Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
Факультет інформаційних технологій
Кафедра інформаційних технологій та комп'ютерної інженерії

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеня *магістра*

Студентки Піньковської Анастасії Костянтинівни

академічної групи *126м – 22 – 1*

спеціальності *126 «Інформаційні системи та технології»*

на тему: *Розробка інформаційної технології конфігурування компонентів автоматизованої системи виробництва*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		Рейтинговою	**Інституційною**	
кваліфікаційної роботи	*к.т.н., доц. Соколова Н.О.*			
розділів:				
Рецензент	*Клименко А.В.*			
Нормоконтролер	*Коротенко Г.М.*			

Дніпро
2023

ЗАТВЕРДЖЕНО:

завідувач кафедри

Інформаційних технологій та комп'ютерної інженерії

(повна наз ва)

_____ *д.т.н., проф. Гнатушенко В.В.*

(підпис) (прізвище, ініціали)

«_____» _____2023року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня *магістра*

студентки Піньковської А.К. **академічної групи** *126м-22-1*

спеціальності: *126 «Інформаційні системи та технології»*

на тему «*Розробка інформаційної технології конфігурування компонентів автоматизованої системи виробництва»*

затверджену наказом ректора НТУ «Дніпровська політехніка»від *09.10.2023 р. №1227-с*

Розділ	Зміст	Терміни виконання
Розділ 1	Проаналізувати стан області рішення задач	9.10.2023 – 25.10.2023
Розділ 2	Провести огляд існуючих рішень та технологій розробки, проаналізувати варіанти оптимального рішення для розробки власного проекту	25.10.2023 – 15.11.2023
Розділ 3	Реалізувати розробки технології конфігурування компонентів автоматизованої системи	15.11.2023 – 10.12.2023

Завдання видано _____ *доц. Соколова Н.О.*

(підпис) (прізвище, ініціали)

Дата видачі: *09.10.2023 р.*

Дата подання до екзаменаційної комісії:_____

Прийнято до виконання _____ _____

(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 63 с., 28 рис., 6 джерел, 2 додатки.

Об’єкт дослідження: розробка інформаційної технології конфігурування компонентів автоматизованої системи виробництва.

Предмет дослідження: важливість конфігураторів в автоматизованих системах.

Мета роботи полягає в аналізі підходів та розробці інформаційної системи конфігурування автоматизованих систем виробництва.

Використані методі дослідження: теоретичний метод - моделювання, аналіз, порівняння та абстрагування.

У вступі подано стан проблеми, сформульована мета і підхід до виконання даної кваліфікаційної роботи, а також обґрунтована її актуальність та сформовані задачі, які потрібно вирішити.

У першому розділі розглянуто основи предметної області, огляд існуючих рішень, аналіз аналогів та визначення проблем, визначення доцільних методів уникнення проблем аналогів.

У другому розділі були розглянуті інженерні та архітектурні рішення. Були обрани методи розв’язання задачі, а саме: мова програмування та фреймворк.

У третьому розділі розроблено публічну частину додатку (фронт-енд), за допомогою інструментів та використовуючи рішення, обрані у попередніх розділах.

У результаті дослідження та розробки інформаційної технології конфігурування компонентів автоматизованої системи виробництва вдалося досягти поставленої мети та розкрити важливі аспекти використання конфігураторів у сучасних автоматизованих виробничих системах. Використані методи дослідження забезпечили обґрунтованість вибору технологій та ефективність розробленого рішення. Робота має практичне значення для підприємств, що використовують автоматизовані системи виробництва, а також може слугувати основою для подальших досліджень у галузі розробки інформаційних технологій.

АВТОМАТИЗОВАНІ СИСТЕМИ, JAVASCRIPT, NODEJS, REACT, WEB APPLICATION, КОНФІГУРАТОР, FRONT END, BACK END.

ABSTRACT

Explanatory note: 63 pages, 28 figures, 6 sources, 2 attachments.

Research object: development of information technology for configuring components of an automated production system.

Research subject: the importance of configurators in automated systems.

The purpose of the work is to analyze approaches to the development of an information system for configuring automated production systems.

Research methods used: theoretical method - modeling, analysis, comparison and abstraction.

The introduction presents the state of the problem, formulates the goal and approach to the performance of this qualification work, as well as substantiates its relevance and the formulated tasks that need to be solved.

In the first chapter, the basics of the subject area, an overview of existing solutions, analysis of analogues and identification of problems, determination of expedient methods of avoiding problems of analogues are considered.

In the second section, engineering and architectural solutions were considered. Methods of solving the problem were chosen, namely: a programming language and a framework.

In the third section, the public part of the application is developed (front-end), using the tools and using the solutions selected in the previous sections.

As a result of research and development of information technology for configuring components of an automated production system, it was possible to achieve the set goal and reveal important aspects of the use of configurators in modern automated production systems. The research methods used ensured the validity of the choice of technologies and the effectiveness of the developed solution. The work has practical significance for enterprises using automated production systems, and can also serve as a basis for further research in the field of information technology development.

AUTOMATED SYSTEMS, JAVASCRIPT, NODEJS, REACT, WEB APPLICATION, CONFIGURATOR, FRONT END, BACK END.

ЗМІСТ

ВСТУП

Людина завжди шукає нові способи полегшення життя та створення комфорту, і однією з технологічних інновацій, яка реалізує ці прагнення, є розробка інформаційної технології конфігурування (конструктора) компонентів автоматизованої системи виробництва. За останні роки ми стали свідками великого розквіту інтернету речей (IoT), де комп'ютери та сенсори вбудовані у різні пристрої, спрощуючи та автоматизуючи низку процесів.

У наш час інформаційні технології визначають наш повсякденний досвід, і тепер вони входять і в область автоматизованих систем виробництва. Застосування конфігурування компонентів дозволяє створювати індивідуальні рішення для підприємств, враховуючи їх потреби та вимоги. Процес конфігурування, схожий на конструювання, стає важливим етапом у створенні ефективних та оптимізованих систем виробництва.

Згідно з оцінками Інституту McKinsey[1], впровадження автоматизації в різних галузях та завданнях може вести до значного зростання продуктивності. Оцінки вказують на можливість досягнення зростання у розмірі до 1,4% на рік, що створює перспективи для підвищення ефективності та економічного розвитку у виробничих секторах. Ця цифра ілюструє потенційні переваги, які можуть виникнути внаслідок автоматизації та підкреслює важливість розгляду та впровадження нових технологій для оптимізації виробничих процесів.

Відповідно до звіту Оксфордського університету[2], прогнозується, що до 2030 року впровадження автоматизації у секторі обробної промисловості може щорічно призводити до значного зростання в економіці. У доларах США це оцінюється на 4,9 трильйона доларів щорічно. Цей величезний показник ілюструє вплив автоматизації на економічний ландшафт та вказує на значний потенціал зростання вартості виробництва та підвищення продуктивності в даному секторі. Дані цього звіту

підкреслюють важливість та перспективи автоматизації в обробній промисловості як ключового фактора для досягнення економічного розвитку та сталого зростання.

Вибір методів розв'язання задач, розгляд технологій, опис архітектури бекенду та фронтенду — це ключові аспекти, які визначають ефективність та успіх розробки інформаційної технології конфігурування компонентів автоматизованої системи виробництва. Інтеграція різноманітних технологій, врахування потреб користувачів та створення зручного інтерфейсу для взаємодії з системою — це завдання, яке вимагає детального вивчення та професійного підходу.

Так як користувачі та підприємства шукають інноваційні рішення для оптимізації виробництва, розробка інформаційної технології конфігурування компонентів стає важливим етапом у впровадженні сучасних технологій у виробничі процеси. У подальших розділах роботи ми розглянемо деталі цього процесу та вивчимо кожен аспект розробки, спрямований на створення ефективної та гнучкої системи конфігурування компонентів.

РОЗДІЛ 1
ТЕХНОЛОГІЯ КОНФІГУРУВАННЯ КОМПОНЕНТІВ АВТОМАТИЗОВАНОЇ СИСТЕМИ ВИРОБНИЦТВА

1.1 Опис предметної області

Сучасна промисловість стає все більш залежною від автоматизованих систем виробництва, які дозволяють підприємствам не лише оптимізувати виробничі процеси, але й реагувати на зміни у вимогах ринку. Розробка інформаційної технології конфігурування компонентів є ключовою в галузі автоматизації виробництва.

Розробка інформаційних технологій конфігурування спрямована на створення систем, які дозволяють ефективно налаштовувати та інтегрувати різноманітні компоненти в автоматизовану систему виробництва. Це дозволяє підприємствам швидше реагувати на зміни у виробничому процесі та адаптувати їх до нових вимог.

Інформаційні технології конфігурування надають системам виробництва гнучкість і скальованість. Це означає, що вони можуть легко адаптуватися до змін обсягів виробництва, вимог клієнтів та технологічних нововведень, забезпечуючи стабільність та ефективність.

Інформаційні технології конфігурування дозволяють ефективно керувати життєвим циклом компонентів виробничих систем. Вони допомагають відстежувати та оновлювати компоненти, контролювати їхню ефективність та забезпечувати довгий термін служби автоматизованих систем.

Ця галузь також активно працює над інтеграцією з іншими передовими технологіями, такими як штучний інтелект, Інтернет речей та аналіз даних. Це дозволяє створювати високоефективні та інтелектуальні системи виробництва.

В сучасному бізнес-середовищі великі компанії стрімко розвивають свою цифрову інфраструктуру, вдосконалюючи онлайн-сервіси для власних клієнтів. Важливим напрямком цього руху стає розширення можливостей конфігураторів

продуктів, і Porsche, як провідна автомобільна компанія, не залишається осторонь цього тренду. В новому витоку цього еволюційного процесу, в травні 2022 року, Porsche взяло на озброєння нові функції та повністю оновило Porsche Car Configurator. Результат - надзвичайно привабливий цифровий сервіс, спрямований на забезпечення максимальної персоналізації та створення інтуїтивно зрозумілого та захоплюючого досвіду для кожного клієнта. Роберт Адер, CMO Porsche AG, підкреслює, що ця інновація стала ключовим етапом на шляху до надання унікального досвіду покупки Porsche через онлайн-платформу.[3]

Однак, разом із всією перевагою, розробка інформаційних технологій конфігурування стикається з викликами щодо кібербезпеки, стандартизації та потреби навчання персоналу.

В цій предметній області продовжується активна розробка нових інструментів та рішень, спрямованих на створення більш ефективних, гнучких та інтегрованих автоматизованих систем виробництва.

1.2 Огляд існуючих рішень

1.2.1 Опис аналогів

У сфері технології конфігурування компонентів автоматизованої системи виробництва, численні компанії впроваджують інноваційні підходи для оптимізації та індивідуалізації виробничих процесів. В цьому розділі ми детально розглянемо аналоги технології конфігурування, які використовуються в різних галузях промисловості. Від автомобільної до електронної промисловості, ми проаналізуємо різноманітні рішення, спрямовані на вдосконалення виробничих процесів та впровадження ефективної системи конфігурування компонентів. Завдяки цьому огляду ви отримаєте унікальну можливість порівняти та зрозуміти різноманіття технологічних підходів, що визначають сучасний стан ринку автоматизованих виробничих систем.

1) Tesla (автомобільна промисловість)

Tesla дозволяє клієнтам конфігурувати свої електричні автомобілі відповідно до власних уподобань. Використання конфігуратора є частиною процесу замовлення автомобіля (рисунок 1.1). Ось деякі основні функції та етапи конфігуратора Tesla:

- Вибір моделі: Клієнти можуть вибрати бажану модель автомобіля Tesla, таку як Model S, Model 3, Model X або Model Y. Кожна модель має свої унікальні характеристики та можливості.
- Вибір конфігурації батареї та приводу: Клієнти можуть визначити тип батареї (стандартна або довготривала) та тип приводу (задній або повний).
- Вибір кольору: Конфігуратор дозволяє вибрати бажаний кольоровий варіант для кузова автомобіля. Кольорова гамма може змінюватися від часу до часу в залежності від доступності.
- Вибір дизайну коліс: Клієнти можуть вибрати дизайн та розмір коліс для свого автомобіля. Це також може впливати на ефективність електрокара.
- Вибір опцій та аксесуарів: Клієнти можуть вибирати серед різноманітних опцій та аксесуарів, таких як панорамний дах, пакет автопілота, преміальний звуковий пакет, інтер'єр із шкіри та інші.

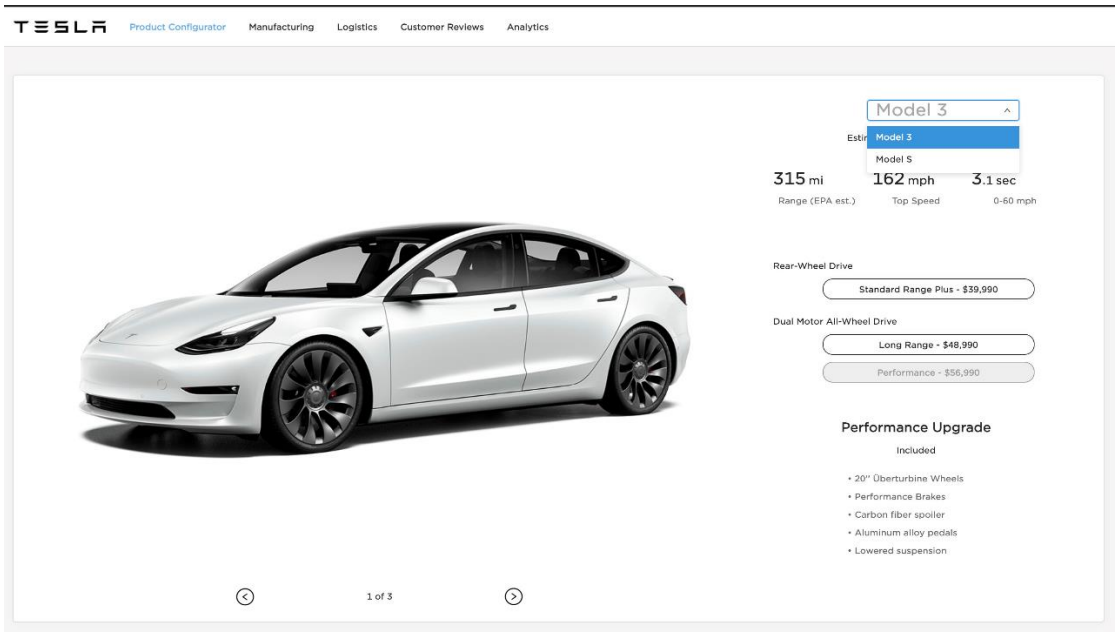

Рисунок 1.1 - Інтерфейс конфігуратора Tesla

- Попередній перегляд та підсумок: Після вибору всіх параметрів, клієнт може переглянути свій автомобіль в інтерактивному режимі, а також перевірити підсумок вартості та обраної конфігурації.
- Оформлення замовлення: Після задоволення конфігурацією, клієнт може розпочати процес замовлення, включаючи введення особистої інформації та вибір способу оплати.

Цей конфігуратор дозволяє клієнтам індивідуалізувати свій автомобіль Tesla відповідно до своїх власних уподобань та вимог, забезпечуючи високий рівень персоналізації для кожного клієнта.

2) Dell (інформаційні технології)

Компанія Dell дозволяє клієнтам створювати власні комп'ютери, вибираючи компоненти, такі як процесор, пам'ять, відеокарта та інші. Це дозволяє користувачам отримати систему, яка повністю відповідає їхнім потребам. Інтерфейс даного конфігуратора представлено на рисунку 1.2.

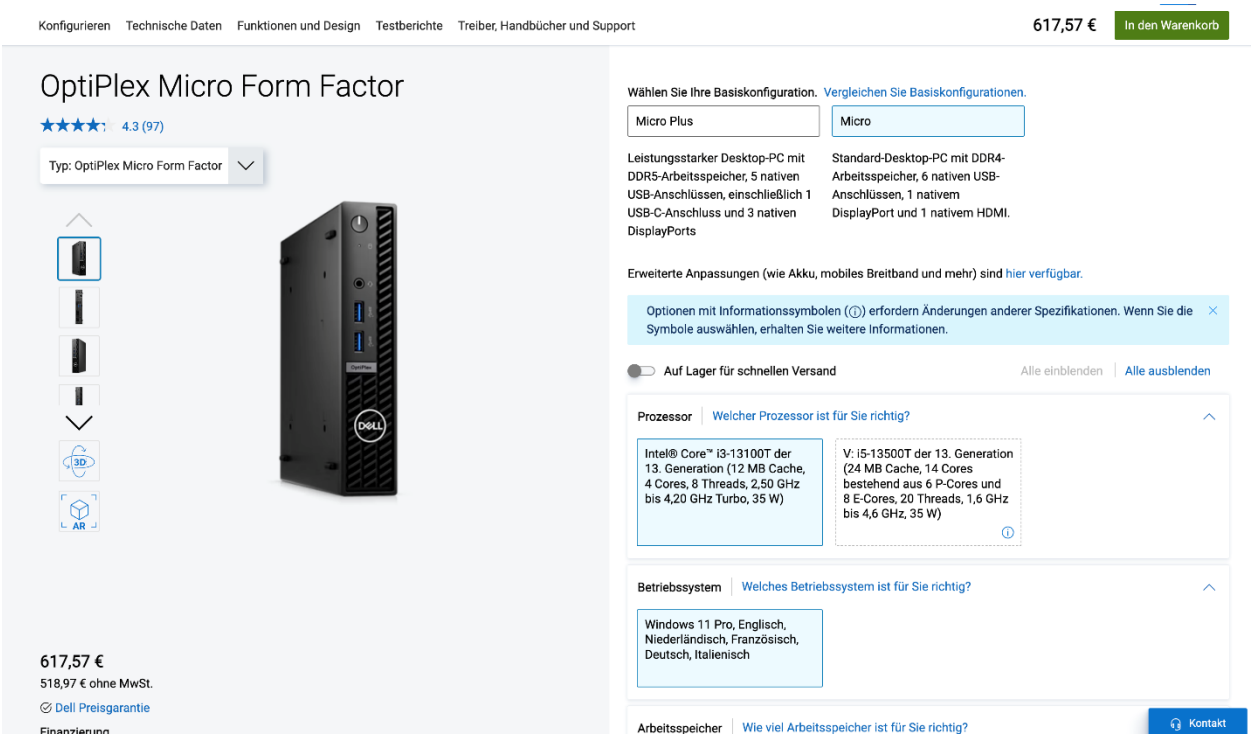

Рисунок 1.2 - Інтерфейс конфігуратора Dell

Деякі ключові елементи конфігуратора Dell:

- Вибір моделі: Клієнти можуть обирати серед різних моделей комп'ютерів, таких як ноутбуки, стаціонарні комп'ютери, робочі станції та інші.
- Процесор та пам'ять: Клієнти можуть вибирати тип процесора (CPU), обсяг оперативної пам'яті (RAM) та інші характеристики.
- Типи зберігання: Вибір можливостей для зберігання даних, таких як твердотільні накопичувачі (SSD) або жорсткі диски (HDD), обсяги пам'яті.
- Графічний адаптер: Вибір відеокарти або графічного адаптера, якщо це необхідно для геймінгу чи важких графічних завдань.
- Дисплей: Вибір типу та розміру дисплея, роздільної здатності та інших параметрів відображення.
- Операційна система та програмне забезпечення: Визначення попередньо встановленої операційної системи та інших програм.
- Інші опції та аксесуари: Вибір додаткових опцій, таких як додаткові порти, клавіатури, миші та інші аксесуари.
- Ціна та підсумок: Клієнти можуть переглядати зміни вартості свого конфігурованого комп'ютера, а також отримувати підсумкову інформацію про обрані параметри.

Після завершення конфігурації, клієнт може здійснити покупку через інтернет або зберегти конфігурацію для майбутнього використання.

3) Nike (спортивний одяг і взуття)

Nike пропонує можливість індивідуалізації взуття через сервіс NIKEiD, де клієнти можуть вибирати дизайн, кольори та матеріали для створення власних кросівок (рисунок 1.3). Зазвичай це стосується моделей кросівок, які підтримують програму NIKEiD.

Основні можливості NIKEiD включають:

- Дизайн верху кросівок: Клієнти можуть вибирати різні матеріали, кольори та дизайни для верху кросівок. Це може включати вибір кольорів для різних частин кросівок, включаючи верх, підошву, шнурівку і так далі.
- Спеціальні елементи дизайну: Додаткові можливості для індивідуалізації можуть включати вибір власного тексту чи ініціалів для вишивки на кросівках.
- Вибір підошви: Деякі моделі дозволяють вибирати тип і кольори підошви.
- Персоналізація знаку "Swoosh": Клієнти можуть вибирати форму, розмір і кольори знаку "Swoosh", який є логотипом Nike.
- Інші деталі: Залежно від моделі і можливостей, клієнти можуть мати можливість вибирати інші деталі, такі як тип шнурівок чи колір підошви.
- Перегляд перед замовленням: Клієнти можуть попередньо переглядати свій дизайн перед оформленням замовлення.

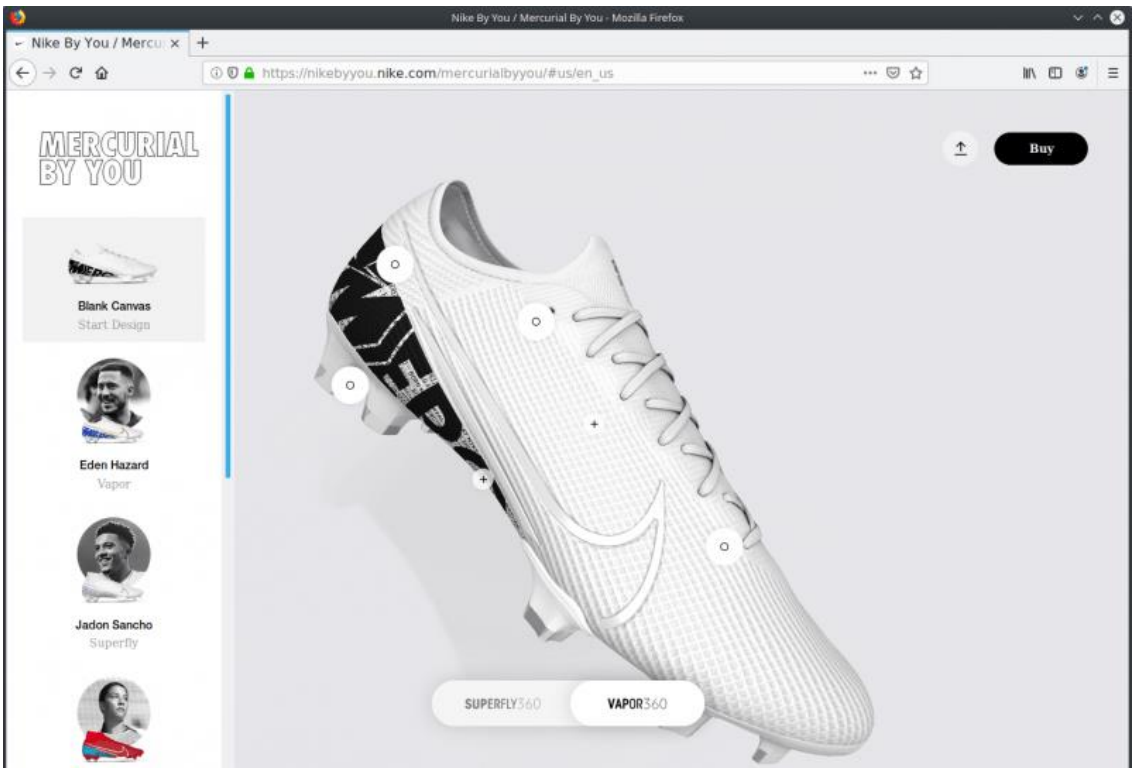

Рисунок 1.3 – Інтерфейс конфігуратора NIKEiD

Такий підхід дозволяє клієнтам створювати унікальний дизайн своїх кросівок, щоб вони відповідали їхнім особистим уподобанням та стилю.

4) HP (електроніка)

HP дозволяє користувачам конфігурувати свої ноутбуки, вибираючи характеристики, такі як тип процесора, обсяг пам'яті, розмір екрану та інші параметри, інтерфейс даного конфігуратора представлений на рисунку 1.4. Конфігуратор HP включає в себе:

- Вибір моделі: Клієнти можуть обирати серед різних моделей ноутбуків чи комп'ютерів, які доступні в асортименті HP.
- Процесор та пам'ять: Вибір типу процесора (CPU), обсяг оперативної пам'яті (RAM) та інших характеристик продукту.
- Тип зберігання даних: Клієнти можуть вибирати між різними опціями для зберігання даних, такими як твердотільні накопичувачі (SSD) або жорсткі диски (HDD).
- Графічний адаптер: Вибір відеокарти або графічного адаптера для тих, хто цінує графічні можливості.
- Дисплей: Вибір розміру та типу дисплея, включаючи дозвіл і технології дисплея.
- Операційна система та програмне забезпечення: Визначення попередньо встановленої операційної системи та наявність програм.
- Інші параметри: Включає в себе такі аспекти, як тип клавіатури, наявність камери, порти та інші додаткові функції.
- Ціна та підсумок: Клієнти можуть переглядати зміни вартості свого конфігурованого пристрою та переглядати підсумкову інформацію про обрані параметри.
- Попередній перегляд: Деякі конфігуратори надають функцію попереднього перегляду, що дозволяє клієнтам переглядати зовнішній вигляд свого пристрою перед замовленням.

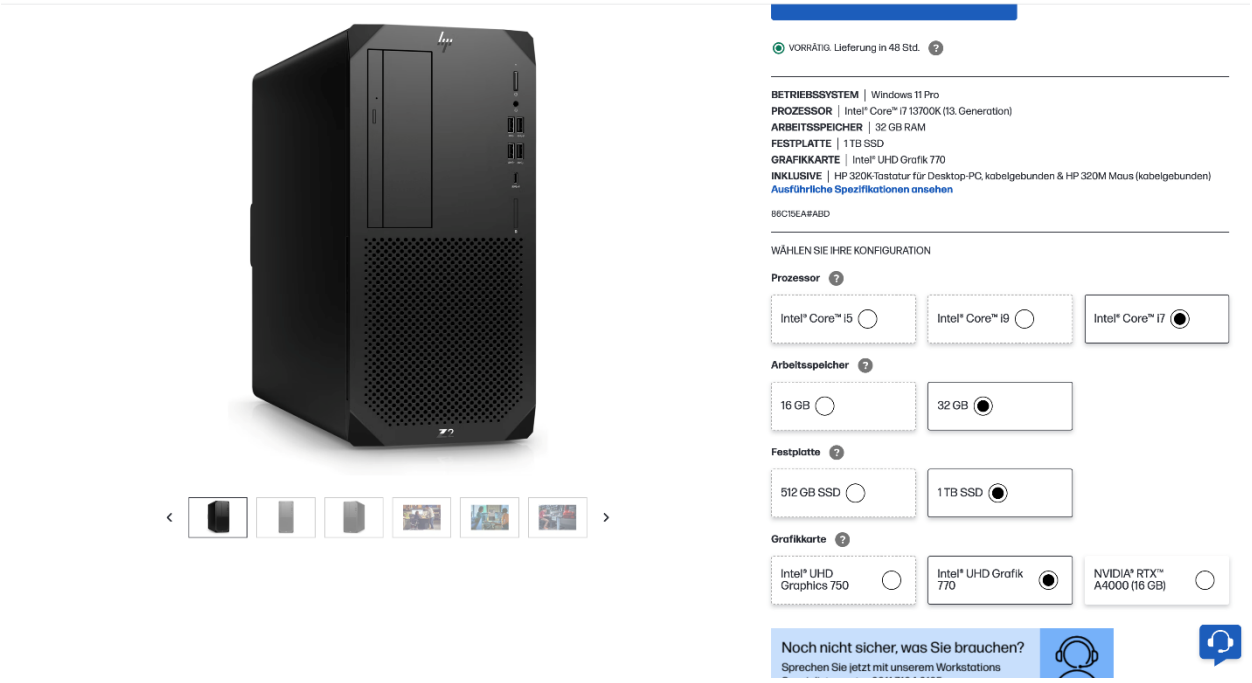

Рисунок 1.4 - Інтерфейс конфігуратора HP 1

Це дозволяє клієнтам індивідуалізувати свої пристрої HP відповідно до своїх потреб та вимог. Важливо перевіряти оновлення на офіційному веб-сайті HP для доступу до найновіших можливостей та моделей.

5) Bespoke Cycling (велосипеди)

Bespoke Cycling може використовувати індивідуалізовані конфігуратори для створення власних велосипедів або велозапчастин для клієнтів (рисунок 1.5).

- Рама: Вибір типу матеріалу для рами (наприклад, карбон, алюміній, сталь), геометрії рами, розмірів та стилізації
- Компоненти: Вибір компонентів, таких як система перемикання швидкостей, гальма, ободи, колеса, сідло, кермо та інші частини велосипеда.
- Трансмісія: Вибір велосипедних шатунів, ланцюга, задньої та передньої перемикачки, касети, шифтерів та інших частин, пов'язаних з трансмісією.
- Кермо та кермова колонка: Вибір типу та форми керма, а також кермової колонки.

- Сідло та підседільний штирь: Вибір сідла та підседільного штиря за зручністю та стилем.
- Колеса та шини: Вибір типу коліс та шин, розмірів та стилю.
- Колір та графіка: Вибір кольору рами, компонентів та іншої графіки або дизайну.
- Персоналізація: Зазвичай, є можливість додаткової персоналізації, такої як гравіювання ім'я або логотипу.

Конфігуратор дозволяє власникам велосипедів отримувати індивідуалізовані продукти, які відповідають їхнім конкретним вимогам і стилю їзди.

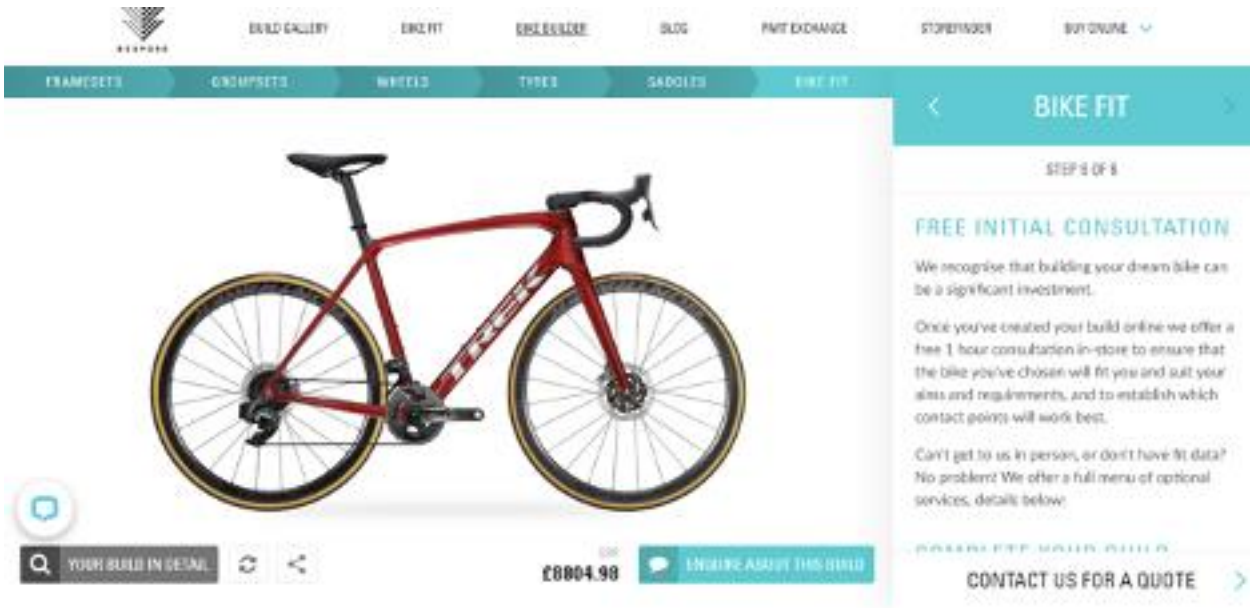

Рисунок 1.4 - Інтерфейс конфігуратора HP 2

1.3 Аналіз аналогів '

1.3.1. Аналіз конфігуратора Tesla

З точки зору програміста, конфігуратор Tesla - це істинна технічна витонченість, що вражає. Спочатку, його висока інтерактивність, завдяки використанню технології WebGL, дозволяє користувачам насолоджуватися 3D-

моделями автомобілів, обертаючи їх на 360 градусів. Це не просто візуалізація, але і зручна та швидка відповідь на кожен крок користувача.

Інтуїтивний дизайн інтерфейсу, забезпечений принципами UX/UI, робить процес конфігурації легким і приємним. Інформація про кожну опцію подається чітко та детально, і користувачі отримують реальний часовий звіт про вартість своїх виборів.

З використанням фронтенд-фреймворків, таких як React або Vue.js, конфігуратор стає динамічним і легко оновлюється без перезавантаження сторінки. Інтеграція з Backend API дозволяє зберігати конфігурації користувачів та спрощує оформлення замовлень.

Проте, навіть з усім технічним вишуканням, конфігуратор Tesla не без своїх викликів. Складність логіки бізнес-правил, особливо при взаємодії різних опцій, може вимагати значних зусиль для розробки та тестування. Також, завантаженість WebGL може відчуватися деякими користувачами, особливо на менш потужних пристроях.

З усім цим, конфігуратор Tesla залишає враження неабиякої технічної майстерності, відображаючи велику увагу до деталей та зручностей для кінцевого користувача.

1.3.2 Аналіз конфігуратора Dell

Дивлячись на конфігуратор Dell, виразно відчувається продуманість технічних рішень. Один з яскравих моментів - це чітко структуроване Backend API, яке дозволяє забезпечити ефективну взаємодію фронтенду та бекенду, що є важливим аспектом розробки.

Використання сучасних фронтенд-фреймворків (наприклад, Angular чи React) робить розробку та управління складною логікою зручними та ефективними завданнями. Також, модульна структура коду дозволяє легко розширювати та модифікувати функціональність, не ламаючи весь додаток.

З використанням бібліотек управління станом (наприклад, Redux), конфігуратор може ефективно керувати станом, що полегшує спільну роботу компонентів. Асинхронні запити до бекенду роблять можливим отримання та оновлення даних без перезавантаження сторінки, що створює динамічний та ефективний інтерфейс.

З іншого боку, конфігуратор Dell має певну кількість недоліків. Споживання ресурсів при великій кількості опцій може становити проблему, зокрема, впливати на пам'ять та пропускну здатність мережі. Забезпечення оновлення даних в реальному часі для всіх користувачів може бути технічно важливим завданням, особливо при великій кількості активних сесій.

Додатково, обмежена можливість скасування змін та залежність від технологій сторонніх розробників створюють свої труднощі у використанні. Додавання нових опцій чи моделей може вимагати переробки коду, що може бути трудомістким завданням.

У підсумку, конфігуратор Dell - це технічно складний, але потужний інструмент, який вимагає балансу між зручністю для користувача та технічною ефективністю.

1.3.3 Аналіз конфігуратора Bespoke Cycling

Розглядаючи конфігуратор Bespoke Cycling з точки зору програміста, видно, що йому вдалося досягти високого ступеня персоналізації для користувачів. Цей конфігуратор дозволяє створити унікальний велосипед, обираючи різні компоненти, матеріали та кольори. Інтерфейс розроблений інтуїтивно зрозуміло, візуалізуючи зміни в реальному часі, що робить процес конфігурації приємним та доступним.

Загалом, конфігуратор вражає деталізованими описами та графічними зображеннями для кожної опції, що допомагає користувачам розуміти вплив кожної компоненти на велосипед. Також, адаптивний дизайн конфігуратора гарантує

комфортне використання на різних пристроях, а швидка реакція на зміни зберігає високий рівень інтерактивності.

Однак, є певні технічні нюанси, на які можна звернути увагу при аналізі. Наприклад, складність у коді може виникнути при великій кількості опцій. Зберігання та синхронізація великої кількості конфігурацій у базі даних може стати завданням, що потребує дбайливого підходу для оптимізації та ефективності.

Також, може виникнути проблема з часом завантаження, особливо при повільному інтернет-з'єднанні, через велику кількість графічних зображень та деталей. Деякі користувачі можуть відчувати нестачу інформації щодо технічних характеристик окремих компонентів, що може вплинути на їх вибір.

Крім того, велика кількість опцій та компонентів може призвести до потреби у системі підтримки та регулярних оновленнях для додавання нових елементів чи виправлення можливих помилок.

У підсумку, Bespoke Cycling спробував вирішити важливий аспект велосипедного ринку, пропонуючи користувачам високий рівень персоналізації. Технічні аспекти вимагають уваги, але з урахуванням користувацької зручності та інноваційного підходу, конфігуратор Bespoke Cycling може стати потужним інструментом для шанувальників велосипедів.

1.4 Визначення доцільних методів уникнення проблем аналогів

Перед тим, як почати пошук методів уникнення проблем, варто визначити загальні переваги та недоліки розглянутих аналогів. Розглядаючи конфігуратори компаній, можна виділити ряд переваг та недоліків. Оцінюючи технічні аспекти конфігураторів різних компаній, можна визначити кілька ключових факторів. Зокрема, важливо відзначити успішне впровадження інтуїтивного дизайну та високого рівня персоналізації. Використання сучасних технологій, таких як React або Vue.js, дозволяє створити динамічний та реактивний інтерфейс, що полегшує взаємодію з користувачем.

Використання WebGL для візуалізації 3D-моделей забезпечує високу якість графіки та інтерактивність.

Щодо систем збереження та обміну конфігураціями, використовуються технології кешування та бази даних для ефективного зберігання і отримання конфігурацій. Розробка API для обміну конфігураціями може забезпечити зручність обміну та взаємодії з іншими користувачами.

Враховуючи інформацію, що була отримана в процесі загального аналізу існуючих рішень, та після виділення недоліків, можна визначити декілька методів уникнення недоліків, щоб при розробці власного продукту врахувати ці аспекти.

Оптимізація фронтенду та бекенду є ключовим методом у вирішенні проблем затримок та споживання ресурсів. Використання асинхронних запитів та кешування служить засобом скорочення часу завантаження сторінки. Завдяки впровадженню Lazy Loading для ресурсів та відтермінюванню завантаження важких елементів до їх реального використання, вдається підвищити продуктивність і поліпшити користувацький досвід.

Розробка гнучкої структури даних із можливістю додавання нових опцій без необхідності змін у вихідному коді є важливим кроком у вирішенні проблеми обмеженої кількості опцій. Використання динамічного підходу до налаштувань виробів робить процес вибору опцій більш гнучким та індивідуалізованим для кожного користувача.

Для полегшення 3D-моделей використовується оптимізація геометрії шляхом зменшення кількості полігонів. Застосування мінімізації текстур дозволяє зменшити розмір файлів та покращити швидкість завантаження. В розгляді альтернатив акцентується на 2D-візуалізації для менш обтяжливих елементів та відтерміньованому завантаженні 3D-моделей лише при реальній потребі користувача.

1.5 Висновки по першому розділу

Виробництво, що спеціалізується на виробництві кабелів, дротів та коннекторів та плетінні автоматизованих систем, ставить перед собою важливе завдання - розробити функціональний та ефективний конфігуратор на власному веб-сайті. З врахуванням попередніх досвідів та уроків, вивчених з подібних інструментів, це завдання вирішується через глибокий і комплексний підхід.

У теперішній час виробництво постає перед проблемою швидкодії свого конфігуратора. З метою уникнення затримок та забезпечення плавної роботи конфігуратора, велика увага приділяється розробці оптимізованого фронтенду та бекенду.

Гнучкість та розширюваність стають ключовими пунктами для розв'язання проблеми обмеження кількості опцій. Створюється гнучка структура даних, що дозволяє додавати нові продукти та їх опції без значних змін у вихідному коді, що відкриває шлях для подальшого розширення.

Великий акцент робиться на візуалізації, намагаючись створити систему, яка надає якісну візуалізацію вибраних опцій та конфігурацій, для поліпшення розуміння клієнтами. Бажано віддавати перевагу 2D-формату для уникнення затримок на сайті.

Забезпечення можливості збереження та повторного виклику конфігурацій додає практичність для користувачів, роблячи весь процес більш зручним і ефективним.

Мобільна доступність стає невід'ємною частиною розробки, оскільки все більше користувачів вибирають мобільні пристрої для використання в Інтернеті. Адаптація конфігуратора до різних мобільних пристроїв стає обов'язковою для зручного використання.

Не менш важливим є здатність конфігуратора підтримувати різні комбінації продукції. Враховуючи широкий асортимент кабелів, дротів та коннекторів,

створення системи, яка здатна охоплювати всі можливі комбінації, є стратегічно важливим завданням.

Усі ці аспекти разом складають комплексний підхід до вирішення завдання та створення конфігуратора, який відповідає високим стандартам ефективності, та гнучкості.

РОЗДІЛ 2
ІНЖЕНЕРІЯ РІШЕНЬ ТА АРХІТЕКТУРА

2.1 Вибір методів розв’язання задач

2.1.1 Огляд та вибір мови програмування

З постійним розвитком інтернет-технологій фронтенд-розробка визначається не лише естетикою інтерфейсу, але і технічною різноманітністю, що ставить перед розробниками завдання забезпечити якісні, швидкі та надійні рішення. Важливо визначити мову програмування, яка відповідатиме вимогам конкретного проєкту.

JavaScript

JavaScript є однією з найпоширеніших мов програмування для веб-розробки, завдяки кільком важливим перевагам. Його велика перевага полягає в тому, що він має широку підтримку усіма сучасними браузерами, що дозволяє використовувати його для створення динамічних та інтерактивних елементів на веб-сторінках. Динамічна природа Javascript, хоча і сприяє швидкості розробки, однак може призводити до потенційних помилок під час виконання програм.

Значущою рисою JavaScript є його багата екосистема, яка нараховує безліч бібліотек та фреймворків. Це робить розробку більш ефективною та прискорює процес створення високоякісних додатків.

Проте, на жаль, існують і деякі недоліки, зокрема, ризик помилок на етапі виконання через динамічну типізацію мови. Така гнучкість може призвести до труднощів у виявленні та виправленні помилок, особливо великих проектах, де може виникнути плутанина типів даних.

Ще однією проблемою є збільшення складності коду великих проектів. Захоплюючись розширенням проекту, JavaScript може стати важкочитаним і складним для управління. Такий стан речей може вимагати додаткових заходів управління кодом для підтримки його легкості та читабельності.

TypeScript

TypeScript як мова програмування, привносить в розробку програмного забезпечення ряд переваг, які видаються особливо цінними в більших та складніших проектах. Однією з найбільш значущих переваг TypeScript є його статична типізація. Це означає, що типи даних визначаються на етапі розробки, що дозволяє знаходити та виправляти помилки на ранніх етапах. Такий підхід значно полегшує виявлення та усунення помилок, що робить код більш надійним та стабільним.

Крім того, TypeScript надає розширені можливості документування коду, що допомагає зрозуміти функціональність та використання різних частин програми. Це стає корисним великим командам розробників та сприяє загальній читабельності проекту.

Однак при всіх своїх перевагах TypeScript має і свої виклики. Інтеграція та оволодіння мовою може вимагати певного часу, особливо для розробників, які раніше працювали з JavaScript. Крім того, використання статичної типізації веде до збільшення обсягу коду через додаткові анотації типів, що може здаватися деяким розробникам додатковим завданням.

Таким чином, вибір TypeScript як мови програмування потребує уважного вивчення та аналізу, але його переваги можуть бути особливо цінними для проектів, де важлива надійність та масштабованість коду.

Dart

Dart представляє собою мову програмування, яка вирізняється деякими важливими перевагами, особливо в контексті розробки мобільних та веб-додатків. Вона славиться високою швидкістю завдяки своєму вбудованому двигуну, що дозволяє ефективно виконувати код та забезпечує рівень продуктивності.

Статична типізація Dart відіграє важливу роль у забезпеченні надійності коду. Визначення типів на етапі розробки сприяє виявленню та усуненню помилок на ранніх етапах, що підвищує стабільність програм та спрощує їх розуміння.

З іншого боку, Dart може стикатися з викликами в певних областях. Однією з них є обмежена екосистема порівняно з іншими мовами, особливо у веб-розробці. Відсутність деяких інструментів та бібліотек може вимагати додаткових зусиль від розробників для забезпечення всіх необхідних функцій.Також слід відзначити, що вивчення та інтеграція Dart може зайняти додатковий час.

2.1.2 Порівняльний аналіз розглянутих мов

Синтаксис та читабельність грають ключову роль у виборі мови програмування. У JavaScript ми маємо простий синтаксис, який, однак, може стати менш читабельним у великих проектах через динамічну типізацію. TypeScript пропонує компроміс: він зберігає простоту JavaScript, але завдяки статичній типізації забезпечує покращену читабельність та надійність коду. Dart, хоча має читабельний синтаксис, стикається з викликами через меншу популярність, що може ускладнити здійснення розробки в умовах обмеженої спільноти.

Щодо продуктивності, JavaScript вражає швидкістю розробки завдяки динамічній типізації, але це може призвести до помилок у великих проектах. TypeScript пропонує статичну типізацію для підвищення продуктивності та забезпечення надійності коду. Dart вражає високою швидкістю виконання, але для вивчення може знадобитися додатковий час.

Спільнота та екосистема грають важливу роль у підтримці мови та розширенні її можливостей. JavaScript має велику та активну спільноту, що підтримує розмаїття екосистеми. TypeScript, розвиваючись з JavaScript, спадкує значну частину цієї спільноти та екосистеми. У Dart спільнота менша, але вона постійно росте, що може вказувати на перспективи його розвитку.

Після ретельного аналізу мов програмування для фронтенд-розробки, JavaScript видається оптимальним вибором для більшості проєктів. З його динамічною природою та широкою підтримкою браузерів, JavaScript забезпечує швидку розробку та гнучкість, що важливо для веб-проєктів.

2.1.3 Розгляд фреймворків обраної мови програмування

Фреймворки JavaScript для створення веб-додатків, на даний час, мають велике значення для розробників. Ці інструменти визначають архітектурні принципи, спрощують розробку та підтримку коду, роблячи її більш ефективною.

React.js

React - це бібліотека JavaScript, призначена для розробки інтерфейсів користувача, яка базується на принципах компонентної архітектури та використовує декларативний підхід до опису взаємодії між елементами інтерфейсу. Розроблений компанією Facebook, React спрощує процес створення складних веб-додатків шляхом введення віртуального DOM, що дозволяє ефективно оновлювати інтерфейс за умови зміни стану додатку. Ключовим елементом React є його фокус на компонентах, які можна складати та використовувати повторно, сприяючи модульності та підтримці чистоти коду. Відзначається великою популярністю у розробників.

React володіє потужною здатністю змінювати підхід до проектування дизайнів та розробки додатків. При створенні інтерфейсу користувача за допомогою React ви перш за все дробите його на компоненти - невеликі блоки. Далі ви описуєте різноманітні візуальні стани для кожного компонента. Останній крок - з'єднання компонентів так, щоб дані безперешкодно протікали через них.

JSX, розширення синтаксису JavaScript, дозволяє вбудовувати розмітку, схожу на HTML, безпосередньо в JavaScript-файл. Незважаючи на інші варіанти написання компонентів, більшість розробників React віддають перевагу лаконічності JSX, використовуючи його в основній частині кодових баз.[4]

Для ефективної роботи з React важливо володіти глибокими знаннями у сфері JavaScript. Окрім цього, обов'язковими є розуміння HTML і CSS, оскільки вони необхідні для створення компонентів, їх візуального оформлення, стилізації та розташування елементів на сторінці, а також застосування візуальних ефектів. Успішна розробка передбачає ознайомлення з основними концепціями фреймворку,

такими як роутинг (навігація між сторінками) та життєвий цикл компонента (події, пов'язані з його створенням, оновленням та знищенням). Розуміння цих концепцій є важливим для розробки складних та масштабних проектів. Крім того, важливо постійно покращувати свої навички, вивчати нові інструменти та оновлення, щоб залишатись актуальним та ефективним розробником.

AngularJS

AngularJS представляє собою JavaScript-фреймворк, розроблений компанією Google, спрямований на створення односторінкових веб-додатків (Single Page Applications - SPA). Його основною метою є спрощення та удосконалення процесу розробки, де важливий акцент робиться на реалізації динамічних взаємодій з користувачем без перезавантаження сторінки.

Однією з ключових особливостей AngularJS є механізм двостороннього зв'язку даних (Two-way Data Binding), який автоматично синхронізує зміни в моделі та представленні. Ця риса спрощує взаємодію з даними та забезпечує їх автоматичне оновлення в реальному часі.

Також важливою концепцією є компонентна архітектура та модульність, що сприяє розділенню функціональності на невеликі, самодостатні компоненти. Це сприяє кращій структуризації додатку та його підтримці в подальшому.

AngularJS використовує механізм ін'єкції залежностей для ефективного керування компонентами та їх залежностями. Цей підхід полегшує тестування та підтримку кодової бази.

Директиви є іншим ключовим аспектом AngularJS, дозволяючи розширювати HTML-розмітку та створювати власні, що можна повторно використовувати компоненти.

Окрім цього, фреймворк пропонує систему фільтрів для обробки та форматування даних в шаблонах, а також життєвий цикл компонентів, який дозволяє визначити функції для різних етапів життєвого циклу, таких як створення, оновлення та знищення.

Vue.js

Vue.js є фреймворком для розробки інтерфейсів користувача з підтримкою створення односторінкових веб-додатків (SPA). Заснований на прогресивному підході, він відзначається своєю легкістю вивчення та зручною інтеграцією, що сприяє його популярності серед розробників. Розроблений Єваном Ю, Vue.js впроваджує концепції реактивності та двостороннього зв'язку даних, що забезпечує автоматичне оновлення інтерфейсу користувача при зміні даних.

Однією з ключових рис Vue.js є його компонентна архітектура, яка підтримує розділення додатку на невеликі, перевикористовувані компоненти. Це створює модульну та підтримувану структуру коду. Додатково, Vue.js використовує директиви для розширення можливостей HTML та шаблонів, що дозволяє інтегрувати логіку в розмітку.

Фреймворк працює на основі екземплярів, що спрощує взаємодію з окремими частинами додатку. Важливою особливістю є також система ін'єкції залежностей для ефективного управління компонентами та їх залежностями.

Vue.js відзначається розширеними можливостями для анімацій та переходів, надаючи інструменти для створення привабливих візуальних ефектів в інтерфейсі користувача.

2.1.4 Порівняльна характеристика розглянутих фреймворків

- Архітектура

React вирізняється за своєю декларативною архітектурою, що дозволяє визначити, як має виглядати інтерфейс у будь-який момент часу. Це спрощує відлагодження та підтримку коду. Angular, натомість, використовує маніпулятивний підхід, де модифікується DOM напряму. Vue, у свою чергу, пропонує гібридний підхід, комбінуючи декларативні та імперативні елементи.

- Віртуальний DOM

React використовує віртуальний DOM для оптимізації оновлення інтерфейсу. Це дозволяє зменшити кількість операцій маніпулювання реальним DOM, забезпечуючи більшу продуктивність. У порівнянні, Vue також використовує віртуальний DOM, тоді як Angular впроваджує стратегію "зон виявлення".

- Компонентна архітектура

У своїй основі React побудований на концепції компонентів, що сприяє повторному використанню та легшій розширюваності. Angular також використовує компонентний підхід, але має більше вбудованих функціональностей, що може створювати додатковий обсяг коду. Vue відомий своєю легкістю введення в розробку завдяки простоті компонентів та їхній гнучкості.

- Стан та управління станом

React пропонує підхід однозв'язкового потоку даних, що спрощує відстеження та розуміння того, як змінюється стан додатку. Angular надає механізм двохстороннього зв'язку, що може призводити до складних залежностей. Vue пропонує концепцію реактивності, що дозволяє автоматично відслідковувати та оновлювати зміни у стані.

- Екосистема та спільнота

React користується активною та великою спільнотою, що призводить до широкої підтримки, додаткових бібліотек та інструментів. Angular має повний набір інструментів від Google, а Vue, хоч і менш популярний, але добре підтримується.

2.2 Інструментарій бек-енд розробки

Бек-енд, зворотня, прихована частина програмного забезпечення, є складовою частиною програмного забезпечення, яка відповідає за обробку логіки бізнес-логіки, зберігання та керування даними. У контексті веб-додатків та сервісів, backend є серверною частиною, яка взаємодіє з клієнтською (frontend) стороною та базою даних. Backend виконує ряд ключових завдань, включаючи обробку запитів від користувачів, авторизацію та аутентифікацію, взаємодію з базою даних для

зберігання та витягування інформації, обчислення та логіку бізнес-процесів. Крім того, backend відповідає за забезпечення безпеки даних, відслідковування стану системи та оптимізацію продуктивності.

Створення бекенду для цього веб-додатка - ґрунтується, насамперед, на використанні передових інструментів та передових технологій, які не просто роблять процес розробки зручним, але й забезпечують високоефективний та легко масштабований сервіс.

Ядром серверної частини веб-додатка є високоефективна платформа Node.js. Це не просто технологія, що надає можливість виконувати JavaScript код на сервері. Використання движка V8 від Google Chrome дозволяє Node.js вирізнятися своєю продуктивністю. Його неблокуюча архітектура відкриває безліч можливостей для ефективного та паралельного обслуговування навіть великого обсягу запитів. Node.js став не просто інструментом, а справжнім каталізатором для швидкої та ефективної обробки запитів, що робить його невід'ємною частиною серверної інфраструктури.

Express.js - фреймворк для Node.js, додає необхідний рівень абстракції, щоб спростити створення веб-додатків. Його лаконічність у поєднанні з багатофункціональністю дозволяє швидко реалізувати потрібні функції.

Веб-додаток має потужний інструмент для взаємодії між клієнтом та сервером, а саме Axios. Цей інтерфейс легко керує всіма комунікаційними аспектами, забезпечуючи зручний інтерфейс для ефективної взаємодії. Завдяки Axios, взаємодія стає максимально простою та зрозумілою, дозволяючи отримувати та обробляти дані без зайвих зусиль.

В застосунку використовуються дві бази даних - MySQL та MongoDB. Використання MongoDB, окрім MySQL, обумовлено специфічним характером даних, що опрацьовуються. MongoDB вибирається для зберігання інформації, пов'язаної з виробництвом (продукція), оскільки цей підхід краще відповідає особливостям нереляційних даних. MongoDB дозволяє гнучко зберігати та опрацьовувати дані без фіксованої схеми, що особливо корисно для обробки

великого обсягу змінюючоїся та неструктурованої інформації. MySQL, як реляційна база даних, використовується для забезпечення структурованої та зв'язаної інформації, наприклад, списків користувачів. Це обрано через здатність до встановлення зв'язків між таблицями, що сприяє однорідності та цілісності даних.

Усі ці технології і інструменти, які включають в себе Node.js, Express, Axios, MongoDB разом з MySQL, та весь функціонал, вбудовані в Docker контейнери, створюючи таким чином не лише потужну функціональність, але й забезпечуючи безпроблемну процедуру запуску програму на сервері та ефективне управління інфраструктурою. У даному випадку використання Docker виправдане з кількох ключових причин. Перш за все, Docker є безкоштовним інструментом, що робить його доступним і економічно вигідним. З іншого боку, він вже давно здобув статус промислового стандарту, що свідчить про його надійність та активне використання в індустрії.

Однією з ключових переваг Docker є його велика поширеність та добре встановлення як індустрійного стандарту. Це робить його логічним вибором для використання в різноманітних проектах та завданнях. Крім того, Docker є доступним практично будь-де і може бути легко встановлений та використаний як на локальному комп'ютері, так і на віддаленому сервері.

Важливим аспектом є простота використання Docker для розгортання різноманітних середовищ і додатків. Він надає зручний та ефективний спосіб деплою та управління програмними компонентами. Його контейнеризаційний підхід дозволяє ізолювати додатки та їхні залежності, забезпечуючи упорядкованість середовища в будь-якому місці.

Отже, використання Docker у цьому контексті виявляється не лише економічно обгрунтованим, але й забезпечує високий рівень стандартизації, доступності та зручності управління програмними компонентами та їх розгортанням.

Загальною метою цієї технічної архітектури є створення не просто веб-додатка, але ефективного, легкого у використанні та масштабованого сервісу для кінцевого користувача.

JSON Web Token (JWT) використовується для створення токенів, які забезпечують перевірку ідентичності користувача. Під час аутентифікації сервер створює JWT токен, який надсилається клієнту. Цей токен зазвичай зберігається у вигляді куки та використовується при кожному наступному запиті.

Потік аутентифікації розпочинається з введення користувачем облікових даних. Express перевіряє ці дані, і якщо вони вірні, сервер створює JWT токен. Цей токен передається клієнту та включається у заголовок кожного майбутнього запиту. Express використовує middleware для перевірки та розшифрування JWT токену, дозволяючи доступ до захищених ресурсів при валідному токені.

Middleware в середовищі Node.js та Express - це функції, які отримують доступ до об'єкта request (запиту), response (відповіді) та next (функції, яку слід викликати для передачі управління наступному middleware в ланцюжку) під час обробки HTTP-запиту. Middleware може виконувати певні операції, модифікувати об'єкти запиту та відповіді, а також визначати, чи має продовжуватися ланцюжок middleware.

Куки використовуються для зберігання токенів або іншої інформації на стороні клієнта, і вони автоматично включаються при кожному запиті. Цей комплексний процес, використовуючи Node.js та Express, надає ефективний механізм для реалізації аутентифікації за допомогою JWT у додатку.

2.3 Інструментарій front-end розробки

Фронтенд, у контексті веб-розробки, визначає ту частину програмного забезпечення, яка безпосередньо відповідає за візуальний та інтерактивний аспекти веб-додатків чи веб-сайтів. Це область, яка концентрується на тому, як користувачі бачать та спілкуються із веб-інтерфейсом.

Узагальнено, фронтенд є тим "лицем" веб-додатка, яке користувачі бачать та взаємодіють із ним, визначаючи їхні враження та комфорт від використання онлайн-платформи.

Фронт-енд частина додатку, розроблена з використанням бібліотеки MUI React (Material-UI для React), відзначається сучасним та естетичним дизайном, а також високою рівнем функціональності. MUI React є розширенням React, що використовує дизайн-мову Material Design від Google, і надає розробникам готові компоненти та стилізацію для швидкої і стильної реалізації інтерфейсу.

Компоненти, які надає Material-UI для React, виступають основним елементом конструкції фронтенд частини додатку. Їх використання дозволяє ефективно та швидко створювати різноманітні елементи інтерфейсу, такі як кнопки, форми, таблиці, карточки, діалоги та інші. Це спрощує процес розробки та сприяє досягненню єдності в дизайні, що є важливим аспектом для впровадження стандартів та забезпечення однорідності користувацького інтерфейсу.

Модульність та композиційна природа Material-UI React дозволяє розробникам створювати власні компоненти та легко їх комбінувати. Це забезпечує високий рівень перевикористання коду та забезпечує легкість управління розширенням функціоналу, що робить процес розробки більш гнучким та піддається масштабуванню.

Темізація та стилізація відіграють ключову роль у дизайні додатку, і Material-UI React надає можливості налаштовування тем для досягнення відповідності дизайну корпоративному стилю або індивідуальним вимогам проекту. Гнучка система стилізації дозволяє реалізовувати індивідуальні зміни та адаптувати компоненти до специфічних потреб користувацького інтерфейсу.

Адаптивний дизайн, підтримуваний Material-UI React, визначається можливістю легкої реалізації адаптивного інтерфейсу, що гарантує оптимальний вигляд та функціональність додатку на різних пристроях та розмірах екранів.

Окрім цього, Material-UI React ідеально взаємодіє з іншими бібліотеками та інструментами, що входять до React-екосистеми, такими як React Router, Redux та інші. Це розширює можливості фронтенд розробки та сприяє гармонійній інтеграції з існуючим кодом, що робить весь процес більш зручним та продуктивним.

В контексті розробки веб-додатку використання технологій, таких як Vite, React та JavaScript, є невід'ємною частиною архітектурного стеку. Проект обирає підхід, що базується на застосуванні бібліотеки React для побудови користувацького інтерфейсу. Однак, враховуючи відмінності в трактуванні React різними браузерами, виникає необхідність використання інструменту, який забезпечує трансформацію React-коду у ванільний JavaScript. В цьому контексті використовується Vite як потужний транспілятор, що виконує цю конвертацію в межах веб-браузера. Такий підхід спрощує управління кодовою базою та забезпечує єдність відображення на різних платформах.

Також важливим елементом конфігурації є використання веб-сервера Nginx для обслуговування фронтенд-частини додатку. Nginx діє як надійний сервер, забезпечуючи оптимізацію розподілення контенту для кінцевого користувача. Його використання сприяє підвищенню продуктивності та забезпеченню стабільної роботи веб-додатку.

В такому конфігураційному стеку кожен компонент має своєрідну функціональність: React забезпечує розробку інтерфейсу, Vite використовується для трансляції коду, а Nginx відповідає за ефективне обслуговування запитів та розподіл контенту. Такий інтегрований підхід робить веб-додаток стійким до змін та оптимізованим для максимальної продуктивності.

2.4 Висновки по другому розділу

Після ретельного вивчення різних мов програмування, включаючи JavaScript, TypeScript та Dart, прийдено до висновку, що JavaScript залишається оптимальним вибором для більшості фронтенд-проєктів. JavaScript виділяється своєю динамічною

природою, широкою підтримкою браузерів, що забезпечує швидку розробку та гнучкість. Незважаючи на наявність конкуруючих мов, таких як TypeScript та Dart, які мають свої переваги, вони можуть виявитися більш складними у вивченні та інтеграції. Проаналізовано технічні аспекти та архітектурні рішення, зосереджуючись на React. Висвітливши його чистий та декларативний підхід, віртуальний DOM та ефективне управління станом, було підтверджено, що React є вибором, що виправдовує свою популярність, ідеально підходячим для розробки сучасних веб-додатків.

Вибір мови програмування та архітектурних рішень для розробки інформаційної технології конфігурування компонентів автоматизованої системи виробництва є критичним для успішності проєкту. Обрана мова програмування, зокрема JavaScript, враховуючи її динамічну природу та широку підтримку, забезпечує швидкість розробки та гнучкість, необхідні для ефективної імплементації конфігурування компонентів.

Після аналізу технічних аспектів та архітектурних рішень можна визначити, що React виправдовує свою популярність завдяки чистому та декларативному підходу, віртуальному DOM і ефективному управлінню станом. Його компонентна архітектура та широка спільнота роблять його ідеальним вибором для розробки сучасного веб-додатку. Архітектурний фреймворк, такий як React, з його чистим підходом, ефективним управлінням станом та компонентною архітектурою, є ідеальним вибором для розробки інформаційної технології, орієнтованої на конфігурування компонентів автоматизованої системи виробництва. У виробничому середовищі, де швидкість розробки та ефективність грають важливу роль, обрані інструменти надають необхідний фундамент для успішної реалізації інноваційної та високотехнологічної інформаційної технології конфігурування виробничих компонентів.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

На рисунку 3.1 представлена структура інформаційної системи.

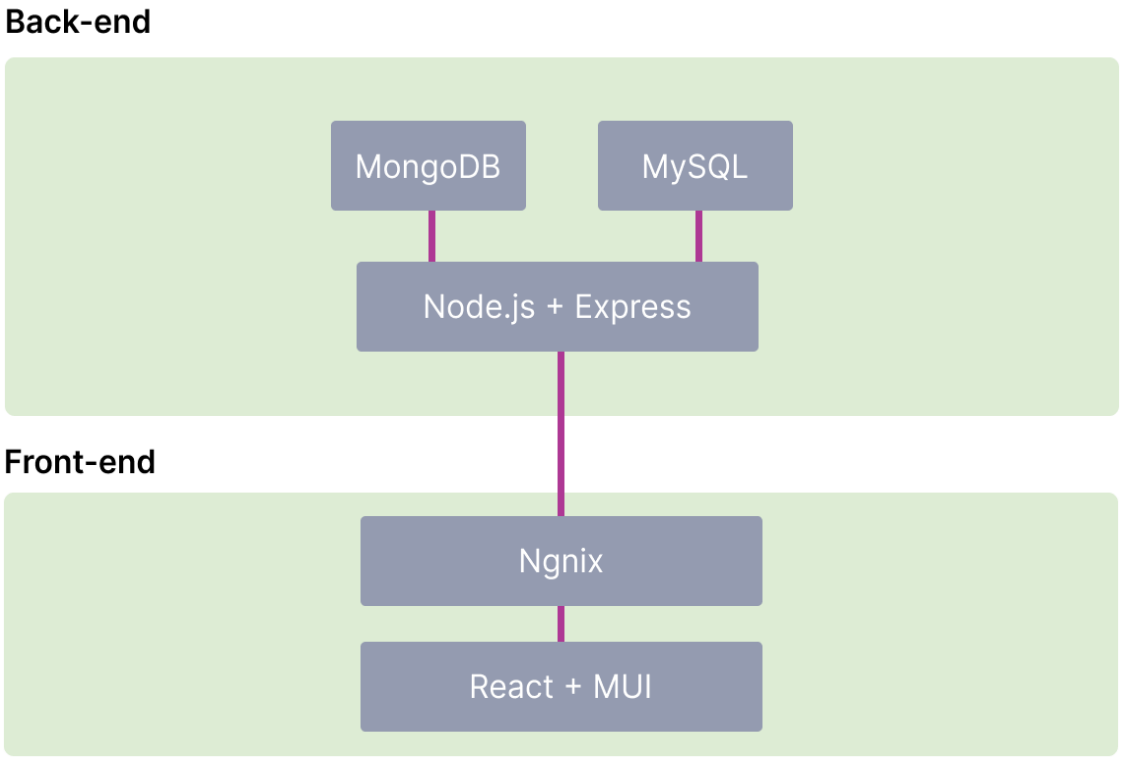

Рисунок 3.1 – Схематичний опис складових застосунку

3.1 Структура сторінок застосунку

Кожна сторінка застосунку має пов’язаний css файл, за допомогою якого створено розмітку сторінки. Приклад такої розмітки для сторінки Wire наведено на рисунку 3.2.

Задача цього CSS-коду - налаштувати вигляд сітки (гріду) для елементів на веб-сторінці. Рядок grid-template-columns вказує, яким буде розподіл ширини колонок. У цьому випадку перша колонка буде займати одну частину доступного простору, а друга колонка - три частини.

```css
#wire-page {
  height: auto;
  display: grid;
  grid-template-columns: 1fr 3fr;
  grid-template-rows: auto;
  grid-template-areas:
    "inputs view"
    "inputs summary";

  gap: 8px;

  justify-items: center;
  align-items: center;
}

#wire-inputs {
  grid-area: inputs;
  width: 100%;
  height: 100%;
}

#wire-view {
  grid-area: view;
  width: 100%;
}

#wire-summary {
  grid-area: summary;
  width: 100%;
  height: 100%;
}
```

Рисунок 3.2 – Пов’язаний .css файл для сторінки Wire

Рядок grid-template-rows вказує, яким буде розподіл висоти рядків. Тут використовується "auto", щоб висота рядків автоматично підганялася під зміст всередині них.

Рядок grid-template-areas може визначити області на гріді та вказати, які елементи будуть в яких частинах гріда.

Загалом, це правила для того, як грід буде розглядати і розміщувати свої елементи в різних частинах сторінки, забезпечуючи при цьому визначені ширини та автоматичну висоту. Для окремих областей визначені такі правила:

- #wire-inputs (inputs):
    1. Розміщений в лівій частині сторінки.

2. Займає всю доступну висоту та ширину своєї області.

- #wire-view (view):
  1. Розміщений в правій верхній частині сторінки.
  2. Займає всю ширину та висоту своєї області.
- #wire-summary (summary):
  1. Розміщений в правій нижній частині сторінки.
  2. Займає всю доступну висоту та ширину своєї області.

Ці елементи визначають, як сторінка буде розташована та виглядати на різних пристроях. Розділення сторінки на секції зображено на рисунку 3.3, зеленою обводкою позначено секцію inputs, помаранчевою – секцію view, рожевою – секцію summary.

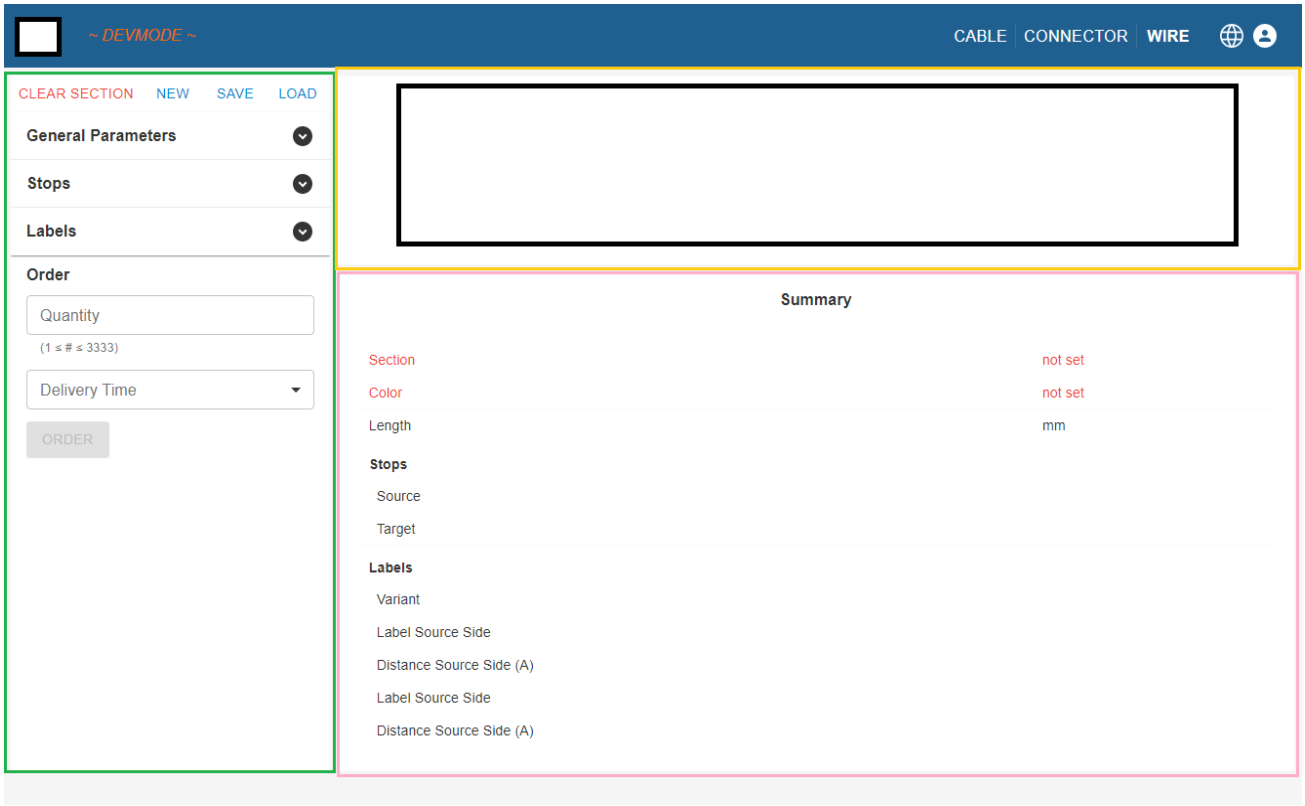

Рисунок 3.3 – Секції сторінки Wire

3.2 Компоненти

3.2.1 Акордеон

Компонент акордеон - це інтерфейсний елемент, який дозволяє користувачеві розгортати та згортати блоки контенту на сторінці. В згорнутому стані акордеон може показувати лише заголовок або короткий огляд змісту блоку, а коли користувач клікає на заголовок, відбувається розгортання для відображення повного вмісту блоку, це можна побачити на рисунку 3.4.

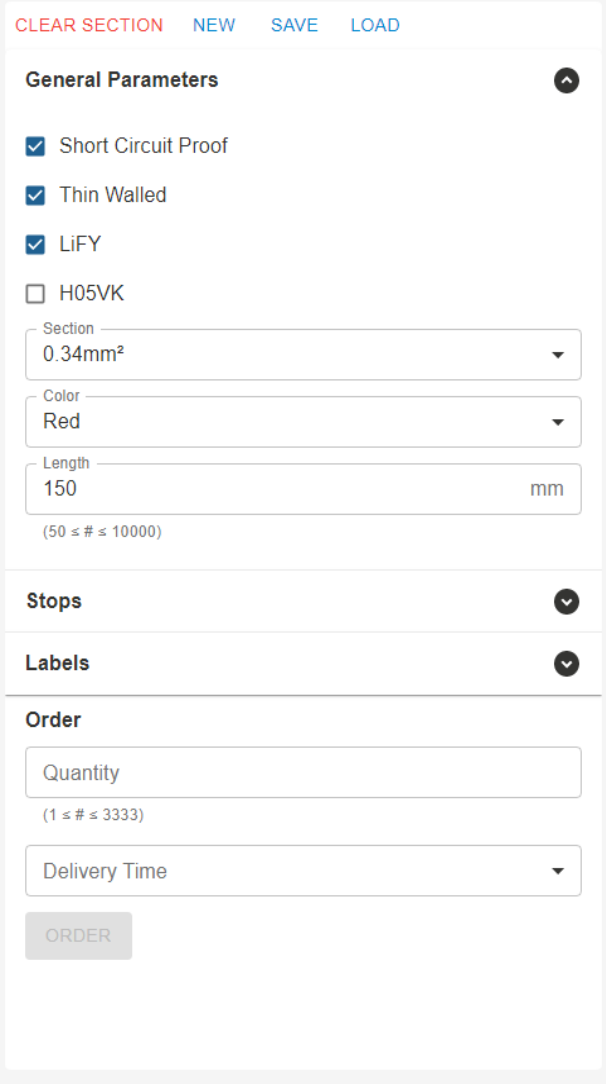

Рисунок 3.4 – Зовнішній вигляд компоненту акордеон в інтерфейсі

Основні елементи акордеону включають заголовки (у вигляді кнопок чи інших елементів керування) та вміст, який розгортається або згортається. Коли один блок розгорнутий, інші можуть автоматично згортатися (залежно від налаштувань), далі буде наведено приклад програмної реалізації такої поведінки блоків акордеону. Акордеони часто використовуються для організації великої кількості інформації або опцій, забезпечуючи користувачеві можливість зосередитися тільки на необхідних елементах. Вони зручні та ефективні для структуризації великих обсягів даних.

Компонент CustomAccordion (рис.3.5) побудований для створення кастомізованого компоненту, що по своїй суті є обгорткою для внутрішніх елементів, так як MUI компонент Accordion являє собою, окремий блок акордеону. Ця обгортка необхідна для загального контролю над всіма блоками, що будуть вкладені всередині, для забезпечення зручного організованого управління та координації функціоналу різних частин акордеону.Обгортка створює відокремлений компонент, який можна легко використовувати в інших частинах додатку. Це сприяє перевикористанню коду та спрощенню конфігурації.

Рисунок 3.5 - Код компоненту CustomAccordion

У компоненті використовується стан expandedUUID, який відслідковує ідентифікатор (UUID) розгорнутої секції акордеону. Він використовується для визначення того, яка саме секція має бути розгорнута в даний момент. Кожна секція

акордеону має унікальний ідентифікатор, представлений як UUID (унікальний ідентифікатор унікальний для кожної секції). Він може бути чимось унікальним для ідентифікації конкретної секції, наприклад, номером чи іншим унікальним значенням.

У розробленому рішенні, використовується JavaScript бібліотека uuid v4, що допомогає легко генерувати унікальні ідентифікатори для окремих компонентів за допомогою вбудованих функцій. Стан expandedUUID встановлений в false при створенні компонента. Це означає, що на початку жодна з секцій акордеону не розгорнута. Коли користувач розгортає конкретну секцію, викликається функція toggleExpansion, яка оновлює стан expandedUUID і присвоює йому ідентифікатор розгорнутої секції. Якщо користувач згортає секцію, expandedUUID встановлюється в false.

Цей механізм дозволяє контролювати, яка саме секція повинна бути розгорнута на даний момент. Коли передається expandedUUID як властивість до кожної секції акордеону, ця секція може визначати свій власний стан (розгорнуто чи згорнуто) на основі порівняння свого ідентифікатора з поточним expandedUUID.

Цей акордеон має кілька розділів (секцій), які можуть бути розгорнуті або згортані. Кожен розділ є окремим компонентом, який передається в якості "дітей" до акордеону. З метою додавання певної функціональності кожній секції акордеону було вирішено використовувати функцію React.Children.map, яка дозволяє пройтися по кожній дочірній секції акордеону. Для кожної секції викликається React.cloneElement, щоб створити новий екземпляр кожного компонента та додати до нього додаткові властивості (пропси), такі як expandedUUID, singleExpand і toggleExpansion. У результаті цей процес перетворює кожну дочірню секцію акордеону, надаючи їй можливість взаємодії зі станом акордеону через передані пропси. Отриманий масив, що складається з модифікованих версій кожної дочірньої секції, готовий до відображення та використання в компоненті акордеону.

Компонент повертає результат відображення всіх дочірніх елементів, які були оброблені та змінені за допомогою React.cloneElement.

3.2.2 Секція акордеону

Чайлд елементом є React-компонентом з назвою Collapsible. Цей компонент використовується для створення розгортного блоку акордеону. Компонент має ряд властивостей, таких як заголовок (caption), ідентифікатор розгорнутого блоку (expandedUUID), функція для розгортання/згортання блоку (toggleExpansion), флаг для розгортання лише одного блоку одночасно (singleExpand), вміст блоку (children), та флаг вимкнення взаємодії з блоком (disabled). Саме в цьому компоненті генерується унікальний ідентифікатор, за яким у компоненті-обгортці було визначено поведінку розгортних блоків. (рис. 3.6)

У залежності від значення singleExpand, компонент рендерить акордеон з можливістю розгортання одного блоку чи акордеон, поведінка якого буде стандартною – відкриття всіх вкладок, без автоматичного закривання попередніх.

Акордеон включає заголовок з іконкою розгортання та деталі, які розгортаються при кліку. Заголовок і вміст локалізовані за допомогою бібліотеки react-i18next. Крім того, можна вимкнути взаємодію з акордеоном, якщо властивість disabled встановлено в true (за замовчуванням).

Цей компонент дозволяє створювати динамічні та гнучкі розгортні блоки з можливістю налаштування, що сприяє розширенню та перевикористанню в різних частинах веб-додатка.

3.2.3 Переклад додатку

react-i18next - це потужна бібліотека для інтерналізації у React-додатках, тобто – перекладу. Замість того, щоб надавати переклад для кожного текстового рядка в коді, інтерфейс визначається так, ніби він вже перекладений, використовуючи ключі перекладу. Використання react-i18next виявляється надзвичайно зручним завдяки його декларативному підходу та гнучкості. Завдяки ньому, можна передавати контекст для точних перекладів, а його інтеграція з React здійснюється легко та

ефективно. Ця бібліотека надає високорівневі компоненти, такі як Trans, які дозволяють декларативно використовувати переклади прямо в JSX. Це робить код більш читабельним та зберігає функціонал інтернаціоналізації поруч із компонентом.[5]

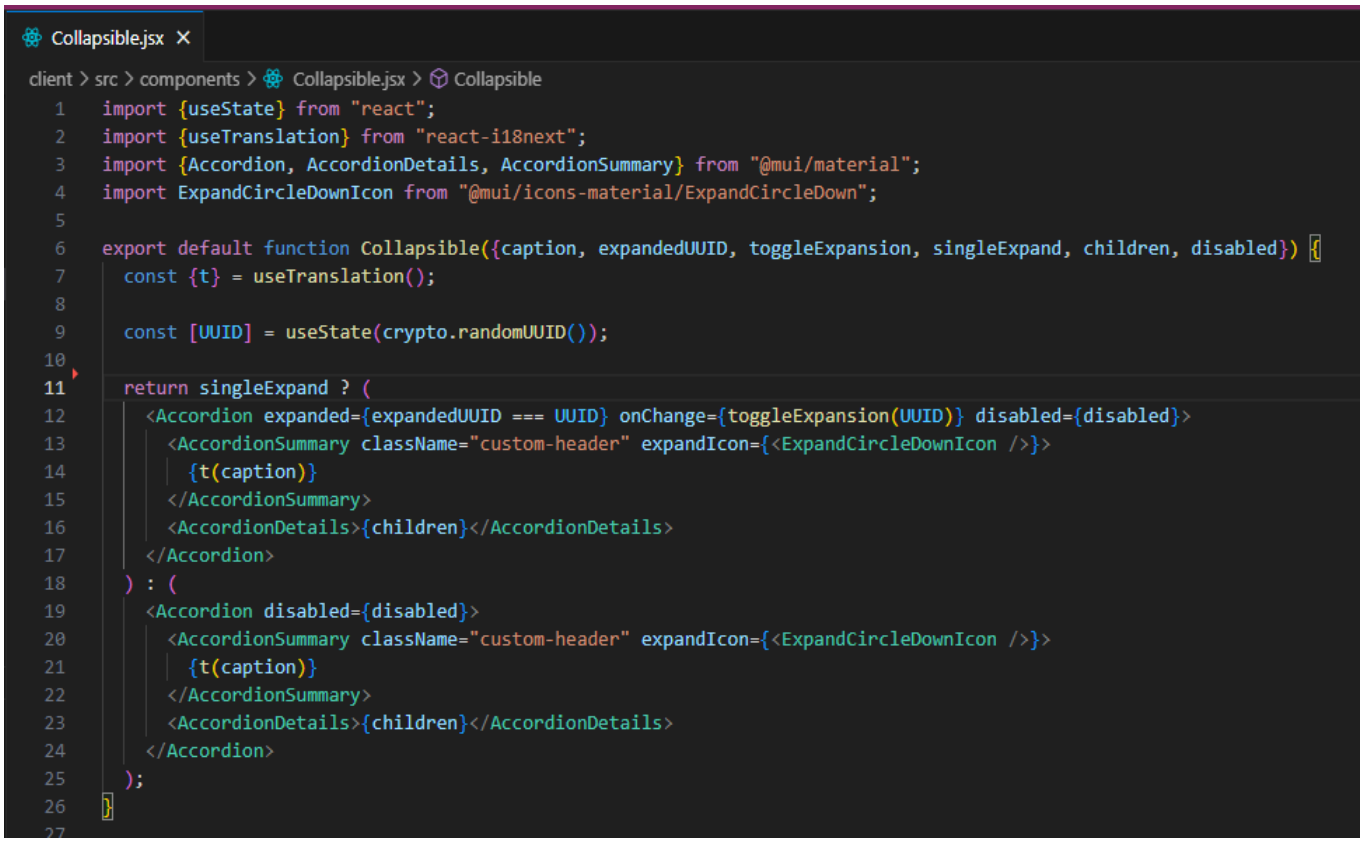

Рисунок 3.6 – Код компоненту секції акордеону

Додатково, react-i18next підтримує динамічне завантаження ресурсів перекладу, що особливо корисно для великих проектів. Також вона підтримує різні формати перекладу та може бути розширена за допомогою плагінів для додаткової функціональності, такої як форматування чисел, валют тощо. У цілому, react-i18next - це інструмент, який робить локалізацію React-додатка простою та ефективною, дозволяючи легко взаємодіяти з різними мовами та забезпечити приємний досвід для користувачів з усього світу.

Здійснено інтернаціоналізацію проекту на англійську та німецьку мови. Для кожної мови, англійської та німецької, були створені відповідні файли перекладу, en.json, фрагмент якого представлений на рисунку 3.7, та de.json зображений на рисунку 3.8 відповідно. У цих файлах був визначений переклад для ключа "color": "Color" для англійської та "Farbe" для німецької.

Після цього в React-коді компонентів був використаний хук useTranslation для отримання функції перекладу t. При виведенні текстового елементу на екран було використано {t('color')}, що автоматично повертало відповідний переклад в залежності від обраної мови. Наприклад, при виборі англійської мови, повертався "Color", а при виборі німецької - "Farbe".

Цей підхід дозволяє автоматизувати процес локалізації текстів, зробити його зручним та легким для розширення. За допомогою цієї імплементації, не потрібно безпосередньо втручатися у визначення обраної мови чи вибору відповідного перекладу, оскільки це вже автоматизовано за допомогою react-i18next.

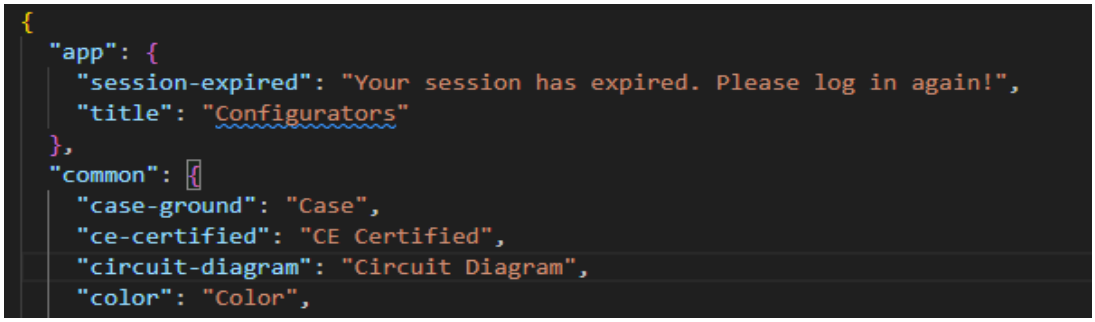

Рисунок 3.7 – Фрагмент коду перекладу застосунку на англійську мову

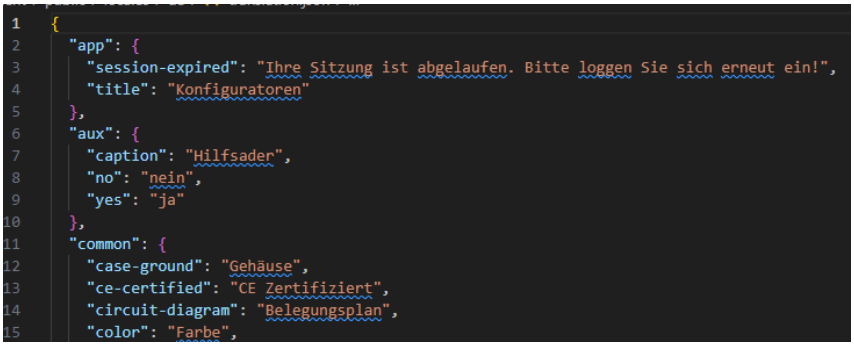

Рисунок 3.8 - Фрагмент коду перекладу застосунку на німецьку мову

Локалізація застосовується практично до кожного окремого кастомного елементу, так як окрім основи елементів ще маються такі допоміжні елементи як підказки (Tooltip) та інше. (рис. 3.9)

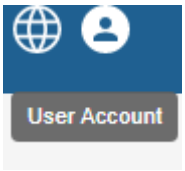

Рисунок 3.9 – Переклад компоненту англійською мовою

Зміна мови у цьому веб-додатку реалізована написанням окремого компоненту для цієї задачі, щоб забезпечити «мобільність» елементу, у випадку змін в інтерфейсі. Було використано іконку Глобуса з бібліотеки іконок MUI, додано до неї підказку, яка також має свій переклад. Реалізація випадаючого списку була реалізована за допомогою MUI компоненту <Menu>, що містить в собі два дитячих елементи <MenuItem>. Також, всередині цього компоненту була реалізована функція зміни мови, що пізніше використовується у методі onClick елементу <MenuItem>. (рис. 3.10).

```
const handleClose = (e) => {
  i18n.changeLanguage(e.target.innerText.toLowerCase());
  setAnchorEl(null);
};
```

Рисунок 3.10 – Функція зміни мови

В графічному інтерфейсі користувача зміну інтерфейсу можна побачити у хедері, що змінює лише частини своїх станів в залежності від поточної сторінки, статусу реєстрації чи входу в додаток тощо, тобто користувач має змогу в будь який момент змінити мову додатку не витрачаючи на це час на пошуки, та не маючи загрози втратити зміни у власних конфігураціях. (рис. 3.11).

Рисунок 3.11 – Зміна мови в графічному інтерфейсі користувача

3.2.4 Випадаючий список

Ще одним важливим елементом інтерфейсу, одним з основних для застосунку, являється елемент CustomSelect (рис. 3.12). Він розроблений для створення випадаючих списків, в яких користувач обирає потрібний йому пункт. Цей елемент є складеним, що означає, що кожну його складову можна кастомізувати під певні потреби.

```
<FormControl disabled={disabled}>
  <InputLabel>{t(caption)}</InputLabel>
  <Select value={value ? value.id : ""} label={t(caption)} onChange={(e) => onChangeHandler(data.find((element) => element.id === e.target.value))}>
    {data.map((element, index) => {
      return (
        <MenuItem key={index} value={element.id}>
          {t(element.caption)}
        </MenuItem>
      );
    })}
  </Select>
</FormControl>
```

Рисунок 3.12 – Фрагмент коду компоненту випадаючого списку

Так, наприклад, тег <FormControl> кастомізовано властивістю disabled, що вимикає можливість вибору варіанту в даному списку. При використанні випадаючого списку в конкретних місцях сайту, ця властивість змінюється динамічно в залежності від конкретних задач. Так, наприклад, на рисунку 3.13 зображена поведінка вимкнутого елементу вибору, так підказка для користувача, про необхідність спочатку обрати дані з випадаючого списку “Section”.

Рисунок 3.13 – Стани компоненту випадаючий список

FormControl використовується для управління станом та відображенням форм в React, а також є обгорткою для внутрішніх тегів, тому кожна його властивість застосовується для всього елементу. Окрім властивості disabled, він також може приймати такі властивості як color – зміна кольору компонента, fullWidth – що означає розтягування випадаючого списку на всю ширину батьківського елементу. [6]

3.2.5 Стани компонентів на прикладі компоненту поля для вводу

Однією з основних функцій React додатку є функція useEffect. useEffect - це React-хук, який дозволяє виконувати певні дії в компоненті після того, як відбулися зміни в його життєвому циклі. Основна ідея використання useEffect полягає в тому, щоб реагувати на певні події, які можуть відбутися в компоненті під час його монтажу, оновлення або демонтажу. Хук дозволяє реагувати на зміни в стані компонента або його властивостях.

На прикладі елементу CustomTextField, код якого приведено на рисунку 3.14, що використовується для створення кастомізованих полів вводу клієнтських даних, використано цей хук.

Перший useEffect реагує на зміни у значенні value і оновлює shadowValue. Другий useEffect в свою чергу, реагує на зміни в shadowValue. Він робить дві речі:

- setError(!inputRegex.test(shadowValue)): Встановлює помилку на основі перевірки shadowValue на відповідність заданому регулярному виразу (inputRegex). Якщо shadowValue не відповідає, помилка встановлюється як true, інакше - false.

- !error && onChangeHandler(shadowValue): Якщо помилок немає, викликає функцію onChangeHandler із поточним значенням shadowValue.

-

```
useEffect(() => setShadowValue(value), [value]);

// mimics a mysql string, but: excludes leading or
const inputRegex = new RegExp("^(?!\\s)[^\\x00-\\x1

useEffect(() => {
  setError(!inputRegex.test(shadowValue));
  !error && onChangeHandler(shadowValue);
}, [shadowValue]);
```

Рисунок 3.14 – використання хуку useEffect в коді елементу CustomTextField

Однією з властивостей вкладеного тегу цього компоненту являється властивість multiline, що дозволяє вводити в поле більше ніж одну строку тексту. Приклад елементу з цією властивістю зі значенням true, наведено на рисунку 3.15 . Варто зазначити, що даний вигляд елементу не є виглядом за замовчуванням, і був окремо налаштований при розробці. В пустому вигляді елемент має фіксовані розміри і смуга прокручування, тобто скрол бар, прихований. Кількість строк, які можуть бути введені, встановлено за допомогою іншої властивості цього елементу – rowCount. Її встановлення не є обов’язковим, так як за замовчуванням воно встановлено на 1, і при зміні кількості строк без призначення елементу властивості multiline вигляд елементу не зміниться.

Рисунок 3.15 – Компонент з властивістю multiline

3.2.6 Компонент поля вводу чисельних значень

Компонент CustomNumericField створений для чисельних значень, що перевіряється у коді за допомогою використання useEffect (рис. 3.16) на основі перевірки чи відповідає введене значення типам Integer та Float, та чи заповнене поле тощо.

Так як цей функціонал не передбачений бібліотекою, його було створено власноруч при розробці, з використанням кастомних функцій, за допомогою яких введене значення проходить перевірку на тип даних. Результат, при якому введене число не відповідає вимогам поля, зображений на рисунку 3.17 .

```
useEffect(() => {
  // reset on empty field
  if (isEmpty(shadowValue)) {
    setError(false);
    setErrorMessage("");
    onChangeHandler(shadowValue);
    return;
  }

  if (containsWhiteSpace(shadowValue)) {
    setError(true);
    setErrorMessage(t("error.invalidInput"));
    return;
  }

  if (type === "INTEGER" && !isInteger(shadowValue)) {
    setError(true);
    setErrorMessage(t("error.invalidInput"));
    return;
  }

  if (type === "FLOAT" && !isFloat(shadowValue)) {
    setError(true);
    setErrorMessage(t("error.invalidInput"));
    return;
  }

  if (!isInRange(shadowValue)) {
    setError(true);
    setErrorMessage(rangeInfo());
    return;
  }

  setError(false);
  setErrorMessage("");

  if (type === "INTEGER") return onChangeHandler(parseInt(shadowValue));
  if (type === "FLOAT") return onChangeHandler(parseFloat(shadowValue));
  return onChangeHandler(shadowValue);
}, [shadowValue]);
```

Рисунок 3.16 – Фрагмент коду компоненту CustomNumericField

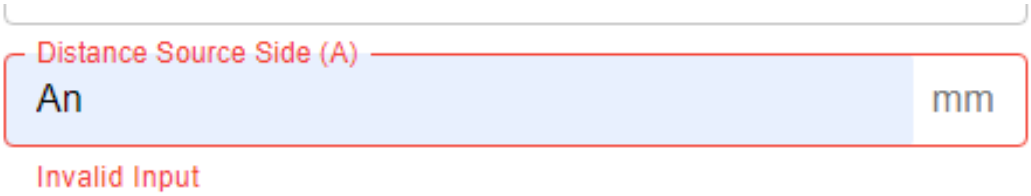

Рисунок 3.17 – Реагування компоненту на дані, що не відповідають вимогам

3.3 Зовнішній вигляд застосунку

Компоненти MUI дотримуються концепцій Material Design, які визначають стандарти вигляду та поведінки елементів інтерфейсу. Це створює єдиний і зрозумілий дизайн, що полегшує користувачам розуміння та взаємодію з інтерфейсом. Приклад поєднання елементів зображений на рисунку 3.18, на ньому видно, що компоненти легко відрізняються між собою, однак в той самий час складається відчуття цілісності та довершеності інтерфейсу.

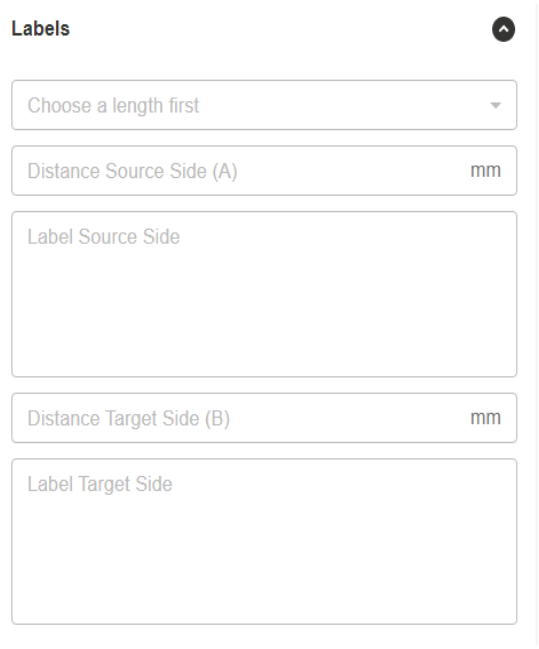

Рисунок 3.18 – Поєднання різних компонентів в інтерфейсі

Так як застосунок розроблений для конкретної задачі на виробництві, він буде розташований на вже існуючому сайті компанії, у якої вже існує свій корпоративний

стиль, важливою задачею було корретно інтегрувати та налаштувати візуальну частину нового додатку, враховуючи цей корпоративний стиль.

Корпоративний стиль - це система графічних елементів, що визначає зовнішній вигляд та візуальну ідентичність компанії. В рамках цього стилю визначаються основні компоненти, такі як корпоративні кольори, шрифти, логотипи, графічні елементи та інші важливі аспекти. Корпоративний стиль забезпечує єдність та консистентність у всіх візуальних матеріалах компанії, сприяючи впізнаваності та встановленню позитивного іміджу.

Його елементи включають в себе офіційні кольори, шрифти для використання у тексті та заголовках, а також інші стилізовані елементи, які відображають атмосферу та цінності компанії. В даному випадку, було прийнято рішення відмовитись від використання корпоративних шрифтів, для уникнення можливих проблем серед користувачів щодо досвіду використання додатку, так як різні браузери та операційні системи можуть різнитися у підтримці шрифтів, що може призвести до неоднакового відображення на різних платформах.

Однак, так як у компанії існує "Brand Guidelines", або ж керівництво по стилю, що являє собою документ, який містить усі важливі елементи дизайну для компанії, такі як логотипи, кольори, шрифти, типографіка, стилізація графіки, принципи дизайну тощо, згідно нього було створено налаштування візуального вигляду застосунку за допомогою MUI Theming.

3.3.1 Створення тем застосунку

MUI Theming - це система, яка дозволяє змінювати зовнішній вигляд та стилі компонентів, роблячи їх більш придатними для індивідуального бренду або дизайну проекту. Основна ідея полягає в тому, щоб мати централізований об'єкт теми, який містить параметри стилізації, такі як кольори, шрифти та розміри. Material-UI використовує ThemeProvider, який дозволяє визначити глобальні налаштування теми для всього застосунку. Це дозволяє змінювати стилі визначених компонентів, дозволяючи їм відповідати вимогам дизайну і не налаштовувати кожен окремий

компонент. По суті, MUI Theming надає гнучкість у зміні вигляду компонентів, та дозволяє не залежати від стандартного зовнішнього вигляду Material-UI.

За допомогою цього потужного інструменту можна налаштовувати окремі властивості елементів, приклад такого налаштування наведений на рисунку 3.19. Всі налаштування, що стосуються зовнішнього вигляду застосунку, містяться в окремому файлі CustomTheme.jsx. Він зберігає в собі налаштування, та його ініціалізація відбувається на моменті запуску додатку, тобто після виклику головного файлу додатку (index.jsx). Даний файл, згідно за документацією MUI, повертає вкладені теги, тобто сам застосунок, що обгорнуті у раніше згаданий ThemeProvider.

```
MuiDialogContent: {
  styleOverrides: {
    root: {
      "& .MuiTableCell-root": {
        padding: "4px 8px 4px 8px",
      },
    },
  },
},
```

Рисунок 3.19 – Налаштування зовнішнього виглядукомпоненту всередині теми

Для елементу MuiTableCell, що розташована всередині компоненту MuiDialogContent, застосована зміна відстані між внутрішнім контентом елемента та його зовнішнім контуром (границею), тобто padding. Варто зазначити, що синтаксис теми побудований таким чином, такі налаштування будуть працювати тільки для елементів, що є дочірніми для MuiDialogContent. Тобто MuiTableCell поза цим батьківським елементом зберігає свій стандартний вигляд.

Кольори застосунку налаштовані у цьому ж файлі теми. Створено дві палітри, для світлої та темної теми застосунку (рис. 3.20).

```
 7 > const lightPalette = {···
44   };
45
46   const darkPalette = {
47     palette: {
48       mode: "dark",
49       primary: {
50         main: "#1F6091",
51         dark: "#575756",
52         contrastText: "#FFFFFF",
53       },
54       secondary: {
55         light: "#FFFFFF",
56         main: "#D9D9D9",
57         dark: "#999999",
58         contrastText: "#000000",
59       },
60       background: {
61         paper: "#212121",
62         default: "#121212",
63       },
64       text: {
65         primary: "#FFFFFF",
66         secondary: "#BDBDBD",
67         disabled: "#616161",
68       },
69       action: {
70         active: "#FFFFFF",
```

Рисунок 3.20 – Палітра для світлої теми додатку

Вибір між палітрами відбувається за допомогою визначення теми, що використовується користувачем за замовчуванням, згідно налаштувань його операційної системи чи браузеру (рис. 3.21).

Хук useMemo дозволяє оптимізувати продуктивність компонентів шляхом кешування результатів виразу та повернення його з кешу, якщо вхідні значення не змінилися. В даному випадку використовується активна палітра та до елементу теми передається аргументом ця активна палітра. Це відбувається тому що налаштування деяких елементів має в собі залежність від кольору, що змінюється в залежності від активної теми.

```
const activeTheme = useMemo(
  () =>
    createTheme({
      ...activePalette,
      ...Theme(activePalette),
    }),
  [activePalette],
);
```

Рисунок 3.21 – Встановлення активної теми додатку

Таким чином, в залежності від налаштувань теми та, як зазначено вище, від налаштувань системи користувача застосунок буде мати вигляд як на рисунку 3.22, або ж як на рисунку 3.23, у випадку використання темної теми.

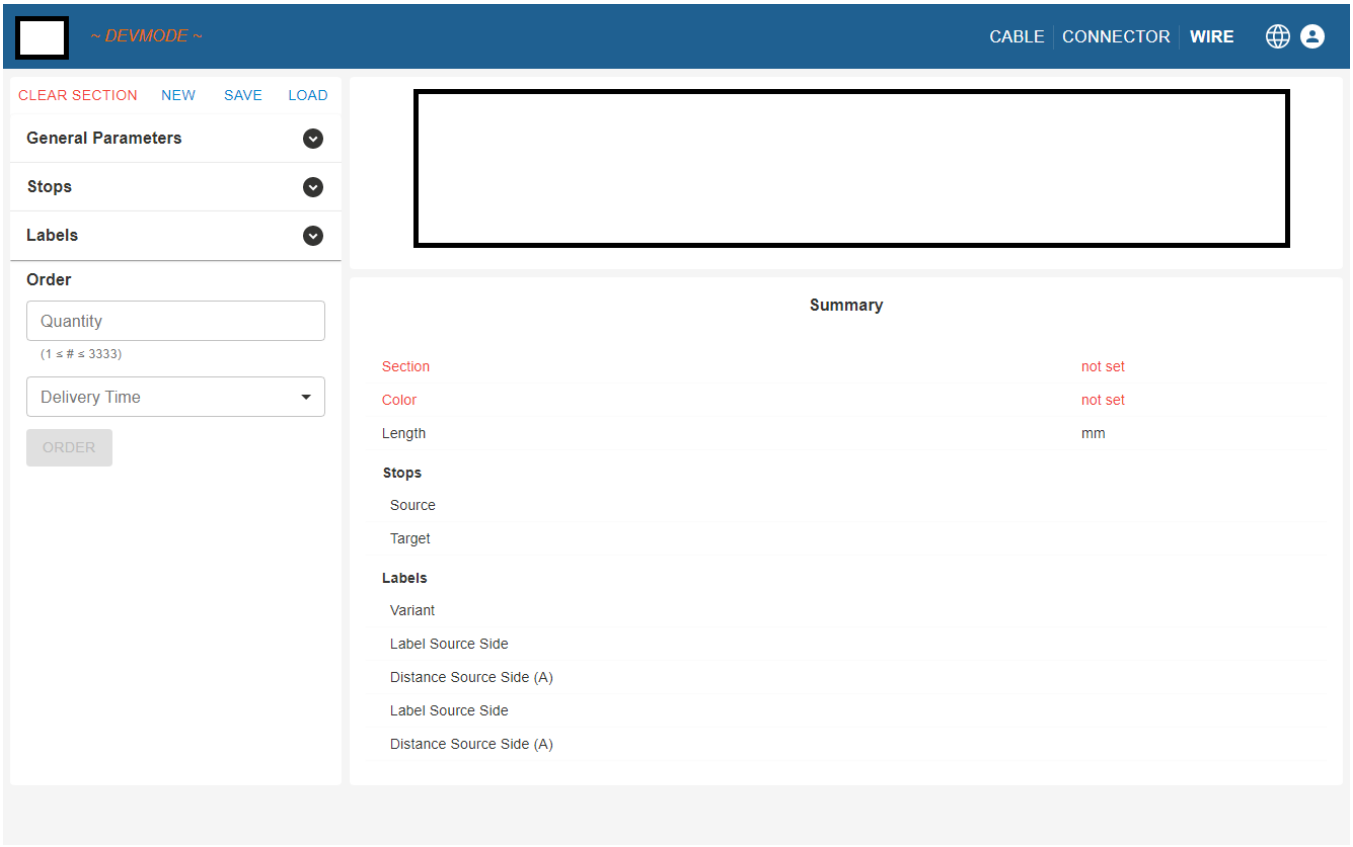

Рисунок 3.22 – Світла тема додатку

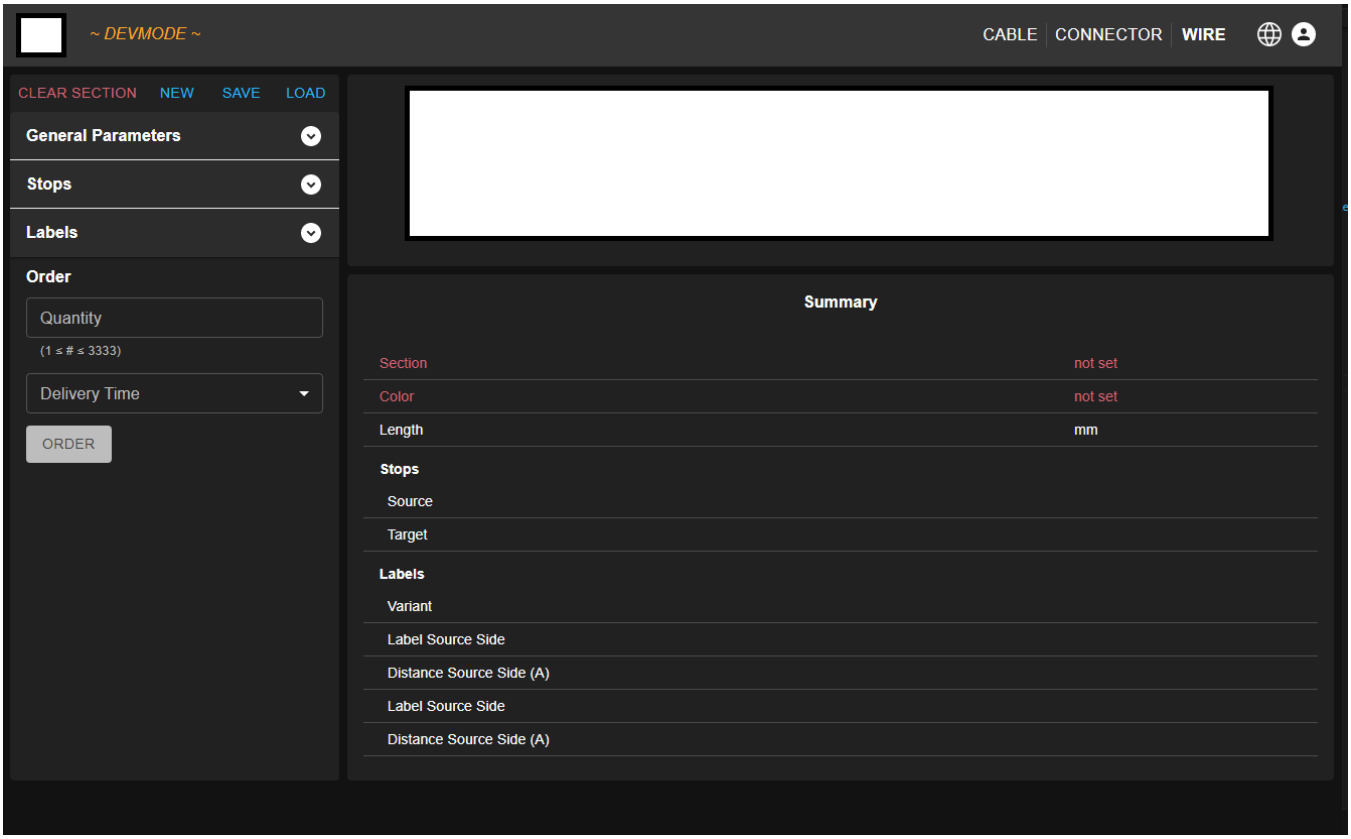

Рисунок 3.23 – Темна тема додатку

3.4 Висновки по третьому розділу

Розроблено значущу частину застосунку за допомогою React + MUI, а саме лицьову (фронт-енд) частину, через яку користувач взаємодіє з застосунком.

Використано компонентний підхід, що при необхідності дозволяє масштабувати та змінювати застосунок без великої витрати часу та зусиль. Компоненти поєднуються і взаємодіють між собою як єдина система, що полегшує не тільки розробку, дороблення чи правлення застосунку, від потреб проекту, що можуть змінюватись, а і сприйняття користувачем інтерфейсу застосунку.

Було створено компоненти під різні потреби і типи даних; визначено поведінку компонентів за різних умов; вирішено проблему локалізації за допомогою додаткової бібліотеки.

Також, запроваджено єдиний стиль застосунку, що відповідає корпортивному стилю компанії, якій належить виробництво.

ВИСНОВКИ

У рамках даної кваліфікаційної роботи було успішно реалізовано інформаційну технологію конфігурування компонентів автоматизованої системи виробництва для підприємства, спеціалізуючогося на виробництві кабелів, дротів та коннекторів.

Фронтенд конфігуратора було впроваджено з використанням сучасних технологій, таких як JavaScript та React, що забезпечили високий рівень інтерактивності та ефективність веб-інтерфейсу. Завдяки цьому, користувачам надано зручний та інтуїтивно зрозумілий інструмент для конфігурації продуктів.

Бекенд системи, розроблений за межами основного обсягу дипломної роботи, використовував технології MongoDB для зберігання даних та Node.js разом із фреймворком Express.js для забезпечення оптимальної обробки запитів та взаємодії з базою даних. Ця інтеграція дозволила створити стійку та швидкодіючу серверну частину, необхідну для ефективної роботи конфігуратора.

Важливими особливостями проекту є гнучка структура даних, що дозволяє легко додавати нові продукти та опції, а також уважний підхід до візуалізації обраних опцій та конфігурацій. Можливість збереження та повторного виклику конфігурацій робить використання конфігуратора практичним та зручним для кінцевих користувачів.

У результаті комплексного підходу до вирішення завдання було досягнуто високих стандартів ефективності та гнучкості конфігурування компонентів. Розроблений конфігуратор є значущим кроком у впровадженні сучасних інформаційних технологій у виробничий процес та сприяє подальшому розвитку автоматизованого виробництва.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. McKinsey Institute A future that works automation, employment and productivity, 2017, с. 4
2. Oxford Economics How robots change the world, 2019, с. 5-24
3. Karolina Mahrla Porsche expands online sales to include customer-configured cars, 2022, URL: https://newsroom.porsche.com/en/2022/products/porsche-new-car-configurator-29875.html
4. React Documentation Thinking in React URL: https://react.dev/learn/thinking-in-react
5. Reacti18next What is react-i18next?, 2023, URL: https://react.i18next.com/
6. MUI Docs FormControl APIhttps://mui.com/material-ui/api/form-control/

ДОДАТОК А

Відомість матеріалів кваліфікаційної роботи

№ з/п	Позначення	Найменування	Кільк. аркушів	Примітки
1				
2		Документація		
3				
4	**ІТКІ.КР.22.08.ДА.ПЗ**	Пояснювальна записка	63	Формат А4
5				
6		Презентація		
7				
8		Диск CD-R з презентацією	1	Диск CD-R
9				

					ІТКІ.КР.22.08.ДА.ПЗ.				
Зм.	Ар-куш	№ докум.	Підпис	Дата					
Розроб.		Піньковська А. К.			**Матеріали кваліфікаційної роботи**	Літ.		Аркуш	Арку-шів
Керівник		Соколова Н.О.				Н		1	1
Рецензент		Клименко А.В				НТУ «ДП», 12; 126-22м-1			
Н.контр.		Коротенко Г.М.							
Зав. каф.		Гнатушенко В.В.							

ДОДАТОК Б

CustomTheme.jsx

```
const darkPalette = {
  palette: {
    mode: "dark",
    primary: {
      main: "#1F6091",
      dark: "#575756",
      contrastText: "#FFFFFF",
    },
    secondary: {
      light: "#FFFFFF",
      main: "#D9D9D9",
      dark: "#999999",
      contrastText: "#000000",
    },
    background: {
      paper: "#212121",
      default: "#121212",
    },
    text: {
      primary: "#FFFFFF",
      secondary: "#BDBDBD",
      disabled: "#616161",
    },
```

Встановлення активної теми застосунку

```
const activeTheme = useMemo(
    () =>
      createTheme({
        ...activePalette,
        ...Theme(activePalette),
      }),
    [activePalette],
  );
```

CustomSelect.jsx

```
import {useTranslation} from "react-i18next";
import {InputLabel, FormControl, Select, MenuItem} from "@mui/material";

export default function CustomSelect({caption, value, data, onChangeHandler, disabled})
{
```

```
  const {t} = useTranslation();

  return (
    <FormControl disabled={disabled}>
      <InputLabel>{t(caption)}</InputLabel>
      <Select value={value ? value.id : ""} label={t(caption)} onChange={(e) =>
onChangeHandler(data.find((element) => element.id === e.target.value))}>
        {data.map((element, index) => {
          return (
            <MenuItem key={index} value={element.id}>
              {t(element.caption)}
            </MenuItem>
          );
        })}
      </Select>
    </FormControl>
  );
}
```

Кастомізація стилю елементів в компоненті теми

```
  MuiDialogContent: {
        styleOverrides: {
          root: {
            "& .MuiTableCell-root": {
              padding: "4px 8px 4px 8px",
            },
          },
        },
      },
```