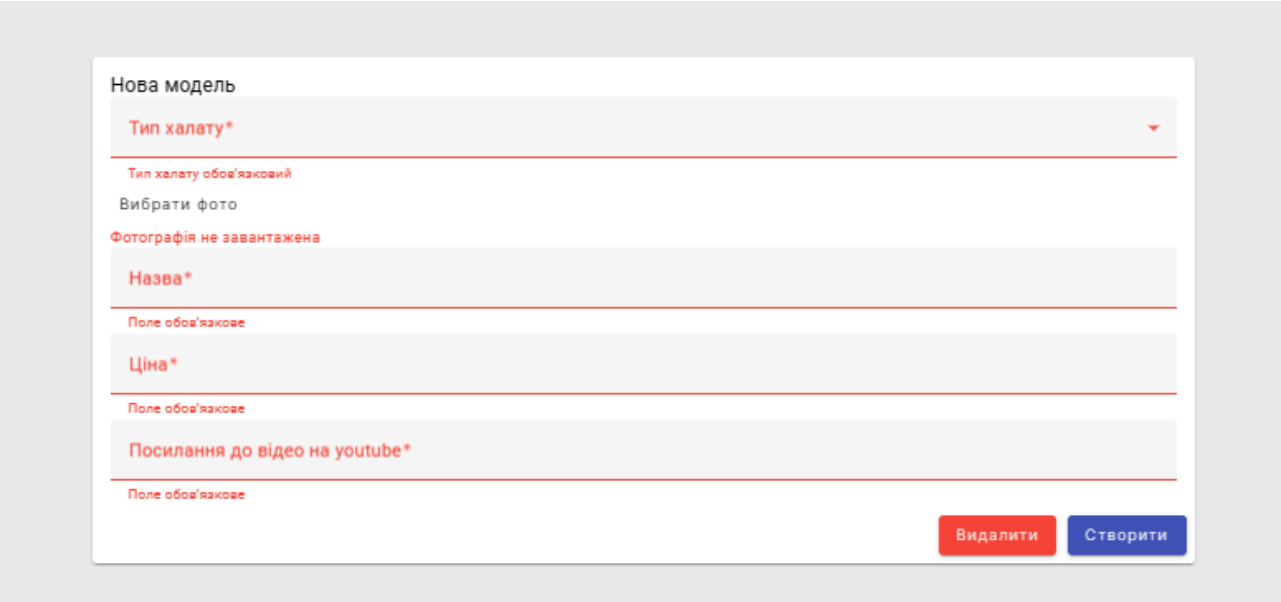


Рисунок 4.24 – Характеристика пальто.

Джерело: побудовано автором

Після заповнення всіх полів, додавання фото і відео необхідно натиснути клавішу «Створити» і нова модель буде додана до відповідного підкаталогу каталогу моделей.

Також можна редагувати і оновлювати вже існуючі моделі. Спочатку обирається модель, яку потрібно оновити чи видалити (рис.4.25).

Рисунок 4.25 – Редагування даних моделі.

Джерело: побудовано автором

Потім вносяться відповідні зміни: назва, ціна, фото, відео і натискається клавіша «Оновити». На екрані з´являється сповіщення про оновлення моделі і додавання її до каталогу (рис.4.26).

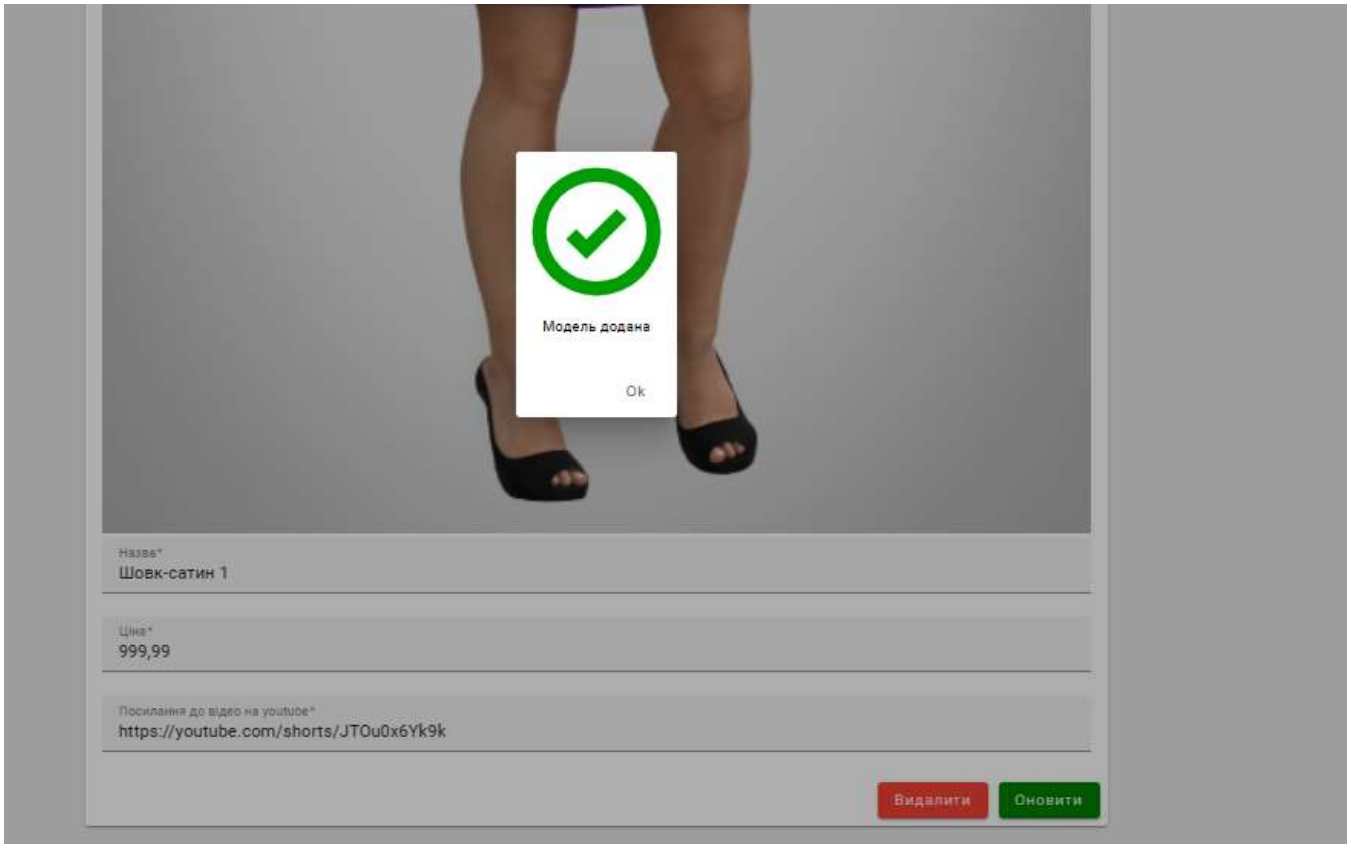


Рисунок 4.26 – Сповіщення про оновлення моделі.

Джерело: побудовано автором

4.8. Створення бази даних

База даних написана мовою SQL (Structured Query Language) в середовищі PostgreSQL, складається з 14 таблиць. Її, як і серверну частину додатку, встановлено на віртуальній машині сервісу AWS EC2.

Базу захищено логіном і паролем; крім того порт 5432, на якому вона працює, закрито для вхідних запитів.

```sql
drop schema public cascade;
create schema public;

create table "user"(
    id serial primary key ,
    username text,
    password text,
    active bool
);

create table client(
    id serial primary key,
    first_name text,
    last_name text,
    email text,
    user_id int,
    constraint client_user_fk
        foreign key (user_id) references "user"(id)
);

create table employee(
    id serial primary key,
    first_name text,
    last_name text,
    email text,
    phone_number text,
    registered_at timestamp,
```

Рисунок 4.27 – Створення Бази даних

Джерело: побудовано автором

4.9. Тестування інформаційної технології

Для забезпечення ефективності та надійності функціонування реалізованих модулів інформаційної системи було проведено ретельне функціональне тестування. Цей процес включав у себе верифікацію коректності обробки вхідних даних на формах створення, редагування та видалення інформації про користувачів, замовлення та каталоги моделей. Додатково було

випробувано навігацію додатку та основні функціональні аспекти інформаційних технологій.

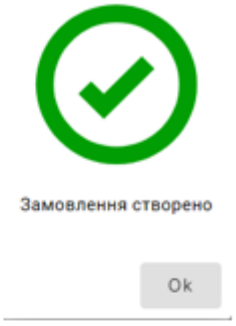
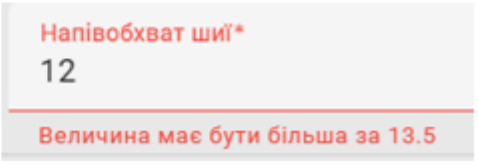
Первинний фокус було спрямовано на перевірку інтегральності та функціонування гіперпосилань у головному меню, щоб забезпечити відкриття відповідних сторінок. Під час перевірки навігаційних переходів увага була приділена відображенню актуальних даних на різних сторінках. Ключовою задачею було також забезпечення надійності процесів створення, редагування та видалення замовлень та інформації користувачів, з акцентом на коректність відображення цих даних на основних сторінках системи, таких як «Головна сторінка», «Каталог», «Замовлення», та «Акаунт».

Тестування було виконано за методологією «чорного ящика», яка передбачає аналіз функціонування системи без втручання у її внутрішній механізм та структуру програмного коду. Результати та деталі проведеного тестування були систематизовані та представлені у вигляді таблиці 4.6.

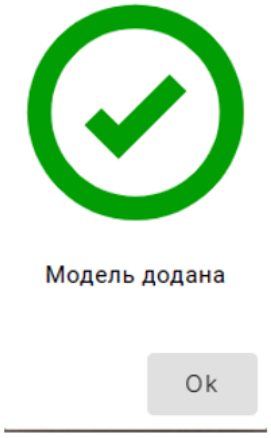
Таблиця 4.6 – Тестування форм на валідні та невалідні дані

№	Назва тест кейсу	Очікуваний результат	Фактичний результат	0/1
1	Перевірка форми створення акаунта введенням коректних даних.	Акаунт успішно створений.	Акаунт створений коректно і відображається на верхній панелі.	1
2	Перевірка форми створення акаунту введенням невалідних даних.	Відображається текст повідомлення з помилкою.	Щось пішло не так 	1

Продовження таблиці 4.6

№	Назва тест кейсу	Очікуваний результат	Фактичний результат	0/1
3	Перевірка форми створення замовлення введенням коректних даних.	Замовлення успішно створено.		1
4	Перевірка форми створення замовлення введенням некоректних даних.	Відображається текст повідомлення з помилкою.		1
5	Перевірка введення розмірних ознак клієнта.	За замовчуванням перехід до таблці з розрахунками.	Дані відображаються і з‘являється таблиця з прокалькульованими розрахунками.	1
6	Перевірка введення розмірних ознак клієнта.	Статус не створено, повідомлення про помилку підсвічується червоним кольором.		1

Продовження таблиці 4.6

№	Назва тест кейсу	Очікуваний результат	Фактичний результат	0/1
7	Додавання і коригування моделей	Модель успішно додано.		1
8	Додавання і коригування моделей за допомогою некоректних даних.	Відображається текст повідомлення з помилкою.		1

Джерело: побудовано автором

Тестуючи інформаційну систему, серйозних помилок чи недоліків не встановлено. Весь функціонал працює на необхідному рівні.

ВИСНОВКИ

Огляд та дослідження проблемної області підтримки діяльності конструктора одягу показали значну потребу в оптимізації взаємодії між ательє, підприємствами легкої промисловості та замовниками. Сучасні публікації та аналіз цільової аудиторії підтверджують релевантність та необхідність розробки таких інформаційних систем. Вивчення наявних аналогів інформаційних систем показало, що більшість існуючих рішень не зосереджується на специфіці українського ринку та особливостях будови тіла українських споживачів, що підкреслює унікальність та значимість розроблюваної системи.

Ретельний аналіз технологій, які є на ринку для створення інформаційної системи, дозволили вибрати найбільш ефективні та адаптивні рішення для розробки. Це забезпечує гнучкість системи та її можливість адаптації до специфічних потреб користувачів.

Розробка моделей і структур інформаційної системи, а також її втілення, були виконані з урахуванням сучасних стандартів та потреб кінцевих користувачів. Це забезпечило створення зручного та інтуїтивно зрозумілого інтерфейсу. Реалізована функціональність інформаційної системи включає необхідні інструменти для ефективної взаємодії між ательє, виробниками та замовниками. Функціональне тестування, проведене за методом "чорного ящика", підтвердило високу надійність та ефективність системи у виконанні основних задач, що дозволило розмістити її на хостингу.

Практична цінність - розроблена інформаційна система значно покращує процеси взаємодії з клієнтами, сприяючи підвищенню якості виготовлення одягу та ефективності роботи підприємств легкої промисловості. Створення альбому варіантів плечового одягу та впровадження інноваційних українських підходів до побудови лекал відкриває нові можливості для індивідуалізації продукції.

Система вперше представляє на українському ринку можливість замовлення одягу з урахуванням фізичних особливостей будови тіла місцевих

споживачів, що є значним внеском у розвиток вітчизняної легкої промисловості та підтримку національних традицій

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Відділ зв’язків з громадськістю та засобами масової інформації. «Більшість держзамовлень для легпрому повинні залишатися всередині країни” [Електронний ресурс] / Відділ зв’язків з громадськістю та засобами масової інформації // Міністерство економіки України. – https://www.me.gov.ua/Documents/Detail?lang=uk-UA&id=f3670bf5-b762-4f0a-adb9-303c97d27ceb&title=BilshistDerzhzamovlenDliaLegpromuPovinniZalishatisiaVserediniKraini-MinistrEkonomikiYuliiaSviridenkoNaZustrichiZPredstavnikamiGaluzi (accessed: 09.12.2023).

2. Глухова С. В. Україна: легка промисловість [Електронний ресурс] / С. В. Глухова // ВУЕ. – Режим доступу: https://vue.gov.ua/Україна:_легка_промисловість#.D0.92.D0.B8.D1.80.D0.BE.D0.B 1.D0.BD.D0.B8.D1.86.D1.82.D0.B2.D0.BE_.D0.BE.D0.B4.D1.8F.D0.B3.D1.83

3. Про затвердження плану заходів з підтримки легкої промисловості України на 2022-2024 роки [Електронний ресурс] : Розпорядж. від 16.02.2022 р. № 169-р. –https://zakon.rada.gov.ua/laws/show/169-2022-р#Text (accessed: 09.12.2023).

4. Шевеленко М. О. Впровадження інноваційних технологій у виробництво форменого обмундирування, професійного, спеціального одягу [Електронний ресурс] / М. О. Шевеленко // Освіта і наука - 2021 : матеріали студент. науково-практ. конф., Київ, 31 трав. 2021 р. – Київ, 2021

5. Якимчук Д. М. Застосування інноваційних технологій в легкій промисловості [Електронний ресурс] / Д. М. Якимчук // "Сучасні технології промислового комплексу : базові процесні інновації – 2018” : матеріали IV-ої

Міжнар. наук.-практ. конф., Херсон, 12–16 верес. 2018 р. – Херсон, 2018. – http://ekhsuir.kspu.edu/handle/123456789/7734 (accessed: 09.12.2023)..

6. Орлова Н. Підготовка майбутніх викладачів дизайну до проектування одягу засобами 3d-технологій [Електронний ресурс] / Н. Орлова // Дизайн-освіта: проблеми та перспективи : Матеріали III Всеукр. наук.-практ. Інтернет--конф., (присвяч. міжнар. Дню дизайнера), Полтава, 11–12 квіт. 2018 р. – Полтава, 2018. – http://dspace.pnpu.edu.ua/bitstream/123456789/11318/1/Orlova.pdf (accessed: 09.12.2023).

7. Можливості сучасних програм для візуалізації одягу [Електронний ресурс] / К. Пашкевич [та ін.] // Актуальні проблеми сучасного дизайну : матеріали III Міжнар. науково-практ. конф., Київ, 22 квіт. 2021 р. – Київ, 2021. – https://er.knutd.edu.ua/handle/123456789/17974 (accessed: 09.12.2023)..

8. Пашкевич К. Л. Сучасні інформаційні технології дизайну одягу [Електронний ресурс] / К. Л. Пашкевич, О. В. Єжова, Т. В. Струмінська // Дизайн одягу в полікультурному просторі : Монографія, Київ. – Київ, 2020. – https://er.knutd.edu.ua/handle/123456789/16290 (accessed: 09.12.2023)..

9. Kuska A. What is the Best Clothing Design Software for 3D Rendering & Modeling Apparel? | Cad Crowd [Electronic resource] / A. Kuska // cadcrowd. – https://www.cadcrowd.com/blog/what-is-the-best-clothing-design-software- for-3d-rendering-modeling-apparel/ (accessed: 09.12.2023).

10. Чугуєвець А. В. Тривимірне проектування у clo3d очима студента / А. В. Чугуєвець, О. В. Захаркевич // KyivTex&Fashion : матеріали VI International Scientific-Practical Conference, Київ, 20 жовт. 2022 р. – Київ, 2022

11. Y.-X. W. &. Z.-D. L. Virtual Clothing Display Platform Based on CLO3D and Evaluation of Fit [Electronic resource] / Yan-Xue Wang &. Zheng-Dong Liu sci // Journal of Fiber Bioengineering and Informatics. – 2020. – Vol. 13, no. 1. – P. 37–49. –https://doi.org/10.3993/jfbim00338 (accessed: 09.12.2023).

12. Huang S. CLO3D-Based 3D Virtual Fitting Technology of Down Jacket and Simulation Research on Dynamic Effect of Cloth [Electronic resource] / Shuxian

Huang, Li Huang // Wireless Communications and Mobile Computing. – 2022. – Vol. 2022. – P. 1–11. –https://doi.org/10.1155/2022/5835026 (accessed: 09.12.2023).

13. El-Newashy R. Physical Evaluation and CLO3D Simulation of Some Characteristics of Bio-treated Knitted Polyester/Lycra Fabrics [Electronic resource] / Rania El-Newashy, Hosam El-Din Zakaria El-Sayed // Egyptian Journal of Chemistry. 2022. – P. 0. – Mode of access: https://doi.org/10.21608/ejchem.2022.150365.6512 (date of access: 01.01.2023).

14. CLO3D Simulation versus Real Drape Test for Assessment of Garment Drape Coefficient [Electronic resource] / Basmaa Ashmawi [et al.] // Journal of Textiles, Coloration and Polymer Science. – 2021. – Vol. 18, no. 2. – P. 111–119. – https://doi.org/10.21608/jtcps.2022.151108.1130 (accessed: 09.12.2023).

15. Jankoska M. Application CAD methods in 3D clothing design [Electronic resource] / Maja Jankoska // Tekstilna industrija. – 2020. – Vol. 68, no. 4. – P. 31–37. – Mode of access: https://doi.org/10.5937/tekstind2004031j

16. Digital- 3D проектування одягу Online [Електронний ресурс] // Курси і майстер-класи в Академії стилю та дизайну Андре Тан!. – https://academy.andretan.com.ua/digital---3d (accessed: 09.12.2023)..

17. 3D моделирование одежды | Программа для дизайна одежды VIDYA [Електронний ресурс] // САПР одежды Assyst | Программа для создания одежды. https://assyst-cis.com/3d-modelirovanie/ (accessed: 09.12.2023)..

18. Гаюр А. Застосування мобільних додатків для вимірювання індивідуальних особливостей фігури споживача / А. Гаюр, А. Чугуєвець, О. Дітковська // Хмельницький. – Хмельницький, 2022.

19. Petrosova I. The System of Selection and Sale of Ready-to-Wear Clothes in a Virtual Environment [Electronic resource] / I. Petrosova, E. Andreeva, M. Guseva // 2019 International Science and Technology Conference "EastConf" : Materials of the international conference, Vladivostok, 1–2 May 2019. – Vladivostok, 2019. – https://doi.org/10.1109/EastConf.2019.8725390 (accessed: 09.12.2023).

20. Борщевська Н. Віртуальна мода: одяг, створений цифровим способом [Електронний ресурс] / Н. Борщевська, В. Зіркевич // Актуальні

проблеми сучасного дизайну : матеріали III Міжнар. науково-практ. конф., Київ, 22 квіт. 2021 р. – Київ, 2021. –https://er.knutd.edu.ua/handle/123456789/17955 (accessed: 09.12.2023)..

21. Santos R. People Are Buying Digital Clothes Because That's a Thing Now [Electronic resource] / R. Santos // vice.com. – https://www.vice.com/en/article/akvzqz/virtual-digital-clothes-fashion-game-skins-metaverse (accessed: 09.12.2023)..

22. Альтман Д. Віртуальний одяг від діджитал-ритейлера Dress-X [Електронний ресурс] / Д. Альтман // https://bazilik.media. – https://bazilik.media/virtualnyj-odiah-vid-didzhytal-rytejlera-dress-x/. (accessed: 09.12.2023).

23. Park H. Do Augmented and Virtual Reality Technologies Increase Consumers' Purchase Intentions? The Role of Cognitive Elaboration and Shopping Goals. [Electronic resource] / H. Park, S. Kim // Clothing and Textiles Research Journal. 2021. –https://journals.sagepub.com/doi/full/10.1177/0887302X21994287 (accessed: 09.12.2023).

24. Лавренко М. А. Комп'ютерні технології для віртуального примірювання одягу [Електронний ресурс] / М. А. Лавренко, К. Л. Пашкевич // Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі : матеріали Міжнар. науково-практ. конф., Київ. – Київ. – https://er.knutd.edu.ua/bitstream/123456789/13322/1/Пашкевич.pdf (accessed: 09.12.2023).

25. Advanced Virtual Apparel Try Using Augmented Reality (AVATAR) [Electronic resource] / Sourav Shaw [et al.] // Advances in Intelligent Systems and Computing. – Singapore, 2020. – P. 479–487. –https://doi.org/10.1007/978-981-15-4032-5_44 (accessed: 09.12.2023).

26. Virtual Fashion Mirror [Electronic resource] / Deepthi Prakash [et al.] // 2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP), Chennai, India, 28–29 September 2020. – [S. l.], 2020. – https://doi.org/10.1109/icccsp49186.2020.9315257 (accessed: 09.12.2023).

27. Hong H. Towards Multi-pose Guided Virtual Try-on Network [Electronic resource] / H. Dong, X. Liang. – [S. l.],2019. – https://openaccess.thecvf.com/content_ICCV_2019/papers/Dong_Towards_Multi-Pose_Guided_Virtual_Try-On_Network_ICCV_2019_paper.pdf (accessed: 09.12.2023).

28. Minar M. R. 3D Reconstruction of Clothes using a Human Body Model and its Application to Image-based Virtual Try-On [Electronic resource] / M. R. Minar, T. Tuan. – [S. l.], 2020. –https://minar09.github.io/c3dvton/cvprw20_3d.pdf (accessed: 09.12.2023).

29. Yang H. Towards Photo-Realistic Virtual Try-On by Adaptively Generating-Preserving Image Content [Electronic resource] / H. Yang, R. Zhang, X. Guo. – [S.l.], 2020. – https://openaccess.thecvf.com/content_CVPR_2020/html/Yang_Towards_Photo-Realistic_Virtual_Try-On_by_Adaptively_Generating-Preserving_Image_Content_CVPR_2020_paper.html (accessed: 09.12.2023)..

30. Ge C. Disentangled Cycle Consistency for Highly-Realistic Virtual Try-On [Electronic resource] / C. Ge, Y. Song. – [S. l.], 2021. – https://openaccess.thecvf.com/content/CVPR2021/html/Ge_Disentangled_Cycle_Co ns istency_for_Highly-Realistic_Virtual_Try-On_CVPR_2021_paper.html (accessed: 09.12.2023).

31. Applicability of a Single Depth Sensor in Real-Time 3D Clothes Simulation: Augmented Reality Virtual Dressing Room Using Kinect Sensor [Electronic resource] / S. B. Adikari [et al.] // Advances in Human-Computer Interaction, Kahir, 18 May 2020. – Kahir, 2020. – https://www.hindawi.com/journals/ahci/2020/1314598/ (accessed: 09.12.2023).

32. Ласіцина Д. Віртуальні простори для презентації модних колекцій [Електронний ресурс] / Д. Ласіцина // KyivTex&Fashion : V Міжнар. науково-практ. конф. текстил. та fashion технологій, Київ, 21 жовт. 2021 р. – Київ, 2021. – https://er.knutd.edu.ua/bitstream/123456789/19129/1/KyivTex&Fashion2021_P136-137.pdf. (accessed: 09.12.2023).

33. Zhylenko T. Mobile Application to Calculate the Parameters of Top Wear Basic Design [Electronic resource] / T. Zhylenko, A. Kudryavtsev O. Zakharkevich// Science and innovation. – 2019. – Vol. 15 №3. – P. 24–34. –https://scinn-eng.org.ua/archive/15%283%29/15%283%2903 (accessed: 09.12.2023).

34. Zakharkevich O. Development of a Mobile Application to Study Sewing Techniques for Manufacturing fur and Leather Clothes [Electronic resource] / O. Zakharkevich, O. Paraska, J. Koshevko, G. Shvets, A. Shvets, T. Zhylenko // FIBRES & TEXTILES in Eastern Europe. – 2023. – Vol. 31. – P. 1–10. – https://sciendo.com/article/10.2478/ftee-2023-0011 (accessed: 09.12.2023)..

35. Zakharkevich O. Development of an algorithm for the reasoned selectioned of machines for leather garments manufacturing [Electronic resource] / O. Zakharkevich, T. Zhylenko, J. Koshevko, G. Shvets // Eastern-European Journal of Enterprise Technologies. – 2023. – Vol. 5. – P. 86–94. – https://journals.uran.ua/eejet/article/view/287482/283710 (accessed: 09.12.2023)..

36. Naychuk D. Testing projection technique of fabric cutting in the apparel manufacturing process. / Naychuk D., Zakharkevich O., Zhylenko T., Prybeha D., Kuleshova S., Koshevko J. // *AIP Conference Proceedings*. 2023. 2889. – https://pubs.aip.org/aip/acp/article-abstract/2889/1/040008/2928319/sting-projection-technique-of-fabric-cutting-in?redirectedFrom=fulltext (accessed: 09.12.2023)..

37. Шатілов О. В. Моделювання процесу управління стратегічною гнучкістю підприємства / О. В. Шатілов. // Ефективна економіка. – 2013. – №1.

38. Мінухін С. В. Методи і моделі проектування на основі сучасних CASE-засобів. Навчальний посібник / С. В. Мінухін, О. М. Беседовський, С. В. Беседовський. – Харків, 2008. – 272 с. – (Харківський національний економічний університет).

39. What is Black Box Testing? - Check Point Software [Electronic resource]. URL: https://www.checkpoint.com/cyber-hub/cyber-security/what-is-penetration-testing/what-is-black-box-testing/ (accessed: 09.12.2023).

ДОДАТОК А
ПЛАНУВАННЯ РОБІТ

А.1. Ідентифікація ідеї проекту

Задля підвищення рівня конкурентоспроможності українських виробників шляхом капітальних інвестицій у модернізацію основних засобів; ефективного використання коштів державного бюджету у сфері підготовки та перепідготовки кадрів для легкої промисловості, спрощення роботи ательє та підприємств легкої промисловості по виробленню одягу масового використання за допомогою дигіталізації, а також можливості клієнта самостійно обирати бажану модель за рахунок аналізу її 3D зображення.

Необхідно створити інформаційну системну підтримку, яка відповідатиме наступним вимогам: швидка побудова лекал одягу майстром за розмірними ознаками фігури клієнта, візуалізація бажаної моделі одягу, збереження даних заказу клієнта, наповнення портфоліо.

А.2. Деталізація мети методом SMART

Метою проекту є створення інформаційної системи є забезпечення актуальної інформації про сервіс, його послуги, інформацію про товари, можливість їх замовленн. Результати деталізації мети проекту методом SMART подано у табл. А.1.

Таблиця А.1 – Деталізація мети методом SMART

Specific (конкретна)	Розробити інформаційну систему підтримки діяльності конструктора одягу.
Measurable (вимірювання)	Результатом роботи інформаційної системи є задоволеність і оцінка замовника.
Achievable (досяжна, узгоджена)	Мета даного проекту вважається досяжною, оскільки розробник володіє необхідними навичками у створенні веб-сторінок засобами html, css, javascript. Мета була узгоджена з вимогами та потребами замовника.
Relevant (реалістична)	Для реалізації продукту проекту є всі необхідні технічні та програмні засоби, доступ до мережі Інтернет. Розробник кваліфікований для виконання поставлених задач.
Time-framed (обмежена в часі)	Інформаційна система розроблюється з обмеженням у часі на основі наперед сформованого календарного плану (діаграми Ганта).

А.3. Планування змісту структури робіт ІТ – проекту

Під час планування розробки інформаційної системи підтримки діяльності конструктора одягу була створена WBS. Ця ієрархічна структура побудована з метою логічного розподілу усіх робіт з виконання проекту і подана у графічному вигляді. Структура складається з сукупності рівнів, кожен з яких формується в результаті розподілу роботи попереднього рівня на складові наступного рівня. В результаті елементом найнижчого рівня є група робіт, або так званий робочий пакет (work package).

WBS-структура для даного проекту представлена на рис. А.1

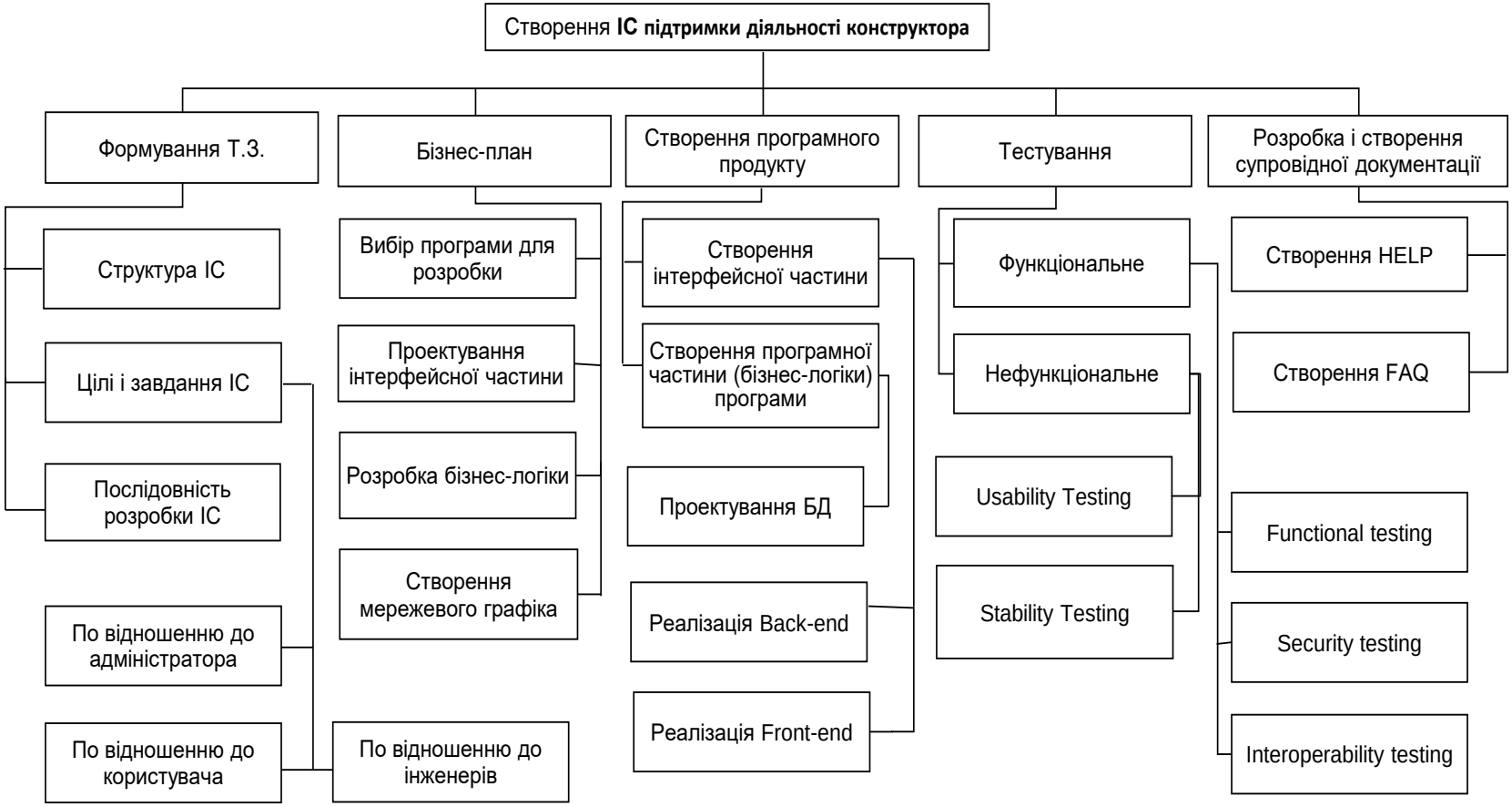


Рисунок А.1 – WBS-структура інформаційної системи підтримки діяльності конструктора одягу

Після побудови структури WBS проекту виконується етап розробки OBS (Organization structure). Організаційна структура проекту (OBS) також має ієрархічну структуру управління проектом і демонструє відносини між учасниками проекту.

Організаційна структура проекту:

організовується на рівні підприємства;

її елементи становлять призначення на рівні пакетів робіт структури декомпозиції робіт (WBS);

надає можливість контролювати доступ користувачів до інформації відповідного рівня.

OBS-структура для даного проекту представлена на рис. А.2.

Рисунок А.2 – Організаційна структура

А.4. Побудова календарного графіку виконання ІТ – проекту

Діаграма Ганта (Gantt Chart) – є однією з найпопулярніших інструментів для унаочнення календарного плану в проектному менеджменті.

Діаграма Ганта допомагає контролювати відсоток робіт, виконаних за кожним завданням. Керівникам проектів дуже важливо правильно розподілити завдання і бути впевненими в тому, що проект буде завершений вчасно. Основна увага діаграм Ганта полягає у відображенні відсотку завершення робіт по кожному завданню.

Графік виконання дипломного проекту представлено у вигляді Діаграми Ганта на рис. А.3.

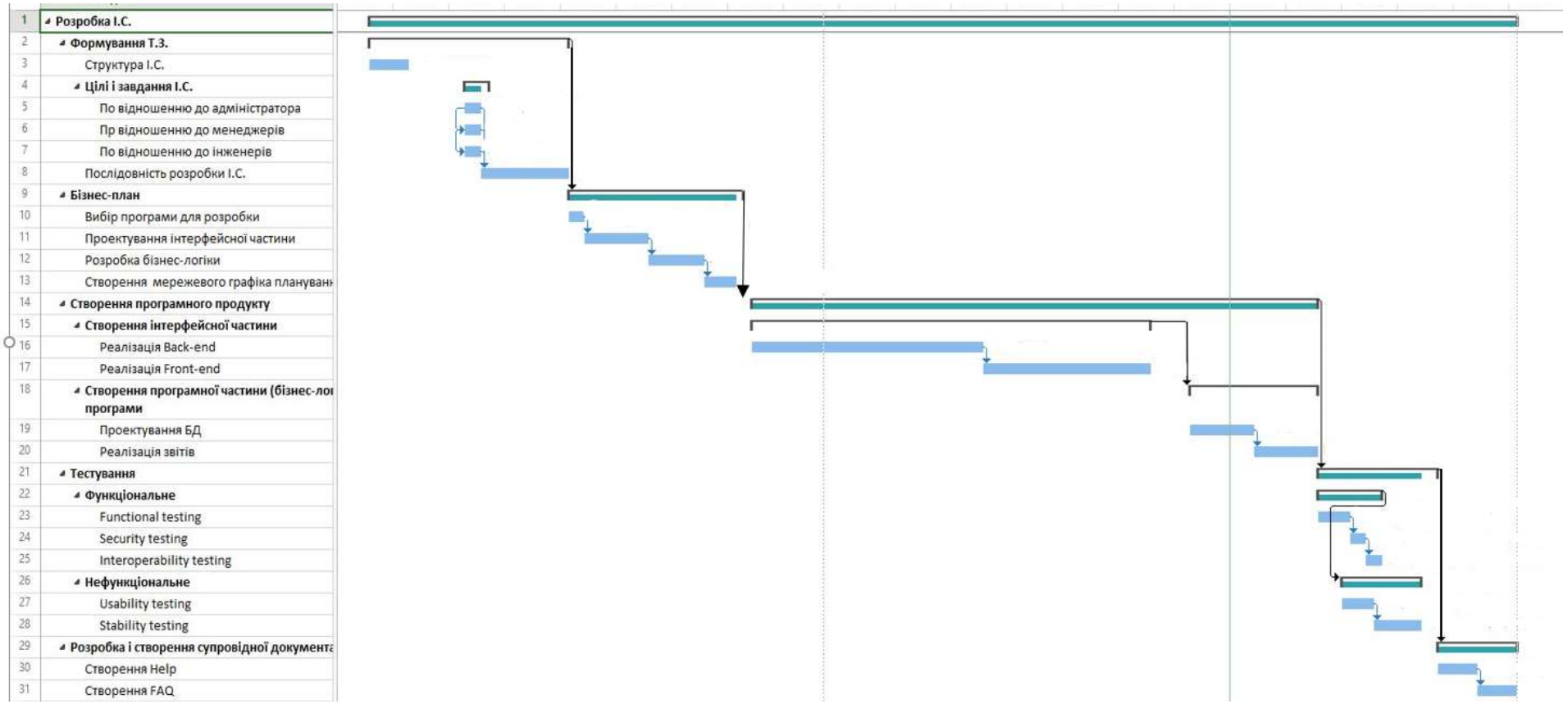


Рисунок А.3 – Діаграма Ганта

А.5. Планування ризиків проекту

Під час створення нового проекту необхідно врахувати можливі ризики, які можуть трапитись та вплинути на реалізацію проекту. Тому необхідно забезпечити заходи для їх уникнення чи зменшення збитків.

Основними ризиками під час розробки інформаційної системи можна вважати:

- зміна ТЗ на етапі розробки;
- зміщення календарного плану;
- збої при виготовлені продукту;
- висока залежність від ключових співробітників (недостатня кількість навичок для створення запланованого функціоналу);
- перебій в роботі хостингу;
- інформаційні втрати через поломку технічних засобів;
- зміщення дедлайну.

Далі наведемо шкалу ймовірності виникнення та величину втрат (табл.А.2).

Таблиця А.2 – Шкала оцінювання ймовірності виникнення та величини витрат

Оцінка	Ймовірність виникнення	Величина втрат
1	Слабоймовірна	Мінімальна
2	Малоймовірна	Низька
3	Ймовірна	Середня
4	Досить ймовірна	Висока
5	Майже можлива	Максимальна

Згідно наведених даних була проведена класифікація ризиків для даного проекту(табл. А.3).

Таблиця А.3 – Класифікація ризиків дипломного проекту.

№	Назва ризику	Ймовірність	Величина втрат
1.	Зміна ТЗ на етапі розробки.	2	4
2.	Зміщення календарного плану.	2	3
3.	Збої при виготовлені продукту.	3	5
4.	Висока залежність від ключових співробітників.	3	3
5.	Перебій в роботі хостингу.	1	5
6.	Інформаційні втрати через поломку технічних засобів.	2	2
7.	Зміщення дедлайну.	1	2

За даною класифікацією, була побудована матриця ризиків, представлена на рис. А.4.

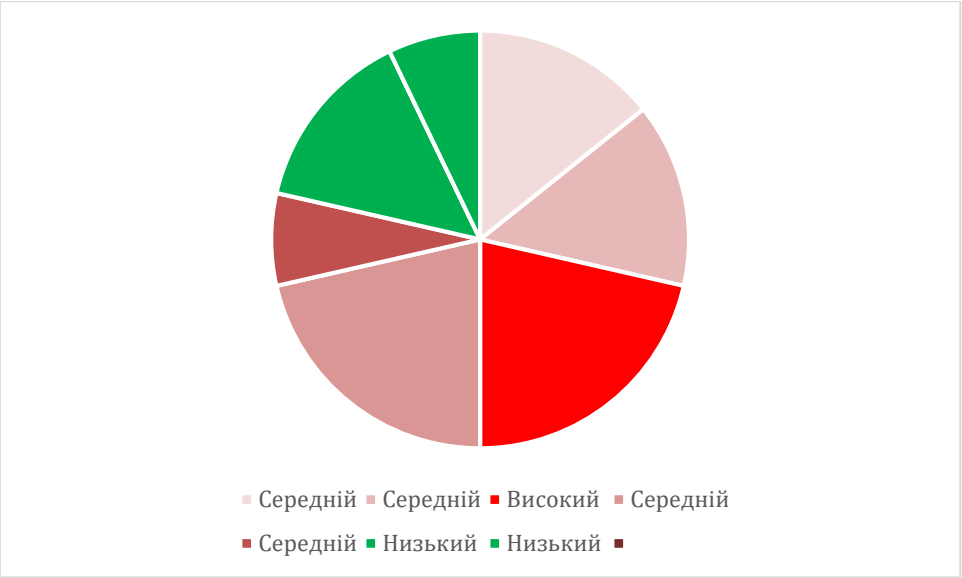


Рисунок А.4 – Діаграма ризиків

З аналізу матриці ризиків проведено оцінку рівня ризику для кожного ризику в проекті(табл. А.4).

Таблиця А.4 – Оцінка рівнів ризику

№	Назва ризику	Ймовірність	Величина втрат	Рівень ризку
1	Зміна ТЗ на етапі розробки	2	4	Середній
2	Зміщення календарного плану.	2	3	Середній
3	Збої при виготовлені продукту.	3	5	Високий
4	Висока залежність від ключових співробітників.	3	4	Середній
5	Перебій в роботі хостингу.	1	5	Середній
6	Інформаційні втрати через поломку технічних засобів.	2	2	Низький
7	Зміщення дедлайну.	1	2	Низький

ДОДАТОК Б
КОД РЕАЛІЗАЗІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Б.1. Скрипт для створення бази даних

create_tables.sql

```
drop schema public cascade;
create schema public;
create table "user"(
    id serial primary key ,
    username text,
    password text,
    active bool
);
create table client(
    id serial primary key,
    first_name text,
    last_name text,
    email text,
    user_id int,
    constraint client_user_fk
        foreign key (user_id) references "user"(id)
);
create table employee(
    id serial primary key,
    first_name text,
    last_name text,
    email text,
    phone_number text,
    registered_at timestamp,
    user_id int,
    constraint employee_user_fk
        foreign key (user_id) references "user"(id)
);
create table client_metrics(
    client_id int,
    neck_semi_circumference float,
    chest_semi_circumference_1 float,
    chest_semi_circumference_2 float,
    chest_semi_circumference_3 float,
    waist_semi_circumference float,
    shoulder_width float,
    chest_height float,
    chest_height_1 float,
    back_armhole_height float,
    back_length_till_waist float,
    shoulder_height_sidelong float,
    chest_width float,
    chest_center float,
    back_width float,
    waist_length_front float,
    neck_base_to_front_waist_line_distance float,
    constraint client_metrics_client_fk
        foreign key (client_id) references client(id)
);
create table address(
    id serial primary key,
    city text,
    street text,
    building_number text,
    apartment_number text,
    zip_code text
);
create table client_details(
    client_id int,
    birth_date date,
    phone_number text,
    address_id int,
    constraint client_details_client_fk
        foreign key (client_id) references client(id),
```

```
    constraint client_details_address_fk
        foreign key (address_id) references address(id)
);
create table product_metrics(
    id serial primary key ,
    neck_semi_circumference float,
    chest_semi_circumference_1 float,
    chest_semi_circumference_2 float,
    chest_semi_circumference_3 float,
    waist_semi_circumference float,
    shoulder_width float,
    chest_height float,
    chest_height_1 float,
    back_armhole_height float,
    back_length_till_waist float,
    shoulder_height_sidelong float,
    chest_width float,
    chest_center float,
    back_width float,
    waist_length_front float,
    neck_base_to_front_waist_line_distance float,
    increase_to_width_by_chest_line float,
    increase_to_armhole_depth float,
    increase_to_neck_back float
);
create table user_role(
    user_id int,
    role text
);
create table token(
    id bigserial,
    value text,
    type text,
    user_id int
);
create table coat_model(
    id serial primary key,
    name text,
    price float,
    coat_type text,
    img_path text,
    video_url text
);
create table pattern_data (
    id serial primary key,
    basis_grid_width float,
    basis_grid_length float,
    armhole_depth float,
    back_width float,
    file_width float,
    armhole_width float,
    back_neck_width float,
    back_neck_height float,
    shoulder_cut_slope float,
    shoulder_cut_end float,
    side_slope_top float,
    back_armhole_slope float,
    product_balance float,
    file_neck_width float,
    file_neck_depth float,
    chest_dart float,
    shoulder_slope float,
    armhole float,
    total_dart_deviation_by_waist_line float,
    side_dart float,
    file_dart float,
    back_dart float,
    increase_to_width_by_chest_line float,
    increase_to_width_by_waist_line float,
    increase_to_armhole_width float,
    increase_to_armhole_depth float,
    increase_to_neck_back float,
    increase_to_back_width float,
    increase_to_file_width float
);
create table "order"(
    id serial primary key,
    client_id int,
    employee_id int,
    coat_model_id int,
```

```
    pattern_data_id int,
    product_metrics_id int,
    created_at timestamp,
    status text,
    img_path text,
    constraint fk_order_client
        foreign key (client_id) references client(id),
    constraint fk_order_employee
        foreign key (employee_id) references employee(id),
    constraint fk_order_coat_model
        foreign key (coat_model_id) references coat_model(id),
    constraint fk_order_pattern_data
        foreign key (pattern_data_id) references pattern_data(id),
    constraint fk_order_product_metrics
        foreign key (product_metrics_id) references product_metrics(id)
);
create table review(
    id serial primary key,
    content text,
    rating smallint,
    created_at timestamp,
    coat_model_id int,
    client_id int,
    constraint fk_review_coat_model
        foreign key (coat_model_id) references coat_model(id),
    constraint fk_review_client
        foreign key (client_id) references client(id)
);
create table review_reply(
    review_id int primary key,
    content text,
    created_at timestamp,
    constraint fk_review_reply_review
        foreign key (review_id) references review(id)
);
```

Б.2. Розрахунок даних лекала

PatternData.java

```
@Entity
@Table(name = "pattern_data")
@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
@Builder
public class PatternData {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name = "basis_grid_width")
    private Double basisGridWidth;
    @Column(name = "basis_grid_length")
    private Double basisGridLength;
    @Column(name = "armhole_depth")
    private Double armholeDepth;
    @Column(name = "back_width")
    private Double backWidth;
    @Column(name = "file_width")
    private Double fileWidth;
    @Column(name = "armhole_width")
    private Double armholeWidth;
    @Column(name = "back_neck_width")
    private Double backNeckWidth;
    @Column(name = "back_neck_height")
    private Double backNeckHeight;
    @Column(name = "shoulder_cut_slope")
    private Double shoulderCutSlope;
    @Column(name = "shoulder_cut_end")
    private Double shoulderCutEnd;
    @Column(name = "side_slope_top")
    private Double sideSlopeTop;
    @Column(name = "back_armhole_slope")
    private Double backArmholeSlope;
    @Column(name = "product_balance")
```

```
    private Double productBalance;
    @Column(name = "file_neck_width")
    private Double fileNeckWidth;

    @Column(name = "file_neck_depth")
    private Double fileNeckDepth;
    @Column(name = "chest_dart")
    private Double chestDart;
    @Column(name = "shoulder_slope")
    private Double shoulderSlope;
    @Column(name = "armhole")
    private Double armhole;
    @Column(name = "total_dart_deviation_by_waist_line")
    private Double totalDartDeviationByWaistLine;
    @Column(name = "side_dart")
    private Double sideDart;
    @Column(name = "file_dart")
    private Double fileDart;
    @Column(name = "back_dart")
    private Double backDart;
    @Column(name = "increase_to_width_by_chest_line")
    private Double increaseToWidthByChestLine;
    @Column(name = "increase_to_width_by_waist_line")
    private Double increaseToWidthByWaistLine;
    @Column(name = "increase_to_armhole_width")
    private Double increaseToArmholeWidth;
    @Column(name = "increase_to_armhole_depth")
    private Double increaseToArmholeDepth;
    @Column(name = "increase_to_neck_back")
    private Double increaseToNeckBack;
    @Column(name = "increase_to_back_width")
    private Double increaseToBackWidth;
    @Column(name = "increase_to_file_width")
    private Double increaseToFileWidth;
}
```

PatternCalculatorController.java

```
@RestController
@RequestMapping("/patterns")
@RequiredArgsConstructor
public class PatternCalculatorController {
    private final PatternCalculatorService patternCalculatorService;

    @PostMapping("/calculate")
    public ResponseEntity<PatternDataResponse> calculatePatternData(@RequestBody @Valid
ProductMetricsDTO request){
        return ResponseEntity.ok(patternCalculatorService.calculatePatternData(request));
    }
}
```

PatternCalculatorService.java

```
@Service
@RequiredArgsConstructor
public class PatternCalculatorService {
    private final PatternCalculator patternCalculator;
    private final PatternDataMapper patternDataMapper;
    private final PatternDataRepository patternDataRepository;

    public PatternDataResponse calculatePatternData(ProductMetricsDTO request) {
        PatternData patternData = patternCalculator.doCalculate(request);
        return patternDataMapper.entityToResponse(patternData);
    }

    public PatternData calculatePatternDataAndSave(ProductMetricsDTO request) {
        PatternData patternData = patternCalculator.doCalculate(request);
        return patternDataRepository.save(patternData);
    }
}
```

PatternCalculator.java

```
@Component
public class PatternCalculator {
    private static final RoundingMode ROUNDING_MODE = RoundingMode.HALF_UP;
```

```
    public PatternData doCalculate(ProductMetricsDTO request){
        // Пт
        BigDecimal increaseToWidthByWaistLine = BigDecimal
                .valueOf(request.increaseToWidthByChestLine())
                .multiply(BigDecimal.valueOf(0.5))
                .setScale(1, ROUNDING_MODE);
        // Пс
        BigDecimal increaseToBackWidth = BigDecimal
                .valueOf(request.increaseToWidthByChestLine())
                .multiply(BigDecimal.valueOf(0.25))
                .setScale(1, ROUNDING_MODE);;
        // Пп
        BigDecimal increaseToFileWidth = BigDecimal
                .valueOf(request.increaseToWidthByChestLine())
                .multiply(BigDecimal.valueOf(0.25))
                .setScale(1, ROUNDING_MODE);;
        // Ппр
        BigDecimal increaseToArmholeWidth = increaseToWidthByWaistLine
                .subtract(increaseToBackWidth.add(increaseToFileWidth))
                .setScale(1, ROUNDING_MODE);

        // АА1
        BigDecimal basisGridWidth = BigDecimal
                .valueOf(request.clientMetrics().chestSemiCircumference2())
                .add(BigDecimal.valueOf(request.increaseToWidthByChestLine()))
                .setScale(1, ROUNDING_MODE);
        // АТ
        BigDecimal basisGridLength = BigDecimal
                .valueOf(request.clientMetrics().backLengthTillWaist())
                .setScale(1, ROUNDING_MODE);
        // АГ
        BigDecimal armholeDepth = BigDecimal
                .valueOf(request.clientMetrics().backArmholeHeight())
                .add(BigDecimal.valueOf(request.increaseToArmholeDepth()))
                .setScale(1, ROUNDING_MODE);
        // Аа
        BigDecimal backWidth = BigDecimal
                .valueOf(request.clientMetrics().backWidth())
                .add(increaseToBackWidth)
                .setScale(1, ROUNDING_MODE);
        // А1а1
        BigDecimal fileWidth = BigDecimal
                .valueOf(request.clientMetrics().chestWidth())
                .add(
                        BigDecimal
                                .valueOf(request.clientMetrics().chestSemiCircumference2())

.subtract(BigDecimal.valueOf(request.clientMetrics().chestSemiCircumference1()))
                ).add(increaseToFileWidth)
                .setScale(1, ROUNDING_MODE);
        // аа1
        BigDecimal armholeWidth = basisGridWidth
                .subtract(backWidth)
                .subtract(fileWidth)
                .setScale(1, ROUNDING_MODE);
        // АА2
        BigDecimal backNeckWidth = BigDecimal
                .valueOf(request.clientMetrics().neckSemiCircumference())
                .divide(BigDecimal.valueOf(3), ROUNDING_MODE)
                .add(BigDecimal.valueOf(request.increaseToNeckBack()))
                .setScale(1, ROUNDING_MODE);
        // А2А3
        BigDecimal backNeckHeight = backNeckWidth
                .divide(BigDecimal.valueOf(3), ROUNDING_MODE)
                .add(BigDecimal.valueOf(0.3))
                .setScale(1, ROUNDING_MODE);
        // аП
        BigDecimal shoulderCutSlope = BigDecimal.valueOf(2);
        // А3П1
        BigDecimal shoulderCutEnd = BigDecimal.valueOf(request.clientMetrics().shoulderWidth());
        // А1А4
        BigDecimal productBalance = BigDecimal
                .valueOf(request.clientMetrics().neckBaseToFrontWaistLineDistance())
                .subtract(BigDecimal.valueOf(request.clientMetrics().backLengthTillWaist()))
                .setScale(1, ROUNDING_MODE);
        // А4А5
        BigDecimal fileNeckWidth = backNeckWidth;
        // А4А6
        BigDecimal fileNeckDepth = fileNeckWidth
                .add(BigDecimal.valueOf(1))
```

```
                .divide(BigDecimal.valueOf(1.5), ROUNDING_MODE)
                .setScale(1, ROUNDING_MODE);

        // Г1Г5
        BigDecimal chestDart = BigDecimal
                .valueOf(request.clientMetrics().chestCenter())
                .add(BigDecimal.valueOf(0.5))
                .setScale(1, ROUNDING_MODE);
        // А5Г6 = Вг1
        BigDecimal shoulderSlope = BigDecimal
                .valueOf(request.clientMetrics().chestHeight())
                .subtract(
                        BigDecimal
                                .valueOf(request.clientMetrics().waistLengthFront())
                                .subtract(BigDecimal

.valueOf(request.clientMetrics().neckBaseToFrontWaistLineDistance()))
                ).setScale(1, ROUNDING_MODE);
        // А7А8
        BigDecimal armhole = BigDecimal
                .valueOf(request.clientMetrics().chestSemiCircumference2())
                .subtract(BigDecimal.valueOf(request.clientMetrics().chestSemiCircumference1()))
                .multiply(BigDecimal.valueOf(2))
                .add(BigDecimal.valueOf(1.5))
                .setScale(1, ROUNDING_MODE);
        // Рв.т.
        BigDecimal totalDartDeviationByWaistLine = BigDecimal
                .valueOf(request.clientMetrics().chestSemiCircumference2())
                .add(BigDecimal.valueOf(request.increaseToWidthByChestLine()))
                .subtract(
                        BigDecimal
                                .valueOf(request.clientMetrics().waistSemiCircumference())
                                .add(increaseToWidthByWaistLine)
                ).setScale(1, ROUNDING_MODE);
        // Рб.в
        BigDecimal sideDart = totalDartDeviationByWaistLine
                .divide(BigDecimal.valueOf(3), ROUNDING_MODE)
                .add(BigDecimal.valueOf(1))
                .setScale(1, ROUNDING_MODE);
        // Рв.п
        BigDecimal fileDart = totalDartDeviationByWaistLine
                .divide(BigDecimal.valueOf(3), ROUNDING_MODE)
                .subtract(BigDecimal.valueOf(1))
                .setScale(1, ROUNDING_MODE);
        // Рв.с
        BigDecimal backDart = totalDartDeviationByWaistLine
                .divide(BigDecimal.valueOf(3), ROUNDING_MODE)
                .setScale(1, ROUNDING_MODE);

        return PatternData.builder()
                .basisGridWidth(basisGridWidth.doubleValue())
                .basisGridLength(basisGridLength.doubleValue())
                .armholeDepth(armholeDepth.doubleValue())
                .backWidth(backWidth.doubleValue())
                .fileWidth(fileWidth.doubleValue())
                .armholeWidth(armholeWidth.doubleValue())
                .backNeckWidth(backNeckWidth.doubleValue())
                .backNeckHeight(backNeckHeight.doubleValue())
                .shoulderCutSlope(shoulderCutSlope.doubleValue())
                .shoulderCutEnd(shoulderCutEnd.doubleValue())
                .productBalance(productBalance.doubleValue())
                .fileNeckWidth(fileNeckWidth.doubleValue())
                .fileNeckDepth(fileNeckDepth.doubleValue())
                .chestDart(chestDart.doubleValue())
                .shoulderSlope(shoulderSlope.doubleValue())
                .armhole(armhole.doubleValue())
                .totalDartDeviationByWaistLine(totalDartDeviationByWaistLine.doubleValue())
                .sideDart(sideDart.doubleValue())
                .fileDart(fileDart.doubleValue())
                .backDart(backDart.doubleValue())
                .increaseToWidthByChestLine(request.increaseToWidthByChestLine())
                .increaseToWidthByWaistLine(increaseToWidthByWaistLine.doubleValue())
                .increaseToArmholeWidth(increaseToArmholeWidth.doubleValue())
                .increaseToArmholeDepth(request.increaseToArmholeDepth())
                .increaseToNeckBack(request.increaseToNeckBack())
                .increaseToBackWidth(increaseToBackWidth.doubleValue())
                .increaseToFileWidth(increaseToFileWidth.doubleValue())
                .build();
    }
}
```

calculator.component.html

```
<form [formGroup]="metricsForm" (ngSubmit)="toNextForm()" *ngIf="formCreated">
  <div class="first-calc">
    <div class="input-fields" *ngFor="let inputField of inputMetrics">
      <div class="input-field">
        <div class="span-wrapper">
          <span>{{inputField.text}}</span>
        </div>
        <mat-form-field>
          <mat-label>{{inputField.text}}</mat-label>
          <input
            matInput
            [formControlName]="inputField.name"
            type="number"
            [readonly]="!isEditable">
          <mat-error *ngIf="metricsForm.get(inputField.name).errors?.['required']">Поле
обов'язкове</mat-error>
          <mat-error *ngIf="metricsForm.get(inputField.name).errors?.['min']">Величина має бути
більша за {{findField(inputMetrics, inputField.name).min}}</mat-error>
          <mat-error *ngIf="metricsForm.get(inputField.name).errors?.['max']">Величина має бути
менша за {{findField(inputMetrics, inputField.name).max}}</mat-error>
        </mat-form-field>
      </div>
      <mat-divider></mat-divider>
    </div>
    <div class="submit-button-wrapper" *ngIf="!isMetricsFormFilled && isEditable">
      <button mat-raised-button color="primary" type="submit" [disabled]="!isEditable">Далі</button>
    </div>
  </div>
</form>
<mat-divider *ngIf="isMetricsFormFilled"></mat-divider>
<form [formGroup]="increasesForm" (ngSubmit)="calculate()">
  <div id="scndCalc" class="second-calc" *ngIf="isMetricsFormFilled">
    <div class="input-fields" *ngFor="let inputField of inputIncreases">
      <div class="input-field">
        <div class="span-wrapper">
          <span>{{inputField.text}}</span>
        </div>
        <mat-form-field>
          <mat-label>{{inputField.text}}</mat-label>
          <input matInput [formControlName]="inputField.name" type="number"
[readonly]="!isEditable">
          <mat-error *ngIf="increasesForm.get(inputField.name).errors?.['required']">Поле
обов'язкове</mat-error>
          <mat-error *ngIf="increasesForm.get(inputField.name).errors?.['min']">Величина має бути
більша за {{findField(inputIncreases, inputField.name).min}}</mat-error>
          <mat-error *ngIf="increasesForm.get(inputField.name).errors?.['max']">Величина має бути
менша за {{findField(inputIncreases, inputField.name).max}}</mat-error>
        </mat-form-field>
      </div>
      <mat-divider></mat-divider>
    </div>
    <div class="submit-button-wrapper" *ngIf="isEditable">
      <button mat-raised-button color="primary" type="submit"
[disabled]="isCalculating">Далі</button><mat-spinner *ngIf="isCalculating"></mat-spinner>
    </div>
  </div>
</form>
<div class="flex-center pattern-img" *ngIf="productMetrics">
  <img [src]="patternImgPath" alt="лекала">
</div>
<app-calc-res-fields></app-calc-res-fields>
```

Б.3. Моделі пальто

CoatModel.java

```
@Entity
@Table(name = "coat_model")
@Getter
@Setter
@Builder
```

```
@AllArgsConstructor
@NoArgsConstructor
public class CoatModel {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "name")
    private String name;

    @Column(name = "price")
    private Double price;

    @Column(name = "img_path")
    private String imgPath;

    @Column(name = "coat_type")
    @Enumerated(value = EnumType.STRING)
    private CoatType coatType;

    @Column(name = "video_url")
    private String videoUrl;

    @OneToMany(mappedBy = "coatModel", cascade = CascadeType.REMOVE)
    private List<Review> reviews;
}
```

CoatModelController.java

```
@RestController
@RequestMapping("/coat-models")
@RequiredArgsConstructor
public class CoatModelController {
    private final CoatModelService coatModelService;

    @GetMapping
    public ResponseEntity<List<CoatModelResponse>> getAllCoatModels(){
        return ResponseEntity.ok(coatModelService.getAllCoatModels());
    }

    @GetMapping("/{id}/images")
    public ResponseEntity<Map<Long, byte[]>> getAllOrderImagesForCoatModel(@PathVariable Long id){
        return ResponseEntity.ok(coatModelService.getAllOrderImagesForCoatModel(id));
    }

    @PostMapping
    @Secured("EMPLOYEE")
    public ResponseEntity<CoatModelResponse> createCoatModel(@RequestBody @Valid CoatModelRequest
request) {
        return ResponseEntity
                .status(HttpStatus.CREATED)
                .body(coatModelService.createCoatModel(request));
    }

    @PatchMapping("/{id}/image")
    @Secured("EMPLOYEE")
    public ResponseEntity<Void> attachImageToCoatModel(
            @PathVariable Long id, @RequestParam("file") MultipartFile file
    ) throws IOException {
        coatModelService.attachImageToCoatModel(id, file);
        return ResponseEntity
                .status(HttpStatus.OK)
                .build();
    }

    @PutMapping("/{id}")
    @Secured("EMPLOYEE")
    public ResponseEntity<CoatModelResponse> updateCoatModel(
            @PathVariable Long id, @RequestBody @Valid CoatModelRequest request
    ){
        return ResponseEntity.ok(coatModelService.updateCoatModel(id, request));
    }

    @DeleteMapping("/{id}")
    @Secured("EMPLOYEE")
    public ResponseEntity<Void> deleteCoatModel(@PathVariable Long id) throws IOException {
        coatModelService.deleteCoatModel(id);
        return ResponseEntity
                .status(HttpStatus.OK)
```

```
                .build();
    }
}
```

CoatModelService.java

```
@Service
@RequiredArgsConstructor
public class CoatModelService {
    private final CoatModelRepository coatModelRepository;
    private final CoatModelMapper coatModelMapper;
    private final OrderRepository orderRepository;
    private final CoatModelImageService coatModelImageService;
    private final ImageService imageService;

    public List<CoatModelResponse> getAllCoatModels() {
        return coatModelRepository.findAll().stream()
                .filter(coatModel -> coatModel.getImgPath() != null)
                .map(coatModelMapper::entityToResponse)
                .toList();
    }

    public Map<Long, byte[]> getAllOrderImagesForCoatModel(Long id) {
        CoatModel coatModel = coatModelRepository.findById(id)
                .orElseThrow(() -> new ApplicationException(COAT_MODEL_NOT_FOUND, id));
        List<Order> orders = orderRepository.findAllByCoatModel(coatModel);
        CheckedFunction1<Order, byte[]> mapImageFunction =
                order -> imageService.extractImage(order.getImgPath());

        return orders.stream()
                .filter(order -> order.getImgPath() != null)
                .collect(Collectors.toMap(Order::getId, mapImageFunction.unchecked()));
    }

    public CoatModelResponse createCoatModel(CoatModelRequest request) {
        CoatModel coatModel = CoatModel.builder()
                .name(request.name())
                .coatType(request.coatType())
                .price(request.price())
                .videoUrl(request.videoUrl())
                .build();
        return coatModelMapper.entityToResponse(coatModelRepository.save(coatModel));
    }

    public void attachImageToCoatModel(Long id, MultipartFile file) throws IOException {
        CoatModel coatModel = coatModelRepository.findById(id)
                .orElseThrow(() -> new ApplicationException(COAT_MODEL_NOT_FOUND, id));
        String imgPath = coatModelImageService.saveImageForCoatModel(coatModel, file);
        System.out.println(imgPath);
        coatModel.setImgPath(imgPath);
        coatModelRepository.save(coatModel);
    }

    public CoatModelResponse updateCoatModel(Long id, CoatModelRequest request) {
        CoatModel coatModel = coatModelRepository.findById(id)
                .orElseThrow(() -> new ApplicationException(COAT_MODEL_NOT_FOUND, id));

        coatModel.setCoatType(request.coatType());
        coatModel.setName(request.name());
        coatModel.setVideoUrl(request.videoUrl());
        coatModel.setPrice(request.price());

        return coatModelMapper.entityToResponse(coatModelRepository.save(coatModel));
    }

    public void deleteCoatModel(Long id) throws IOException {
        CoatModel coatModel = coatModelRepository.findById(id)
                .orElseThrow(() -> new ApplicationException(COAT_MODEL_NOT_FOUND, id));
        List<Order> associatedOrders = orderRepository.findAllByCoatModel(coatModel);
        if(!associatedOrders.isEmpty())
            throw new ApplicationException(CANNOT_DELETE_COAT_MODEL, id, associatedOrders.size());

        imageService.removeImageIfPresent(coatModel.getImgPath());
        coatModelRepository.delete(coatModel);
    }
}
```

category.component.html

```
<div *ngIf="category && !hidden">
  <div class="category-images">
    <div class="models-grid">
      <div
        class="model"
        *ngFor="let model of category.models"
        [ngClass]="{selected: selectedModel?.id === model.id}"
        (click)="onModelSelect(model)"
      >
        <img [src]="' ' + model.image" [alt]="category.text">
        <span>{{model.name}}</span>
        <div class="video-bubble bubble" (click)="openVideo(model.videoUrl!)"><mat-
icon>videocam</mat-icon></div>
        <div class="price-bubble bubble">Ціна: {{model.price}}</div>
        <div class="photo-bubble bubble" [ngClass]="{'selected-bubble': selectedModel?.id ===
model.id && arePhotosShowen}" (click)="onShowPhotos(model.id)"><mat-icon>image</mat-icon></div>
      </div>
      <div class="add-model model" *ngIf="isAdmin()" (click)="onSwitchAddingModel()"
[ngClass]="{'selected-add-model': isAddingModel}">
        <div class="inner-add-model">
          <mat-icon fontIcon="add"></mat-icon>
          <p>Додати модель</p>
        </div>
      </div>
    </div>
  </div>
  <div class="container" [ngStyle]="{'width': '95%'}" *ngIf="user.roles.includes('ADMIN') &&
(selectedModel?.name || isAddingModel)">
    <app-add-model></app-add-model>
  </div>
  <div class="container" *ngIf="selectedModel?.name">
    <app-model-photos *ngIf="arePhotosShowen" [modelId]="selectedModel?.id"></app-model-photos>
    <app-calculator></app-calculator>
    <app-reviews></app-reviews>
  </div>
</div>
```

categories.component.html

```
<div class="spinner-wrapper" *ngIf="!categories">
  <mat-spinner></mat-spinner>
</div>
<div class="categories-wrapper" *ngIf="categories">
  <div class="categories-flex">
    <a
      class="category"
      [ngClass]="{selected: selectedCategory && selectedCategory.coatType === category.coatType}"
      *ngFor="let category of categories"
      [routerLink]="['/', url, category.coatType.toLowerCase()]"
      [routerLinkActive]="['selected']"
      (click)="selectCategory(category)"
      >
      <div class="category-image">
        <img [src]="' ' + getCategoryImage(category)" [alt]="category.text">
      </div>
      <span>{{category.text}}</span>
    </a>
  </div>
</div>
<app-category *ngIf="!isCategoryHidden"></app-category>
```

Б.4. Замовлення

Order.java

```
@Entity
@Table(name = "\"order\"")
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class Order {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

```
    @Column(name = "created_at")
    private ZonedDateTime createdAt;

    @Column(name = "status")
    @Enumerated(EnumType.STRING)
    private OrderStatus status;

    @Column(name = "img_path")
    private String imgPath;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "client_id")
    private Client client;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "employee_id")
    private Employee employee;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "coat_model_id")
    private CoatModel coatModel;

    @OneToOne(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
    @JoinColumn(name = "pattern_data_id")
    private PatternData patternData;

    @OneToOne(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
    @JoinColumn(name = "product_metrics_id")
    private ProductMetrics productMetrics;
}
```

OrderController.java

```
@RestController
@RequestMapping("/orders")
@RequiredArgsConstructor
public class OrderController {
    private final OrderService orderService;
    private final OrderImageService orderImageService;

    @PostMapping
    public ResponseEntity<OrderResponse> createOrder(@RequestBody @Valid OrderRequest request){
        return ResponseEntity.ok(orderService.createOrder(request));
    }

    @GetMapping
    public ResponseEntity<List<OrderResponse>> getAllOrdersOfCurrentUser(){
        return ResponseEntity.ok(orderService.getAllOrders());
    }

    @GetMapping("/{id}")
    public ResponseEntity<OrderResponse> getOrderById(@PathVariable Long id){
        return ResponseEntity.ok(orderService.getOrderById(id));
    }

    @GetMapping("/{id}/metrics")
    public ResponseEntity<ProductMetricsDTO> getProductMetricsOfOrderById(@PathVariable Long id){
        return ResponseEntity.ok(orderService.getProductMetricsOfOrderById(id));
    }

    @GetMapping("/unassigned")
    @Secured("EMPLOYEE")
    public ResponseEntity<List<OrderResponse>> getAllUnassignedOrders(){
        return ResponseEntity.ok(orderService.getAllUnassignedOrders());
    }

    @PutMapping("/{id}")
    public ResponseEntity<OrderResponse> updateOrder(@PathVariable Long id, @RequestBody @Valid
OrderRequest request){
        return ResponseEntity.ok(orderService.updateOrder(id, request));
    }

    @GetMapping("/{orderId}/image")
    @Secured({"EMPLOYEE", "CLIENT"})
    public ResponseEntity<String> getOrderImage(@PathVariable Long orderId) throws IOException {
        return ResponseEntity.ok(orderImageService.getOrderImage(orderId));
    }
```

```
    @PatchMapping("/{id}/image")
    @Secured("EMPLOYEE")
    public ResponseEntity<Void> attachImageToOrder(
            @PathVariable Long id, @RequestParam("file") MultipartFile file
    ) throws IOException {
        orderImageService.attachImageToOrder(id, file);
        return ResponseEntity
                .status(HttpStatus.OK)
                .build();
    }

    @DeleteMapping("/{id}/image")
    @Secured("EMPLOYEE")
    public ResponseEntity<Void> removeImageFromOrder(@PathVariable Long id) throws IOException {
        orderImageService.removeImageFromOrder(id);
        return ResponseEntity
                .status(HttpStatus.OK)
                .build();
    }

    @PatchMapping("/assign/{orderId}")
    @Secured("EMPLOYEE")
    public ResponseEntity<Void> assignEmployeeToOrder(@PathVariable Long orderId){
        orderService.assignEmployeeToOrder(orderId);
        return ResponseEntity
                .status(HttpStatus.OK)
                .build();
    }

    @PatchMapping("/{id}/completed")
    @Secured("EMPLOYEE")
    public ResponseEntity<OrderResponse> markOrderAsCompleted(@PathVariable Long id){
        return ResponseEntity.ok(orderService.markOrderAsCompleted(id));
    }

    @PatchMapping("/{id}/cancel")
    @Secured("CLIENT")
    public ResponseEntity<OrderResponse> cancelOrder(@PathVariable Long id){
        return ResponseEntity.ok(orderService.cancelOrder(id));
    }
}
```

OrderService.java

```
@Service
@RequiredArgsConstructor
public class OrderService {
    private final OrderRepository orderRepository;
    private final PatternCalculatorService patternCalculatorService;
    private final PatternCalculator patternCalculator;
    private final CurrentUserUtil currentUserUtil;
    private final ClientRepository clientRepository;
    private final CoatModelRepository coatModelRepository;
    private final OrderMapper orderMapper;
    private final ProductMetricsService productMetricsService;
    private final ProductMetricsMapper productMetricsMapper;
    private final EmployeeRepository employeeRepository;
    private final OrderValidator orderValidator;

    public OrderResponse createOrder(OrderRequest request) {
        ProductMetrics productMetrics = productMetricsService.save(request.productMetrics());
        PatternData patternData =
patternCalculatorService.calculatePatternDataAndSave(request.productMetrics());
        CoatModel coatModel = coatModelRepository.findById(request.coatModelId())
                .orElseThrow(() -> new ApplicationException(COAT_MODEL_NOT_FOUND,
request.coatModelId()));
        Order order = Order.builder()
                .coatModel(coatModel)
                .productMetrics(productMetrics)
                .createdAt(ZonedDateTime.now())
                .patternData(patternData)
                .build();

        return currentUserUtil.getCurrentUser().hasRole(CLIENT)
                ? createOrderAsClient(order)
                : createOrderAsEmployee(order, request.clientId());
    }
```

```
    private OrderResponse createOrderAsClient(Order order){
        Client client = clientRepository.findByUser(currentUserUtil.getCurrentUser());
        order.setClient(client);
        order.setStatus(OrderStatus.PENDING);
        System.out.println(order.getCreatedAt());
        return orderMapper.entityToResponse(orderRepository.save(order));
    }

    private OrderResponse createOrderAsEmployee(Order order, Long clientId){
        if(clientId == null)
            throw new ApplicationException(CLIENT_ID_REQUIRED);

        Client client = clientRepository.findById(clientId)
                .orElseThrow(() -> new ApplicationException(CLIENT_NOT_FOUND, clientId));
        order.setClient(client);

        Order savedOrder = orderRepository.save(order);
        assignEmployeeToOrder(savedOrder.getId());
        return orderMapper.entityToResponse(savedOrder);
    }

    public List<OrderResponse> getAllOrders() {
        List<Order> orders = this.getAllOrdersDependingOnRole();
        return orders.stream()
                .map(orderMapper::entityToResponse)
                .toList();
    }

    private List<Order> getAllOrdersDependingOnRole(){
        User currentUser = currentUserUtil.getCurrentUser();
        if(currentUser.getRoles().contains(EMPLOYEE)){
            Employee employee = employeeRepository.findByUser(currentUser);
            return orderRepository.findAllByEmployee(employee);
        } else {
            Client client = clientRepository.findByUser(currentUser);
            return orderRepository.findAllByClient(client);
        }
    }


    public OrderResponse getOrderById(Long id) {
        Order order = orderRepository.findById(id)
                .orElseThrow(() -> new ApplicationException(ORDER_NOT_FOUND, id));
        return orderMapper.entityToResponse(order);
    }

    public List<OrderResponse> getAllUnassignedOrders() {
        List<Order> orders = orderRepository.findAllUnassignedOrders();
        return orders.stream()
                .map(orderMapper::entityToResponse)
                .toList();
    }

    public OrderResponse updateOrder(Long id, OrderRequest request) {
        Order order = orderRepository.findById(id)
                .orElseThrow(() -> new ApplicationException(ORDER_NOT_FOUND, id));
        orderValidator.validateOrderOwnershipByClient(order);
        orderValidator.assertOrderIsNotCancelled(order);

        ProductMetrics updatedProductMetrics =
productMetricsMapper.dtoToEntity(request.productMetrics());
        updatedProductMetrics.setId(order.getProductMetrics().getId());
        order.setProductMetrics(updatedProductMetrics);

        PatternData updatedPatternData = patternCalculator.doCalculate(request.productMetrics());
        updatedPatternData.setId(order.getPatternData().getId());
        order.setPatternData(updatedPatternData);

        return orderMapper.entityToResponse(orderRepository.save(order));
    }

    public void assignEmployeeToOrder(Long orderId) {
        Employee currentEmployee = employeeRepository.findByUser(currentUserUtil.getCurrentUser());
        Order order = orderRepository.findById(orderId)
                .orElseThrow(() -> new ApplicationException(ORDER_NOT_FOUND, orderId));
        orderValidator.validateBeforeAssignment(order);
        order.setEmployee(currentEmployee);
        order.setStatus(OrderStatus.IN_PROGRESS);
        orderRepository.save(order);
    }
```

```
    public OrderResponse markOrderAsCompleted(Long id) {
        Order order = orderRepository.findById(id)
                .orElseThrow(() -> new ApplicationException(ORDER_NOT_FOUND, id));

        orderValidator.assertOrderIsNotCancelled(order);
        orderValidator.assertOrderIsInProgress(order);
        orderValidator.validateOrderOwnershipByEmployee(order);
        order.setStatus(OrderStatus.COMPLETED);
        return orderMapper.entityToResponse(orderRepository.save(order));
    }

    public OrderResponse cancelOrder(Long id){
        Order order = orderRepository.findById(id)
                .orElseThrow(() -> new ApplicationException(ORDER_NOT_FOUND, id));

        orderValidator.validateOrderOwnershipByClient(order);
        orderValidator.assertOrderIsNotCompleted(order);
        order.setStatus(OrderStatus.CANCELLED);
        return orderMapper.entityToResponse(orderRepository.save(order));
    }

    public ProductMetricsDTO getProductMetricsOfOrderById(Long id) {
        Order order = orderRepository.findById(id)
                .orElseThrow(() -> new ApplicationException(ORDER_NOT_FOUND, id));
        return productMetricsMapper.entityToDto(order.getProductMetrics());
    }
}
```

order.component.html

```
<div class="spinner-wrapper">
  <mat-spinner *ngIf="!order"></mat-spinner>
</div>
<div class="order-description">
  <div class="order-image">
    <img [src]="' ' + order['coatModel'].image" [alt]="order['coatModel']"
*ngIf="order">
  </div>
  <div class="scnd-column">
    <div class="add-image" *ngIf="isEmployee && order && order.status === 'COMPLETED'">
      <span *ngIf="!photoPreview">Нове фото ще не обрано</span>
      <div class="image-preview" *ngIf="photoPreview && photoPreview !== ''">
        <img [src]="photoPreview" alt="Додати фото">
      </div>
      <button mat-raised-button color="primary" *ngIf="photoPreview"
(click)="onSavePhoto()">Зберегти фото</button>
    </div>
  </div>
</div>
<div class="status">
  <button
    mat-raised-button
    *ngIf="order && order.status === 'PENDING' && isEmployee"
    color="primary"
    (click)="onAssignOrder(order)"
  >
    <mat-icon>check</mat-icon>| <span>Прийняти замовлення</span>
  </button>
  <button
    mat-raised-button
    *ngIf="order && order.status === 'IN_PROGRESS' && !user?.roles!.includes('CLIENT')"
    [ngStyle]="{background: 'green', color: 'white'}"
    (click)="onFinishOrder(order)"
  >
    <mat-icon>done_all</mat-icon>|Завершити замовлення
  </button>
  <button
    mat-raised-button
    *ngIf="order && (order.status === 'PENDING' || order.status === 'IN_PROGRESS') &&
user?.roles!.includes('CLIENT')"
    color="warn"
    (click)="onCancelOrder(order)"
  >
    <mat-icon>close</mat-icon>| <span>Скасувати замовлення</span>
  </button>
  <button
      mat-raised-button
      color="warn"
```

```
      *ngIf="order && order.status === 'COMPLETED' && (user?.roles!.includes('EMPLOYEE') ||
user?.roles!.includes('ADMIN')) && hasPhoto()"
      (click)="onRemovePhoto()"
  >
    <mat-icon>close</mat-icon> Видалити фото
  </button>
  <mat-card
    *ngIf="user['roles'] && order && !(!user.roles!.includes('CLIENT') && order['status'] ==
'PENDING')"
    [ngStyle]="{'backgroundColor': order.status === 'CANCELLED' ? '#f44336' : order.status ===
'COMPLETED' ? 'green' : '#3f51b5', 'color': 'white', 'fontFamily': 'Inter, sans-serif', 'padding':
'15px'}"
  >
    Замовлення {{order.status === 'COMPLETED' ? 'завершено' : order.status === 'CANCELLED' ?
'скасовано' : 'виготовляється'}}
  </mat-card>
  <button *ngIf="isEmployee && order  && order.status === 'COMPLETED'" mat-button type="button"
color='primary' (click)="filePicker.click()">Додати фото</button>
  <input type="file" #filePicker (change)="onImagePicked($event)">
</div>
<app-model-photos *ngIf="order && order.status == 'COMPLETED'" [modelId]="order.coatModel.id"
[isClickable]="false" [order]="order"></app-model-photos>
<app-calculator [isEditable]="false"></app-calculator>
```

orders.component.html

```
<div class="orders">
  <mat-form-field>
    <mat-label>Стан замовлення</mat-label>
    <mat-select [(value)]=selectedState (selectionChange)="onCheckStates()">
      <mat-option *ngFor="let state of states" [value]='state.name'>{{state.text}}</mat-option>
    </mat-select>
  </mat-form-field>
  <div class="spinner-wrapper" *ngIf="!displayedOrders">
    <mat-spinner></mat-spinner>
  </div>
  <div class="orders-table" *ngIf="displayedOrders">
    <mat-table [dataSource]="displayedOrders">
      <ng-container matColumnDef="pos">
        <mat-header-cell *matHeaderCellDef>№</mat-header-cell>
        <mat-cell [routerLink]="order.id" routerLinkActive="selected" *matCellDef="let
order">{{order['num']}}</mat-cell>
      </ng-container>
      <ng-container matColumnDef="id">
        <mat-header-cell *matHeaderCellDef>id</mat-header-cell>
        <mat-cell *matCellDef="let order">{{order.num}}</mat-cell>
      </ng-container>
      <ng-container matColumnDef="name">
        <mat-header-cell *matHeaderCellDef>Назва</mat-header-cell>
        <mat-cell [routerLink]="order.id" routerLinkActive="selected" *matCellDef="let
order">{{order['coatModel'].name}}</mat-cell>
      </ng-container>
      <ng-container matColumnDef="date">
        <mat-header-cell *matHeaderCellDef>Дата</mat-header-cell>
        <mat-cell [routerLink]="order.id" routerLinkActive="selected" *matCellDef="let
order">{{order['createdAt']}}</mat-cell>
      </ng-container>
      <ng-container matColumnDef="price">
        <mat-header-cell *matHeaderCellDef>Ціна</mat-header-cell>
        <mat-cell [routerLink]="order.id" routerLinkActive="selected" *matCellDef="let
order">{{order['coatModel']['price']}}</mat-cell>
      </ng-container>
      <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
      <mat-row [ngStyle]="{cursor: 'pointer'}" *matRowDef="let order; columns:
displayedColumns"></mat-row>
    </mat-table>
  </div>
  <div class="no-orders">
    <span *ngIf="displayedOrders?.length === 0">Ще немає замовлень</span>
  </div>
  <mat-divider></mat-divider>
  <router-outlet></router-outlet>
</div>
```

Б.5. Відгуки

Review.java

```
@Entity
@Table(name = "review")
@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
@Builder
public class Review {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "content")
    private String content;

    @Column(name = "rating")
    private Short rating;

    @Column(name = "created_at")
    private LocalDateTime createdAt;

    @ManyToOne
    @JoinColumn(name = "coat_model_id")
    private CoatModel coatModel;

    @ManyToOne
    @JoinColumn(name = "client_id")
    private Client client;

    @OneToOne(mappedBy = "review", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
    @PrimaryKeyJoinColumn
    private ReviewReply reply;
}
```

ReviewController.java

```
@RestController
@RequestMapping("/reviews")
@RequiredArgsConstructor
@Validated
public class ReviewController {
    private final ReviewService reviewService;
    private final ReviewReplyService replyService;

    @GetMapping("/{coatModelId}")
    public ResponseEntity<Page<ReviewResponse>> getReviewsForCoatModel(
            @PathVariable Long coatModelId,
            @RequestParam(value = "page", defaultValue = "0") Integer page
    ) {
        return ResponseEntity.ok(reviewService.getReviewsForCoatModel(coatModelId, page));
    }

    @PostMapping
    @Secured("CLIENT")
    public ResponseEntity<Void> createReview(@RequestBody @Valid ReviewRequest request){
        reviewService.createReview(request);
        return ResponseEntity
                .status(HttpStatus.CREATED)
                .build();
    }

    @PutMapping("/{id}")
    @Secured("CLIENT")
    public ResponseEntity<ReviewResponse> updateReview(
            @PathVariable Long id,
            @RequestParam(value = "content", required = false) String content,
            @RequestParam(value ="rating", required = false) @Min(1) @Max(5) Short rating){
        return ResponseEntity.ok(reviewService.updateReview(id, content, rating));
    }

    @DeleteMapping("/{id}")
    @Secured({"CLIENT", "ADMIN"})
    public ResponseEntity<Void> deleteReview(@PathVariable Long id){
        reviewService.deleteReview(id);
```

```
        return ResponseEntity
                .status(HttpStatus.OK)
                .build();
    }

    @PostMapping("/{reviewId}/reply")
    @Secured("ADMIN")
    public ResponseEntity<ReviewReplyResponse> createReviewReply(@PathVariable Long reviewId,
@RequestParam String content){
        return ResponseEntity
                .status(HttpStatus.CREATED)
                .body(replyService.createReviewReply(reviewId, content));
    }

    @PutMapping("/{reviewId}/reply")
    @Secured("ADMIN")
    public ResponseEntity<ReviewReplyResponse> updateReviewReply(@PathVariable Long reviewId,
@RequestParam String content){
        return ResponseEntity.ok(replyService.updateReviewReply(reviewId, content));
    }

    @DeleteMapping("/{reviewId}/reply")
    @Secured("ADMIN")
    public ResponseEntity<Void> deleteReviewReply(@PathVariable Long reviewId){
        replyService.deleteReviewReply(reviewId);
        return ResponseEntity.ok().build();
    }
}
```

ReviewService.java

```
@Service
@RequiredArgsConstructor
public class ReviewService {
    private final ReviewRepository reviewRepository;
    private final ClientRepository clientRepository;
    private final CurrentUserUtil currentUserUtil;
    private final CoatModelRepository coatModelRepository;
    private final ReviewMapper reviewMapper;
    private final OrderRepository orderRepository;

    @Value("${spring.custom.pagination.page-size}")
    private Integer pageSize;

    public Page<ReviewResponse> getReviewsForCoatModel(Long coatModelId, Integer page) {
        CoatModel coatModel = coatModelRepository.findById(coatModelId)
                .orElseThrow(() -> new ApplicationException(COAT_MODEL_NOT_FOUND, coatModelId));
        Pageable pageable = PageRequest.of(page, pageSize, Sort.by(Sort.Direction.DESC,
"createdAt"));
        Page<Review> reviews = reviewRepository.findAllByCoatModel(coatModel, pageable);
        return reviews.map(reviewMapper::entityToResponse);
    }

    public void createReview(ReviewRequest request) {
        Client client = clientRepository.findByUser(currentUserUtil.getCurrentUser());
        CoatModel coatModel = coatModelRepository.findById(request.coatModelId())
                .orElseThrow(() -> new ApplicationException(COAT_MODEL_NOT_FOUND,
request.coatModelId()));

        assertClientHasOrderForCoatModel(client, coatModel);
        assertClientHasNoReviewsForCoatModel(client, coatModel);

        Review review = Review.builder()
                .client(client)
                .coatModel(coatModel)
                .content(request.content())
                .rating(request.rating())
                .createdAt(LocalDateTime.now())
                .build();
        reviewRepository.save(review);
    }

    public ReviewResponse updateReview(Long id, String content, Short rating) {
        Review review = reviewRepository.findById(id)
                .orElseThrow(() -> new ApplicationException(REVIEW_NOT_FOUND, id));
        validateReviewOwnership(review);

        if(content != null) {
            if(content.isBlank())
```

```
                throw new IllegalArgumentException("content must not be blank string");
            review.setContent(content);
        }
        if(rating != null)
            review.setRating(rating);

        return reviewMapper.entityToResponse(reviewRepository.save(review));
    }

    public void deleteReview(Long id) {
        Review review = reviewRepository.findById(id)
                .orElseThrow(() -> new ApplicationException(REVIEW_NOT_FOUND, id));

        if(currentUserUtil.getCurrentUser().hasRole(Role.CLIENT))
            validateReviewOwnership(review);

        reviewRepository.delete(review);
    }

    private void assertClientHasOrderForCoatModel(Client client, CoatModel coatModel) {
        List<Order> orders = orderRepository.findAllByClientAndCoatModel(client, coatModel);
        boolean hasAtLeastOneCompletedOrder = orders.stream()
                .anyMatch(o -> o.getStatus().equals(OrderStatus.COMPLETED));
        if(!hasAtLeastOneCompletedOrder)
            throw new ApplicationException(CANNOT_CREATE_REVIEW_WITHOUT_ORDER);
    }

    private void assertClientHasNoReviewsForCoatModel(Client client, CoatModel coatModel) {
        Optional<Review> reviewOptional = reviewRepository.findByClientAndCoatModel(client,
coatModel);
        if(reviewOptional.isPresent())
            throw new ApplicationException(CLIENT_ALREADY_HAS_REVIEW_ON_COAT_MODEL,
coatModel.getId());
    }

    private void validateReviewOwnership(Review review) {
        Client client = clientRepository.findByUser(currentUserUtil.getCurrentUser());
        if(!review.getClient().equals(client))
            throw new ApplicationException(NOT_ENTITY_OWNER);
    }
}
```

reviews.component.html

```
<mat-expansion-panel>
  <mat-expansion-panel-header>Відгуки</mat-expansion-panel-header>
  <mat-card-content>
    <span *ngIf="!user.isAuth">
      <a [routerLink]="['/', 'registration']">Зареєструйтеся</a> або <a [routerLink]="['/',
'login']">увійдіть</a>, щоб створити відгук
    </span>
    <span *ngIf="!allowComments && user.isAuth">Немає завершеного замовлення</span>
    <form [formGroup]="reviewForm" (ngSubmit)="onCreateReview()" *ngIf="allowComments">
      <mat-card class="add-review review" *ngIf="user.roles.includes('CLIENT')">
        <div class="card-header">Створити відгук
          <div class="stars">
            <div class="star" *ngFor="let star of stars" (click)="setRating(star.rate);
starX.click()" [ngClass]="{'star-selected': star?.rate <= rating}">
              <svg width="20" height="20" >
                <polygon points="10,1 4,19.8 19,7.8 1,7.8 16,19.8"/>
              </svg>
              <input formControlName="rating" type="radio" name="rating" [value]="star.rate" #starX>
            </div>
          </div>
        </div>
        <mat-card-content>
          <mat-form-field>
            <textarea
              matInput
              cdkTextareaAutosize
              cdkAutosizeMinRows="5"
              name="content"
              formControlName="content"
            ></textarea>
            <mat-error>Поле обов'язкове</mat-error>
          </mat-form-field>
        </mat-card-content>
        <mat-card-actions>
```

```
          <button mat-raised-button *ngIf="review" color="warn" type="button"
(click)="onDeleteReview(review.id)" [disabled]="isReviewFormFreezed">Видалити</button>
          <button mat-raised-button [ngStyle]="{'background': isReviewLoaded ? 'green' : '',
'margin-left': '10px'}" color="primary" type="submit"
[disabled]="isReviewFormFreezed">{{isReviewLoaded ? 'Оновити':'Створити'}}</button>
        </mat-card-actions>
      </mat-card>
    </form>
    <div class="reviews">
      <span *ngIf="!reviews || reviews?.length == 0">Ще немає відгуків</span>
      <mat-card class="review" *ngFor="let review of reviews">
        <div class="card-header">
          {{review.clientFullName}}
          <div class="stars">
            <div class="star" *ngFor="let star of stars" [ngClass]="{'star-selected': star?.rate <=
review?.rating}">
              <svg width="20" height="20" >
                <polygon points="10,1 4,19.8 19,7.8 1,7.8 16,19.8"/>
              </svg>
            </div>
            <button mat-mini-fab color="warn" *ngIf="user.roles.includes('ADMIN')"
(click)="onDeleteReview(review.id)" [ngStyle]="{'margin-left': '10px'}"
[disabled]="isReviewFormFreezed"><mat-icon fontIcon="close"></mat-icon></button>
          </div>
        </div>
        <mat-divider></mat-divider>
        <mat-card-content>
          {{review.content}}
          <span
            class="highlighted-link"
            *ngIf="user.roles?.includes('ADMIN') && reply['id'] !== review['id'] && !review.reply"
            (click)="onAnswer(review.id)"
          >Відповісти</span>
          <span>{{review.createdAt}}</span>
        </mat-card-content>
        <mat-card class="create-reply reply" *ngIf="reply['id'] === review['id'] && !review.reply">
          <div class="card-header reply-header">Адміністратор</div>
          <mat-card-content>
            <mat-form-field>
              <textarea
                matInput
                cdkTextareaAutosize
                cdkAutosizeMinRows="5"
                name="content"
                [(ngModel)]="reply.content"
                required
              ></textarea>
              <mat-error>Поле обов'язкове</mat-error>
            </mat-form-field>
          </mat-card-content>
          <mat-card-actions>
            <button mat-raised-button color="primary" type="submit"
(click)="onReply(review.id)">Відповісти</button>
          </mat-card-actions>
        </mat-card>
        <mat-card class="reply" *ngIf="review.reply" [ngStyle]="{'background': '#f1f1f1'}">
          <div class="card-header reply-header">Адміністратор</div>
          <mat-divider></mat-divider>
          <mat-card-content>{{review.reply.content}}<span>{{review.reply.createdAt}}</span></mat-
card-content>
          <mat-card-actions>
            <button mat-raised-button color="warn" *ngIf="user.roles.includes('ADMIN')"
(click)="onDeleteReply(review.id)" [disabled]="isReviewFormFreezed">Видалити</button>
          </mat-card-actions>
        </mat-card>
      </mat-card>
    </div>
  </mat-card-content>
  <button mat-button *ngIf="!isFirstPage" (click)="prevPage()"><mat-icon
fontIcon="chevron_left"></mat-icon></button>
  <span *ngIf="reviews && reviews.length !== 0" [ngStyle]="{'margin-left': isFirstPage ? '64px' :
'', 'margin-top': '10px', 'display': 'inline-block'}">{{page+1}}</span>
  <button mat-button *ngIf="!isLastPage" (click)="nextPage()"><mat-icon
fontIcon="chevron_right"></mat-icon></button>
</mat-expansion-panel>
```

Б.6. Клієнт (реєстрація, персональні дані, метрики)

Client.java

```
@Entity
@Table(name = "client")
@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
@Builder
public class Client {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "first_name")
    private String firstName;

    @Column(name = "last_name")
    private String lastName;

    @Column(name = "email")
    private String email;

    @OneToOne(fetch = FetchType.LAZY, cascade = CascadeType.REMOVE)
    @JoinColumn(name = "user_id")
    private User user;

}
```

ClientController.java

```
@RestController
@RequestMapping("/clients")
@RequiredArgsConstructor
public class ClientController {
    private final ClientService clientService;

    @PostMapping("/register")
    public ResponseEntity<ClientResponse> registerClient(@RequestBody @Valid
ClientRegistrationRequest request) throws IOException {
        return ResponseEntity
                .status(HttpStatus.CREATED)
                .body(clientService.registerClient(request));
    }

    @PostMapping("/register/resend-email")
    public ResponseEntity<Void> resendRegistrationConfirmationEmail(@RequestBody
ClientRegistrationRequest request) throws IOException {
        clientService.resendEmailRegistrationConfirmationEmail(request);
        return ResponseEntity.ok().build();
    }
}
```

ClientService.java

```
@Service
@RequiredArgsConstructor
public class ClientService {
    private final ClientRepository clientRepository;
    private final UserService userService;
    private final CurrentUserUtil currentUserUtil;
    private final ClientMapper clientMapper;
    private final UniqueEmailValidator uniqueEmailValidator;

    public ClientResponse registerClient(ClientRegistrationRequest request) throws IOException {
        uniqueEmailValidator.assertEmailIsUnique(request.email());
        Client client = Client.builder()
                .firstName(request.firstName())
                .lastName(request.lastName())
                .email(request.email())
                .build();

        if(shouldCreateUserForClient()){
            User user = userService.createUser(request, Set.of(Role.CLIENT));
```

```
            client.setUser(user);
        }

        return clientMapper.entityToResponse(clientRepository.save(client));
    }

    private boolean shouldCreateUserForClient(){
        if(currentUserUtil.hasLoggedUser()){
            if(currentUserUtil.getCurrentUser().hasRole(Role.EMPLOYEE))
                return false;
            else
                throw new AuthorizationServiceException("You don't have authority required for this
action");
        }
        return true;
    }

    public void resendEmailRegistrationConfirmationEmail(ClientRegistrationRequest request) throws
IOException {
        userService.resendRegistrationConfirmationEmail(request);
    }
}
```

ClientDetails.java

```
@Entity
@Table(name = "client_details")
@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
public class ClientDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @MapsId
    @OneToOne
    @JoinColumn(name = "client_id")
    @JsonIgnore
    private Client client;

    @Column(name = "birth_date")
    private LocalDate birthDate;

    @Column(name = "phone_number")
    private String phoneNumber;

    @OneToOne(fetch = FetchType.LAZY, cascade = {CascadeType.REMOVE, CascadeType.MERGE,
CascadeType.PERSIST})
    @JoinColumn(name = "address_id")
    private Address address;
}
```

ClientDetailsController.java

```
@RestController
@RequestMapping("/clients/details")
@RequiredArgsConstructor
@Secured("CLIENT")
public class ClientDetailsController {
    private final ClientDetailsService clientDetailsService;

    @GetMapping
    public ResponseEntity<ClientDetailsResponse> getClientDetailsByCurrentUser(){
        return ResponseEntity.ok(clientDetailsService.getClientDetailsByCurrentUser());
    }

    @PostMapping
    public ResponseEntity<Void> createClientDetailsForCurrentUser(@RequestBody @Valid
ClientDetailsRequest request){
        clientDetailsService.createClientDetailsForCurrentUser(request);
        return ResponseEntity
                .status(HttpStatus.CREATED)
                .build();
    }
```

```
    @PutMapping
    public ResponseEntity<ClientDetailsResponse> updateClientDetailsForCurrentUser(@RequestBody
@Valid ClientDetailsRequest request){
        return ResponseEntity.ok(clientDetailsService.updateClientDetailsForCurrentUser(request));
    }
}
```

ClientDetailsService.java

```
@Service
@RequiredArgsConstructor
public class ClientDetailsService {
    private final ClientDetailsRepository clientDetailsRepository;
    private final CurrentUserUtil currentUserUtil;
    private final ClientRepository clientRepository;
    private final ClientDetailsMapper clientDetailsMapper;

    public void createClientDetailsForCurrentUser(ClientDetailsRequest request) {
        Client client = clientRepository.findByUser(currentUserUtil.getCurrentUser());
        if(clientDetailsRepository.existsById(client.getId()))
            throw new ApplicationException(CLIENT_DETAILS_ALREADY_EXIST);
        ClientDetails clientDetails = clientDetailsMapper.requestToEntity(request);
        clientDetails.setClient(client);
        clientDetailsRepository.save(clientDetails);
    }

    public ClientDetailsResponse getClientDetailsByCurrentUser() {
        Client client = clientRepository.findByUser(currentUserUtil.getCurrentUser());
        ClientDetails clientDetails = clientDetailsRepository.findById(client.getId())
                .orElseThrow(() -> new ApplicationException(CLIENT_DETAILS_NOT_FOUND));
        return clientDetailsMapper.entityToResponse(clientDetails);
    }

    public ClientDetailsResponse updateClientDetailsForCurrentUser(ClientDetailsRequest request) {
        Client client = clientRepository.findByUser(currentUserUtil.getCurrentUser());
        ClientDetails clientDetails = clientDetailsRepository.findById(client.getId())
                .orElseThrow(() -> new ApplicationException(CLIENT_DETAILS_NOT_FOUND));
        ClientDetails updatedDetails = this.doUpdate(clientDetails, request);
        return clientDetailsMapper.entityToResponse(clientDetailsRepository.save(updatedDetails));
    }

    private ClientDetails doUpdate(ClientDetails clientDetails, ClientDetailsRequest request) {
        clientDetails.setBirthDate(request.birthDate());
        clientDetails.setPhoneNumber(request.phoneNumber());
        clientDetails.getAddress().setCity(request.address().city());
        clientDetails.getAddress().setStreet(request.address().street());
        clientDetails.getAddress().setBuildingNumber(request.address().buildingNumber());
        clientDetails.getAddress().setApartmentNumber(request.address().apartmentNumber());
        clientDetails.getAddress().setZipCode(request.address().zipCode());

        return clientDetails;
    }
}
```

ClientMetrics.java

```
@Entity
@Table(name = "client_metrics")
@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
public class ClientMetrics {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long clientId;
    @MapsId
    @OneToOne
    @JoinColumn(name = "client_id")
    @JsonIgnore
    private Client client;
    @Column(name = "neck_semi_circumference")
    private Double neckSemiCircumference;
    @Column(name = "chest_semi_circumference_1")
    private Double chestSemiCircumference1;
    @Column(name = "chest_semi_circumference_2")
    private Double chestSemiCircumference2;
    @Column(name = "chest_semi_circumference_3")
```

```
    private Double chestSemiCircumference3;
    @Column(name = "waist_semi_circumference")
    private Double waistSemiCircumference;
    @Column(name = "shoulder_width")
    private Double shoulderWidth;
    @Column(name = "chest_height")
    private Double chestHeight;
    @Column(name = "chest_height_1")
    private Double chestHeight1;

    @Column(name = "back_armhole_height")
    private Double backArmholeHeight;
    @Column(name = "back_length_till_waist")
    private Double backLengthTillWaist;
    @Column(name = "shoulder_height_sidelong")
    private Double shoulderHeightSidelong;
    @Column(name = "chest_width")
    private Double chestWidth;
    @Column(name = "chest_center")
    private Double chestCenter;
    @Column(name = "back_width")
    private Double backWidth;
    @Column(name = "waist_length_front")
    private Double waistLengthFront;
    @Column(name = "neck_base_to_front_waist_line_distance")
    private Double neckBaseToFrontWaistLineDistance;
}
```

ClientMetricsController.java

```
@RestController
@RequestMapping("/clients/metrics")
@RequiredArgsConstructor
@Secured("CLIENT")
public class ClientMetricsController {
    private final ClientMetricsService clientMetricsService;

    @GetMapping
    public ResponseEntity<ClientMetricsResponse> getClientMetricsByCurrentUser(){
        return ResponseEntity.ok(clientMetricsService.getClientMetricsByCurrentUser());
    }

    @PostMapping
    public ResponseEntity<Void> createClientMetricsForCurrentUser(
            @RequestBody @Valid ClientMetricsRequest request
    ){
        clientMetricsService.createClientMetricsForCurrentUser(request);
        return ResponseEntity
                .status(HttpStatus.CREATED)
                .build();
    }

    @PutMapping
    public ResponseEntity<ClientMetricsResponse> updateClientMetricsForCurrentUser(
            @RequestBody @Valid ClientMetricsRequest request
    ){
        return ResponseEntity.ok(clientMetricsService.updateClientMetricsForCurrentUser(request));
    }
}
```

ClientMetricsService.java

```
@Service
@RequiredArgsConstructor
public class ClientMetricsService {
    private final ClientMetricsRepository clientMetricsRepository;
    private final ClientRepository clientRepository;
    private final CurrentUserUtil currentUserUtil;
    private final ClientMetricsMapper clientMetricsMapper;

    public ClientMetricsResponse getClientMetricsByCurrentUser() {
        Client client = clientRepository.findByUser(currentUserUtil.getCurrentUser());
        ClientMetrics clientMetrics = clientMetricsRepository.findById(client.getId())
                .orElseThrow(() -> new ApplicationException(CLIENT_METRICS_NOT_FOUND));
        return clientMetricsMapper.entityToResponse(clientMetrics);
    }
```

```
    public void createClientMetricsForCurrentUser(ClientMetricsRequest request) {
        Client client = clientRepository.findByUser(currentUserUtil.getCurrentUser());
        if(clientMetricsRepository.existsById(client.getId()))
            throw new ApplicationException(CLIENT_METRICS_ALREADY_EXIST);
        ClientMetrics clientMetrics = clientMetricsMapper.requestToEntity(request);
        clientMetrics.setClient(client);
        clientMetricsRepository.save(clientMetrics);
    }

    public ClientMetricsResponse updateClientMetricsForCurrentUser(ClientMetricsRequest request) {
        Client client = clientRepository.findByUser(currentUserUtil.getCurrentUser());
        ClientMetrics clientMetrics = clientMetricsRepository.findById(client.getId())
                .orElseThrow(() -> new ApplicationException(CLIENT_METRICS_NOT_FOUND));
        ClientMetrics updatedMetrics = this.doUpdate(clientMetrics, request);
        return clientMetricsMapper.entityToResponse(clientMetricsRepository.save(updatedMetrics));
    }

    private ClientMetrics doUpdate(ClientMetrics clientMetrics, ClientMetricsRequest request){
        clientMetrics.setNeckSemiCircumference(request.neckSemiCircumference());
        clientMetrics.setNeckSemiCircumference(request.neckSemiCircumference());
        clientMetrics.setChestSemiCircumference1(request.chestSemiCircumference1());
        clientMetrics.setChestSemiCircumference2(request.chestSemiCircumference2());
        clientMetrics.setChestSemiCircumference3(request.chestSemiCircumference3());
        clientMetrics.setWaistSemiCircumference(request.waistSemiCircumference());
        clientMetrics.setShoulderWidth(request.shoulderWidth());
        clientMetrics.setChestHeight(request.chestHeight());
        clientMetrics.setChestHeight1(request.chestHeight1());
        clientMetrics.setBackArmholeHeight(request.backArmholeHeight());
        clientMetrics.setBackLengthTillWaist(request.backLengthTillWaist());
        clientMetrics.setShoulderHeightSidelong(request.shoulderHeightSidelong());
        clientMetrics.setChestWidth(request.chestWidth());
        clientMetrics.setChestCenter(request.chestCenter());
        clientMetrics.setBackWidth(request.backWidth());
        clientMetrics.setWaistLengthFront(request.waistLengthFront());

clientMetrics.setNeckBaseToFrontWaistLineDistance(request.neckBaseToFrontWaistLineDistance());

        return clientMetrics;
    }
}
```

registration.component.html

```
<mat-spinner *ngIf="isLoading"></mat-spinner>
<mat-card xmlns="http://www.w3.org/1999/html" [ngStyle]="{'display': isLoading ? 'none' : 'block'}">
  <form #f="ngForm" (submit)="onRegister(f)">
  <mat-card-header>
    <mat-card-title>Реєстрація</mat-card-title>
    <mat-card-subtitle>Будь ласка, зареєструйтеся, якщо ви тут вперше, або <a [routerLink]="['/',
'login']">увійдіть</a></mat-card-subtitle>
  </mat-card-header>
  <mat-card-content>
    <div class="names">
      <mat-form-field>
        <mat-label>Прізвище</mat-label>
        <input
          matInput
          type="text"
          name="lastName"
          ngModel
          required
          #surname = "ngModel">
        <mat-error *ngIf="surname.hasError('required')">
          Введіть прізвище
        </mat-error>
      </mat-form-field>
      <mat-form-field>
        <mat-label>Ім'я</mat-label>
        <input
          matInput
          type="text"
          name="firstName"
          ngModel
          required
          #name = "ngModel">
        <mat-error *ngIf="name.hasError('required')">
          Введіть ім'я
        </mat-error>
      </mat-form-field>
```

```
    </div>
    <mat-form-field>
      <mat-label>Пошта</mat-label>
      <input
        matInput
        type="email"
        name="email"
        ngModel
        required
        email
        #email = "ngModel">
      <mat-error *ngIf="email.hasError('required')">
        Введіть пошту
      </mat-error>
      <mat-error *ngIf="email.hasError('email')">
        Неправильна пошта
      </mat-error>
    </mat-form-field>
    <mat-form-field>
      <mat-label>Пароль</mat-label>
      <input
        matInput
        [type]="hide1 ? 'password' : 'text'"
        name="password"
        ngModel
        required
        #password = "ngModel">
      <button mat-icon-button matSuffix type="button" (click)="hide1 = !hide1" [attr.aria-
label]="'Сховати пароль'">
        <mat-icon>{{hide1 ? 'visibility_off' : 'visibility'}}</mat-icon>
      </button>
      <mat-error *ngIf="password.hasError('required')">
        Введіть пароль
      </mat-error>
    </mat-form-field>
    <mat-form-field>
      <mat-label>Підтвердити пароль</mat-label>
      <input
        matInput
        [type]="hide2 ? 'password' : 'text'"
        name="passwordRepeat"
        ngModel
        required
        [appSamePasswords]="password"
        #passwordRepeat = "ngModel">
      <button mat-icon-button matSuffix type="button" (click)="hide2 = !hide2" [attr.aria-
label]="'Сховати пароль'">
        <mat-icon>{{hide2 ? 'visibility_off' : 'visibility'}}</mat-icon>
      </button>
      <mat-error *ngIf="passwordRepeat.hasError('required')">
        Введіть пароль
      </mat-error>
      <mat-error
        *ngIf="passwordRepeat.hasError('notSamePass') &&
               password.hasError('notSamePass') &&
               !passwordRepeat.hasError('required')">
        Паролі не співпадають
      </mat-error>
    </mat-form-field>
  </mat-card-content>
  <mat-card-actions>
    <button class="register-button" *ngIf="!isRegistered" mat-raised-button type="submit"
color="primary">Зареєструватися</button>
    <div class="repeat-email">
      <span>Не надійшов лист?</span><button mat-button type="button" color="primary"
[disabled]="leftSecondsToRepeatEmail > 0" (click)="onRegister(f)">Вислати знову</button>
      <div class="repeat-interval" [ngStyle]="{'position': 'relative'}"
*ngIf="leftSecondsToRepeatEmail > 0">
        <mat-spinner [ngStyle]="{'margin': '0 auto', 'max-height': '35px'}"></mat-spinner>
        <span [ngStyle]="{'position': 'absolute', 'left': '47%', 'top': '20%'}">
        {{leftSecondsToRepeatEmail}}
      </span>
      </div>
    </div>
    </mat-card-actions>
  </form>
</mat-card>
```

user-details.component.html

```
<mat-card>
  <form [formGroup]="userForm" (ngSubmit)="onSubmit()" *ngIf="!isLoading">
    <mat-card-header>Персональні дані</mat-card-header>
    <mat-card-content>
      <div class="field">
        <span>Дата народження</span>
        <mat-form-field>
          <mat-label>Дата народження</mat-label>
          <input
            [matDatepicker]="picker"
            name="birthDate"
            required
            matInput
            formControlName="birthDate">
          <mat-datepicker-toggle matIconSuffix [for]="picker"></mat-datepicker-toggle>
          <mat-datepicker #picker disabled="false"></mat-datepicker>
          <mat-error>Поле обов'язкове</mat-error>
        </mat-form-field>
      </div>
      <div class="field">
        <span>Номер телефону</span>
        <mat-form-field>
          <mat-label>Номер телефону</mat-label>
          <input
            name="phoneNumber"
            type="number"
            required
            matInput
            formControlName="phoneNumber">
          <mat-error>Поле обов'язкове</mat-error>
        </mat-form-field>
      </div>
      <mat-divider></mat-divider>
    </mat-card-content>
    <mat-card-header>Адреса</mat-card-header>
    <mat-card-content formGroupName="address">
      <div class="field" *ngFor="let addressField of newAddress; let i = index">
        <span>{{addressField.text}}</span>
        <mat-form-field>
          <mat-label>{{addressField.text}}</mat-label>
          <input
            [name]="addressField.name"
            [required]="addressField.name !== 'apartmentNumber'"
            [formControlName]="addressField.name"
            matInput>
          <mat-error>Поле обов'язкове</mat-error>
        </mat-form-field>
      </div>
    </mat-card-content>
    <mat-card-actions>
      <button mat-raised-button color="primary" type="submit">Зберегти</button>
    </mat-card-actions>
  </form>
</mat-card>
```

Б.7. Працівники ательє

Employee.java

```
@Entity
@Table(name = "employee")
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "first_name")
    private String firstName;

    @Column(name = "last_name")
    private String lastName;

    @Column(name = "email")
```

```
    private String email;

    @Column(name = "phone_number")
    private String phoneNumber;

    @Column(name = "registered_at")
    private LocalDateTime registeredAt;

    @OneToOne(fetch = FetchType.LAZY, cascade = CascadeType.REMOVE)
    @JoinColumn(name = "user_id")
    private User user;

    @OneToMany(mappedBy = "employee", fetch = FetchType.LAZY)
    private List<Order> orders;
}
```

EmployeeController.java

```
@RestController
@RequestMapping("/employees")
@Secured("ADMIN")
@RequiredArgsConstructor
public class EmployeeController {
    private final EmployeeService employeeService;

    @GetMapping
    public ResponseEntity<List<EmployeeResponse>> getAllEmployees(){
        return ResponseEntity.ok(employeeService.getAllEmployees());
    }

    @PostMapping("/register")
    public ResponseEntity<EmployeeResponse> registerEmployee(@RequestBody
EmployeeRegistrationRequest request) throws IOException {
        return ResponseEntity
                .status(HttpStatus.CREATED)
                .body(employeeService.registerEmployee(request));
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> dismissEmployee(@PathVariable Long id){
        employeeService.dismissEmployee(id);
        return ResponseEntity.ok().build();
    }
}
```

EmployeeService.java

```
@Service
@RequiredArgsConstructor
public class EmployeeService {
    private final EmployeeRepository employeeRepository;
    private final EmployeeMapper employeeMapper;
    private final UserService userService;
    private final ClientRepository clientRepository;

    public List<EmployeeResponse> getAllEmployees(){
        return employeeRepository.findAll().stream()
                .filter(e -> !e.getUser().hasRole(Role.ADMIN))
                .filter(e -> e.getUser().getActive())
                .map(employeeMapper::entityToResponse)
                .toList();
    }

    @Transactional
    public EmployeeResponse registerEmployee(EmployeeRegistrationRequest request) throws IOException
{
        Employee employee = employeeMapper.requestToEntity(request);
        if(wasDismissed(employee)){
            return restoreEmployee(employee.getEmail());
        }
        assertEmailIsUnique(request.email());
        User user = userService.createUser(request, Set.of(Role.EMPLOYEE));
        employee.setUser(user);
        employee.setRegisteredAt(LocalDateTime.now());
        return employeeMapper.entityToResponse(employeeRepository.save(employee));
    }
```

```
    public void dismissEmployee(Long id) {
        Employee employee = employeeRepository.findById(id)
                .orElseThrow(() -> new ApplicationException(EMPLOYEE_NOT_FOUND, id));
        assertEmployeeHasNoOrdersInProgress(employee);
        employee.getUser().setActive(false);
        employeeRepository.save(employee);
    }

    private EmployeeResponse restoreEmployee(String email){
        var employee = employeeRepository.findByEmail(email);
        employee.getUser().setActive(true);
        return employeeMapper.entityToResponse(employeeRepository.save(employee));
    }

    private void assertEmailIsUnique(String email){
        if(employeeRepository.existsByEmail(email) || clientRepository.existsByEmail(email))
            throw new ApplicationException(EMAIL_ALREADY_EXISTS, email);
    }

    private boolean wasDismissed(Employee employee){
        if(employeeRepository.existsByEmail(employee.getEmail())){
            var existingEmployee = employeeRepository.findByEmail(employee.getEmail());
            return !existingEmployee.getUser().getActive();
        }
        return false;
    }

    private void assertEmployeeHasNoOrdersInProgress(Employee employee){
        boolean hasOrdersInProgress = employee.getOrders()
                .stream()
                .anyMatch(o -> o.getStatus().equals(OrderStatus.IN_PROGRESS));
        if(hasOrdersInProgress)
            throw new ApplicationException(EMPLOYEE_HAS_UNFINISHED_ORDERS, employee.getId());
    }
}
```

employee.component.html

```
<mat-card>
  <mat-card-header>Дані працівника</mat-card-header>
  <mat-card-content>
    <mat-spinner *ngIf="isLoading" [ngStyle]="{margin: '0
auto'}"></mat-spinner>
    <div class="employee-data" *ngIf="!isLoading">
      <div class="employee-field" *ngFor="let field of
employeeFields">
        <span>{{field.fieldNameTranslation}}</span>
        <span>{{field.fieldValue}}</span>
        <mat-divider></mat-divider>
      </div>
    </div>
  </mat-card-content>
  <mat-card-actions *ngIf="!isLoading">
    <button mat-raised-button color="warn"
(click)="onDeleteEmployee()"><mat-icon>close</mat-
icon>Звільнити</button>
  </mat-card-actions>
</mat-card>
```

employees.component.html

```
<div class="users" >
  <div class="users-table">
    <mat-table [dataSource]="employees" *ngIf="displayedColumns">
      <ng-container matColumnDef="firstName">
        <mat-header-cell *matHeaderCellDef>Ім'я</mat-header-cell>
        <mat-cell *matCellDef="let employee">{{employee.firstName}}</mat-cell>
```

```
      </ng-container>
      <ng-container matColumnDef="lastName">
        <mat-header-cell *matHeaderCellDef>Прізвище</mat-header-cell>
        <mat-cell *matCellDef="let employee">{{employee.lastName}}</mat-cell>
      </ng-container>
      <ng-container matColumnDef="email">
        <mat-header-cell *matHeaderCellDef>Пошта</mat-header-cell>
        <mat-cell *matCellDef="let employee">{{employee.email}}</mat-cell>
      </ng-container>
      <ng-container matColumnDef="phoneNumber">
        <mat-header-cell *matHeaderCellDef>Телефон</mat-header-cell>
        <mat-cell *matCellDef="let employee">{{employee.phoneNumber}}</mat-cell>
      </ng-container>
      <ng-container matColumnDef="registeredAt">
        <mat-header-cell *matHeaderCellDef>Зареєстровано</mat-header-cell>
        <mat-cell *matCellDef="let employee">{{employee.registeredAt}}</mat-cell>
      </ng-container>
      <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
      <mat-row [routerLink]="employee.id" routerLinkActive="selected" [ngStyle]="{cursor:
'pointer'}" *matRowDef="let employee; columns: displayedColumns"></mat-row>
    </mat-table>
  </div>
  <div class="no-employees" *ngIf="employees?.length === 0">
    <span>Ще немає працівників</span>
  </div>
  <mat-divider></mat-divider>
  <router-outlet></router-outlet>
  <app-employee-registration></app-employee-registration>
</div>
```

employee-registration.component.html

```
<form #f="ngForm" (ngSubmit)="onRegisterEmployee(f)">
  <mat-card class="register-card">
    <mat-card-header>Зареєструвати нового робітника</mat-card-header>
    <mat-card-content>
      <div class="names">
        <mat-form-field>
          <mat-label>Прізвище</mat-label>
          <input
            matInput
            type="text"
            name="lastName"
            ngModel
            required
            #name = "ngModel">
          <mat-error *ngIf="name.hasError('required')">
            Введіть прізвище
          </mat-error>
        </mat-form-field>
        <mat-form-field>
          <mat-label>Ім'я</mat-label>
          <input
            matInput
            type="text"
            name="firstName"
            ngModel
            required
            #name = "ngModel">
          <mat-error *ngIf="name.hasError('required')">
            Введіть ім'я
          </mat-error>
        </mat-form-field>
      </div>
      <mat-form-field>
        <mat-label>Пошта</mat-label>
        <input
          matInput
          type="email"
          name="email"
          ngModel
          required
          email
          #email="ngModel">
        <mat-error *ngIf="email.hasError('required')">
          Введіть пошту
        </mat-error>
        <mat-error *ngIf="email.hasError('email')">
          Неправильна пошта
```

```
        </mat-error>
      </mat-form-field>
      <mat-form-field>
        <mat-label>Номер телефону</mat-label>
        <input
          matInput
          type="text"
          name="phoneNumber"
          ngModel
          required
        >
      </mat-form-field>
      <mat-form-field>
        <mat-label>Пароль</mat-label>
        <input
          matInput
          [type]="hide ? 'password' : 'text'"
          name="password"
          ngModel
          required
          #password="ngModel">
        <button mat-icon-button matSuffix (click)="hide = !hide" [attr.aria-label]="'Сховати
пароль'">
          <mat-icon>{{hide ? 'visibility_off' : 'visibility'}}</mat-icon>
        </button>
        <mat-error *ngIf="password.hasError('required')">
          Введіть пароль
        </mat-error>
      </mat-form-field>
      <mat-form-field>
        <mat-label>Підтвердити пароль</mat-label>
        <input
          matInput
          [type]="hide ? 'password' : 'text'"
          name="passwordRepeat"
          ngModel
          required
          [appSamePasswords]="password"
          #passwordRepeat="ngModel">
        <button mat-icon-button matSuffix (click)="hide = !hide" [attr.aria-label]="'Сховати
пароль'">
          <mat-icon>{{hide ? 'visibility_off' : 'visibility'}}</mat-icon>
        </button>
        <mat-error *ngIf="passwordRepeat.hasError('required')">
          Введіть пароль
        </mat-error>
        <mat-error
          *ngIf="passwordRepeat.hasError('notSamePass') &&
               password.hasError('notSamePass') &&
               !passwordRepeat.hasError('required')">
          Паролі не співпадають
        </mat-error>
      </mat-form-field>
    </mat-card-content>
    <mat-card-actions>
      <button mat-raised-button color="primary" type="submit">Зареєструвати</button>
    </mat-card-actions>
  </mat-card>
</form>
```