**Зміст**

1. Вступ
2. База практики
   1. Контактна інформація
   2. Опис бази практики
3. Огляд літератури
4. Опис завдання
5. Виконання завдання
6. Висновок
7. Список використаної літератури

**Вступ**

Увесь сучасний світ практично «живе» в інтернеті, тому створення та підтримка веб-сайтів є життєво необхідним. Крім того різні підприємства, навчальні заклади, міністерства мають свою власну сторінку в Інтернеті. Це найкраща реклама, інформаційний портал для будь якої організації, важливий крок компанії для розширення кордонів власного бізнесу та здобуття нової аудиторії. Так само для фахівця своєї справи веб-сторінка – це чудова можливість розмістити своє резюме та показати свої здібності.

Проте інформаційні сторінки – це всього лиш невелика частина того, що можна зустріти на просторах Інтернету. Соціальні мережі, інтернет-магазини та інші види вебсайтів є не менш поширеними та необхідними. Місце для спілкування, ведення бізнесу, все це можна робити на вебсайтах в інтернеті, і саме тому фахівці зі створення веб-продукції такі необхідні.

1. **База практики**
   1. **Контактна інформація**

Код ЄДРПОУ: 38821414

Україна, 79000б м. Львів, вул. Володимира Великого 4, 405

E-mail: [info@linkupst.com](mailto:info@linkupst.com)

* 1. **Опис бази практики**

LinkUp Studio – продукто-орієнтована компанія, що займається створенням програмного забезпечення впродовж 5 років. Заснована у 2013 році у Львові, осередку ІТ в Україні. Реалізує проекти разом з партнерами більш як з 20 країн по всьому світу. Команда складається з 25+ професіоналів, що розробляють веб і мобільні додатки.

Бізнес експертиза включає маркетплейси різного типу, соціальні мережі, інформаційні сервіси, онлайн магазини, вебсайти у таких індустріях: туризм, мистецтво, їжа та напої, дозвілля, онлайн освіта, благодійність, спорт та фітнес, здоров’я, готельно-ресторанний бізнес, нерухомісь, роздрібна торгівля та інше.

Технології LinkUp Studio:

* Web design, UI/UX expertise;
* Back-end development: Ruby on Rails, PHP (Laravel), NodeJS;
* Front-end development: HTML5, CSS3. Ionic, AngularJS, JavaScript, jQuery, React Native, ReactJS;
* CMS custom development;
* Mobile development (Android, iOS);
* SaaS-applications;
* QA Testing (Manual, Automation).

Цінності компанії, які допомагають підтримувати унікальну атмосферу в компанії:

1. Робити те, що любиш;
2. Вкладати в продукт;
3. Сприймати проект особистісно;
4. Працювати в одній команді з замовником;
5. Підтримувати чесні і довірливі стосунки;
6. Цінувати проактивність і потік ідей;
7. Заохочувати соціальну відповідальність.
8. **Огляд літератури**
   1. **Теорія**

Веб-розробка – процес створення веб-сайту чи веб-аплікації. Основними етапами процесу є веб-дизайн, верстка сторінок, програмування для веб на стороні клієнта і сервера, а також конфігурування веб-сервера.

В основі ідеї WWW лежать так звані hypermedia документи або web-документи. Ці документи можуть містити як текстову, так і не текстову інформацію (наприклад, зображення чи звук), а також посилання. Посилання – це покажчики, за допомогою яких можна вільно помилатися на окремий документ, який може знаходитися на іншому кінці світу.

HTML – це мова, яка дозволяє представляти інформацію в Інтернеті.

CSS – спеціальна мова, що використовуються для відображення сторінок, написаних мовами розмітки даних. Найбільше CSS використовують для візуальної презентації сторінок.

JavaScript – базується на принципах прототипного програмування. Найпоширеніше і найвідоміше застосування мови – написання сценаріїв для веб-сторінок, але, також, використовується для впровадження сценаріїв керування об’єктами вбудованими в інші програми. JavaScript – скріптова мова, призначена для створення інтерактивних веб-сторінок.

Angular – це платформа для розробки мобільних і десктопних веб-аплікацій. Вона дозволяє з «коробки» створювати великі і складні аплікації.

В основному, Angular призначений для створення комплексних enterprise-застосунків, а саме односторінкових веб застосунків (Single Page Application).

SPA – веб застосунок, розташований на одній фізичній HTML сторінці. Така сторінка одноразово завантажує усі необхідні ресурси (JavaScript, CSS, images тощо.) і більше не перезавантажується. Переходи за посиланнями не приводять до реального перезавантаження сторінки, а її вміст змінюється «на льоту», тобто динамічно. За необхідністю виконується запит на сервер для отримання даних, і після їх отримання формується контент «нової» сторінки.

Всередині фреймворку реалізовано:

* Модульність;
* Анімації;
* Маршрутизація;
* Робота з бекендом;
* Зберігання/обробка/відображення даних;
* Робота з формами та шаблонами тощо.

Архітектура Angular:

* + Module;
  + Component;
  + Template;
  + Service;
  + СRouter;
  + Pipe;
  + Directives.

**Модулі (Module)**

Модулі **–** структурні одиниці застосунку, які інкапсулюють певну логіку. В Angular це структури, які зберігають певні компоненти, директиви та сервіси, об’єднані певною логікою.

**Компонента (Component)**

Компонента– typescript клас, який зберігає дані та логіку відображення цих даних у шаблоні (представленні). Шаблон тісно пов’язаний з компонентом. Дані з комонента можна з легкістю відображати у шаблоні, використовуючи спеціальний синтаксис. Компонент може також «знімати» дані з шаблона та отримувати їх безпосередньо у скрипті.

Компонента керує відображеннями предствалення на екрані. Як правило, компоненти використовуються для створення простого віджету в користувацькому інтерфейсі, в той же час вони можуть представляти з себе набір ще більш простих компонентів всередині себе.

**Шаблон (Template)**

Шаблон – фрагмент html-коду з додаванням спеціального синтаксису. Він дозволяє впроваджувати в шаблон дані з компонента без використання innerHTML та подібних методів. Шаблон прописується у компоненті та є частиною конфігурації.

Шаблон – це html-розмітка, в якій можна описувати взаємодії з DOM на основі моделі даних і подій класу компонента.

**Сервіс (Service)**

Сервіс в Angular являє собою typescript класи, які виконують задачі, пов’язані з отриманням, зберіганнями та обробкою даних. Наприклад, логування, перетворення даних для подальшої передачі у компонент, звернення до backend та ін. На відміну від компонентів та директив сервіси не працюють з представленнями (шаблонами) напряму.

**Задачі сервісів:**

* Надання даних застосунку. Сервіс сам може зберігати дані у пам’яті або, з метою отримання даних, звертатися до якогось джерела даних, наприклад, до сервісу;
* Сервіс може організувати канал взаємодії між окремими компонентами застосунку;
* Сервіс може інкапсулювати бізнес-логіку, різноманітні обчислювальні задачі, задачі з логування, які краще виносити поза компоненти. Таким чином, код компонентів буде зосереджений, безпосередньо, на роботі з представленням. До того ж можемо розв’язати проблему повторення коду, якщо нам знадобиться виконати одну й ту саму задачу у різних компонентах і класах.

**Роутер (Router)**

Роутер– маршрутизатор, який призначений для переходу між керанами з метою відображення різного контенту. Іншими словами, коли в адресному рядку браузера змінюється фрагмент URL, маршрутизатор відстежує ці зміни та завантажує ту або іншу частину застосунку.

**Особливості Angular**

**Виявлення змін**

Кожна компонента має свій власний детектор змін, який гарантує перевірку привязок даних, визначених шаблоном.

**Введення залежностей**

Введення залежностей (англ. Dependency Injection) – це композиція структурних шаблонів проектування, при якій за кожну функцію аплікації відповідає один, умовно незалежний об’єкт (сервіс), котрий може мати необхідність використовувати інші об’єкти (залежності), відомі йому інтерфейси. Залежності передаються (вводяться) сервісу в момент його створення.

**Директиви**

Директиви дозволяють отримувати прямий доступ до DOM елементів. Вони бувають двох видів: структурні і атрибутні.

Атрибутна директива:

Вводиться сервіс, який представляє посилання на елемент, до якого буде застосовуватись директива.

Структурна директива:

Структурні директиви змінюють структуру DOM за допомогою додавання чи видалення html-елементів. Існує мінімум три вбудовані структурні директиви: ngIf, ngSwitch та ngFor.

**Пайпи**

Пайпа (pipe) – являє собою особливий обробник, який дозволяє форматувати значення, які відображаються.

**Web Workers**

Підтримка Web Worker в Angular призначена для спрощеного розпаралелювання в аплікації. Коли аплікація запускається, Angular проводить всю основну роботу обробки логіки в окремих потоках, ядро виконує розрахунки в своєму робочому потоці, в той час як інші функції можуть взагалі не виконуватися в потоках.

**HTTP**

Найпоширеніший спосіб отримати дані від web-служб – це через HttpClient сервіс доступний для введення залежностей в компоненти. Angular HttpClient доволі простий. Все що потрібно зробити, це викликати метод get і передати йому url. Даний метод get повертає об’єкт Observable. Цей клас являється частиною бібліотеки rxjs, яка використовується у багатьох місцях Angular’у.

Подібно Promise, Observable не містить в собі одразу значення. Замість цього в нього є метод subscribe, де можна зареєструвати зворотній виклик (callback). Цей callbackвикликається як тільки результат буде доступний. Крім promise, Observable може повернути більше одного значення, або навіть потік результатів.

**Angular Router**

Angular Router – потужний JavaScript роутер, створений командою розробників Angular. З бібліотекою можна визначити декільки елементів router outlet, декілька стратегій зіставлення маршруту з рядком запиту, з легкістю отримувати параметри маргруту та використовувати хуки для авторизованого доступу до компоненів. Роутер дозволяє створювати односторінкові застосунки з кількома представленнями та навігацію ними.

*Router-outlet* – директива з бібліотеки роутера, що визначає місце компонента. Зіставлення маршруту проходить через URL браузера.

*Маршрути та шляхи* – об’єкти, що обов’язково вміщують шлях та атрибути компонента (або redirectTo шлях). Шлях належить до частини URL, що визначає унікальне представлення для відображення, а атрибути компонента визначають який компонент Angular буде пов’язаний зі шляхом. Кожен маршрут зіставляє URL-шлях з компонентом. Шлях може бути пустим. Це означає, що він використовується за замовчуванням і, зазвичай, при запуску.

*Стратегії зіставлення маршрутів*

Angular Router передбачає різні стратегії зіставлення маршрутів. За замовчуванням роутер переглядає початок URL та зіставляє його з відповідним маршрутом.

*Параметри маршруту*

Створення маршрутів з параметрами – поширена фіча у веб-застосунках. Angular Router дозволяє отримати доступ до параметрів різними способами:

* Використовуючи сервіс ActivatedRoute;
* Використовуючи ParmMap observable.

*Хуки маршрутів*

Хуки маршрутів – фіча, що дозволяє розробниками виконувати деяку логіку при запиті маршруту. Так можна контролювати доступ користувача. Зазвичай, хуки використовують, щоб перевірити авторизацію користувача перед переходом на сторінку.

*Директива перенаправлення*

В Angular Router передбачено директиву routerLink для створення посилань перенаправлення. Ця директива приймає шлях, пов’язаний з компонентом, до якого здійснюється перенаправлення.

*Декілька outlet та допоміжні маршрути*

Angular Router підтримує декілька елементів outlet в одному застосунку.

У компонента є основний пов’язаний маршрут, але він також може мати декілька допоміжних. Допоміжні маршрути дозволяють здійснювати перенаправлення декількома маршрутами одночасно.

Для створення такого маршруту знадобиться роутер з визначеним атрибутом name. Там буде відображатися компонент, пов’язаний з допоміжним маршрутом.

* Outlet без атрибути name – первинний;
* Усі outlet, окрім первинного, повинні визначати атрибут name.

Потім за допомогою атрибути outlet можна вказати де слід відображатися компоненту.

1. **Опис завдання**



Рисунок 1. Дизайн сторінки.

**Глобальні вимоги:** Сторінка повинна бути максимально схожою до дизайну. Підтримка розмірів екранів мобільних пристроїв обов’язкова (responsive layout) так само як і підтримка браузерів Chrome та Mozilla Firefox останніх версій (Edge і Safari - бажана).

Завдання 1

**Вимоги:** зверстати головну сторінку та сторінку з продуктами.

**Детальний опис:**

Сторінка складається з наступних секцій (section):

* шапка/header (повинна мати фіксовану позицію у вікні браузера, тобто не змінювати позицію відносно вікна при прокрутці (скролі) сторінки),
* секція з пошуком (повинна автоматично масштабуватись до розмірів активного вікна),
* секція з плитками зображень (файли зображень повинні мати різне співвідношення сторін, але відображатись вони мають відповідно дизайну),
* секція з відгуками,
* футер/footer.

Сторінка з продуктами складається з продуктів та кнопки для додавання продуктів.

Завдання 2

**Вимоги:** зверстати форму для логування та реєстрації, а також для додавання нового продукту.

**Детальні вимоги:** на кожне поле продукту повинна бути валідація (кількість символів, введення тільки цифр і т.д.)

1. **Виконання завдання**

**Перший крок – верстка сторінки та її стилів.**

Домашня сторінка складається з п’яти частин: навігація, головна частина, частина з продуктами, частина з відгуками та футер.

Кнопки навігації відображені списком та розміщені за допомогою властивості flex, яка дозволяє гнучко розташувати об’єкти на сторінці.

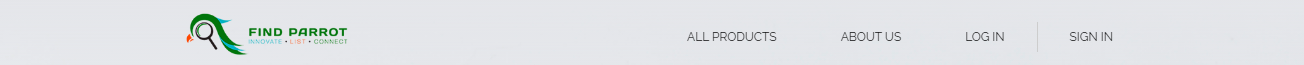


Рисунок 2. Навігація.

При зміні ширини екрану горизонтальний список кнопок зникає і на його місці з’являється кнопка, яка відкриває стовпець посилань навігації. При натисканні кнопки її положення змінюється з трьох гористонтальних ліній до двох перехресних.

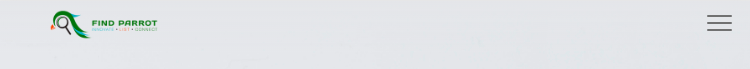


Рисунок 3. Навігація з кнопкою.

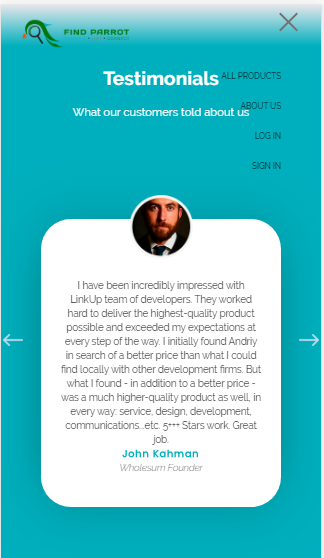


Рисунок 4. Навігація з відкритим меню.

Навігація фіксована до верху екрану і її позиція не змінюється при скролі екрану.

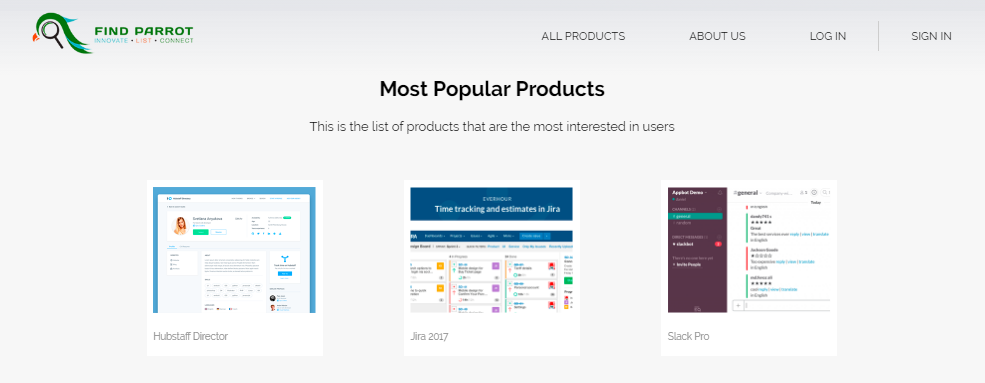


Рисунок 5. Навігація з фіксованою позицією.

На фоні головної частини прикріплене основне фіксоване зображення. Головний текст, групи введення та кнопки розміщені за допомогою flex з напрямком column. Висота основної сторінки рівна 100vh, тобто рівна висоті екрану.

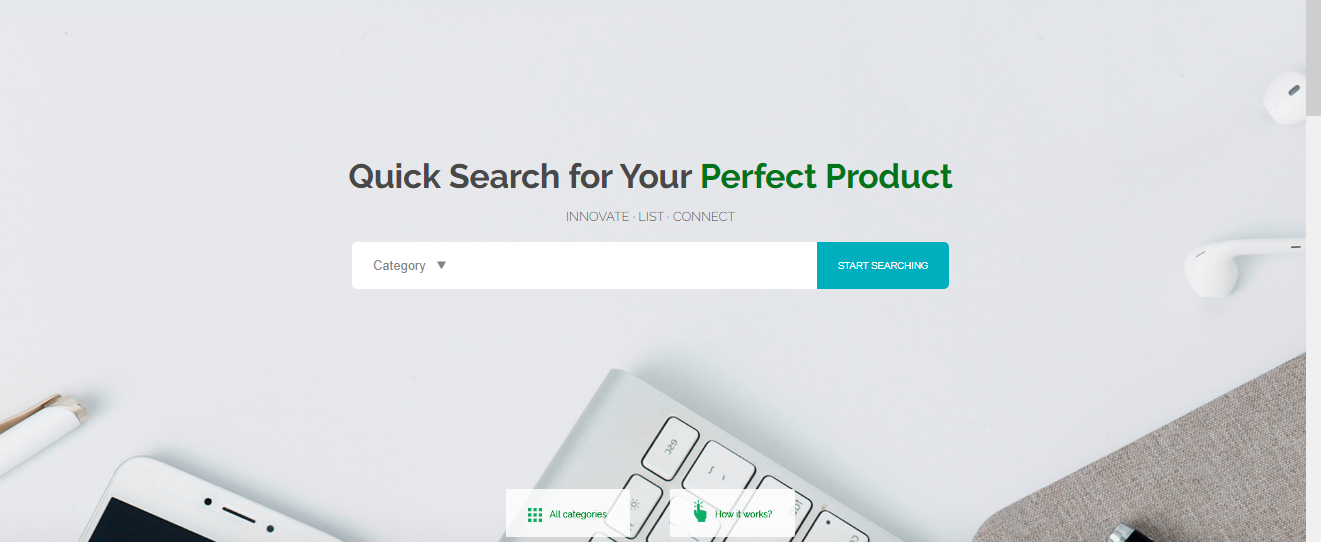


Рисунок 6. Головна сторінка.

Частина з продуктами зображена за допомогою властивості grid, котра, на відміну від flex зберігає фіксовану кількість стовпців та рядків. Відповідно до зміни ширини екранів змінюється кількість стовпців. Посилання All products перенаправлятиме на нову сторінку зі всіма продуктами.

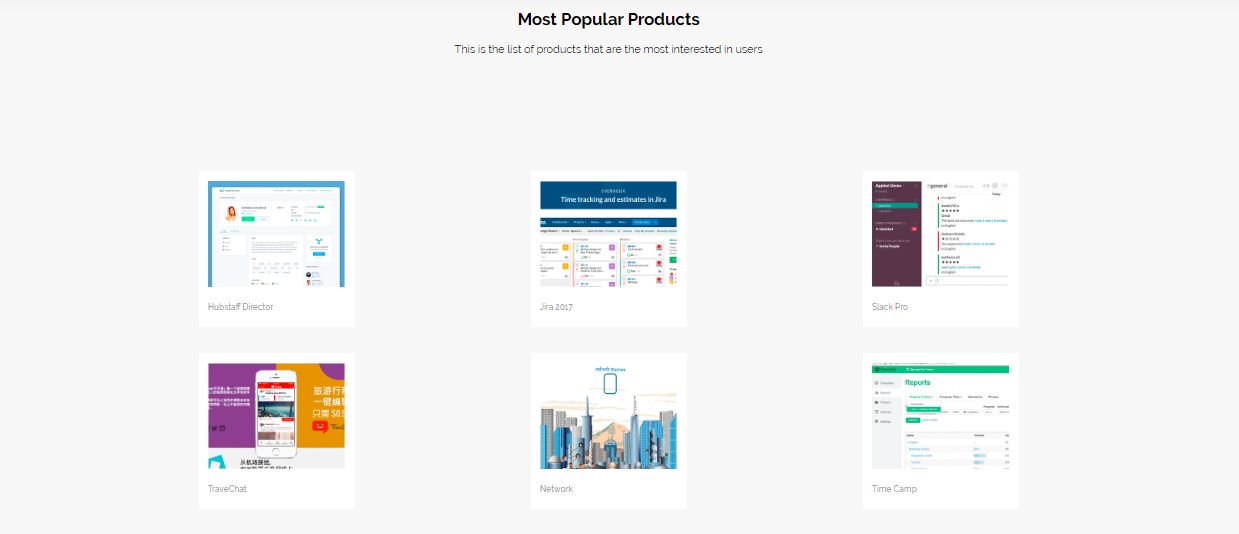


Рисунок 7. Сторінка з продуктами.

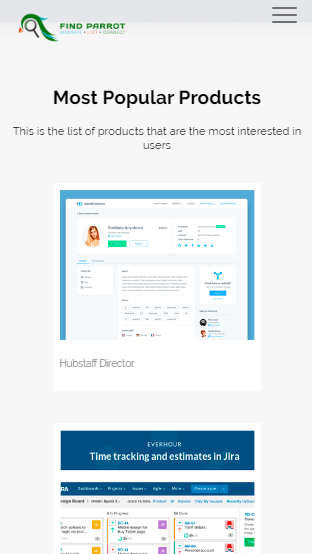
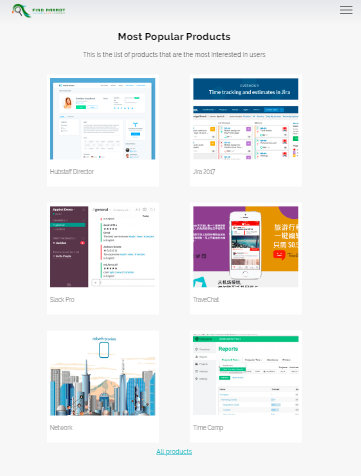


Рисунок 8. Сторінка з продуктами в планшетному та телефонноу режимі відповідно.

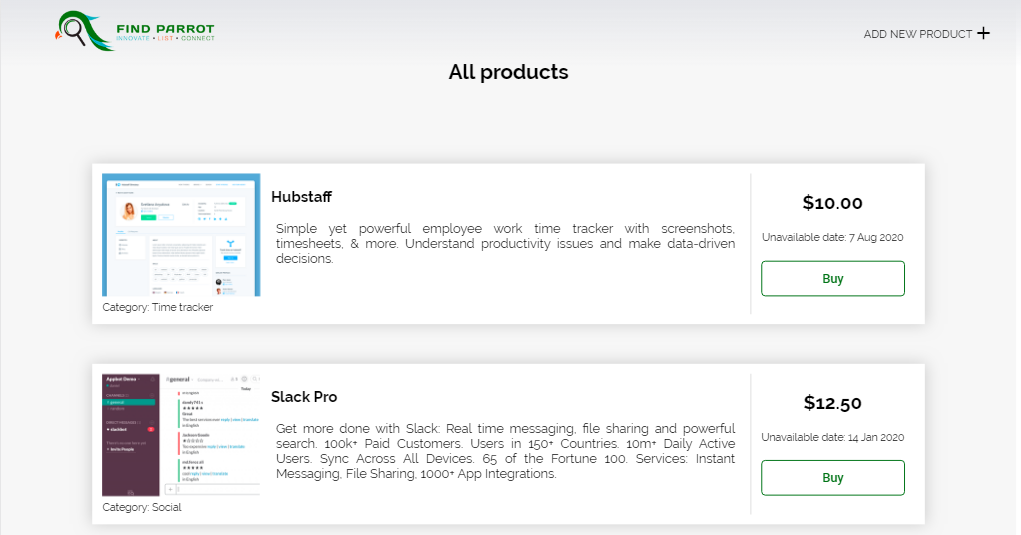
Сторінка зі всіма продуктами складається з навігації, де розміщена іконка бренду, та кнопка для додавання продуктів та загалом всіх продуктів. Продукти складаються з трьох частин: зображення, інформаційний текст та кнопки для покупок, розміщені за допомогою властивості flex, котра при зміні екрана до розміру смартфона, змінює свій напрям з row до column. А також футера.

Рисунок 9. Сторінка зі всіма продуктами.

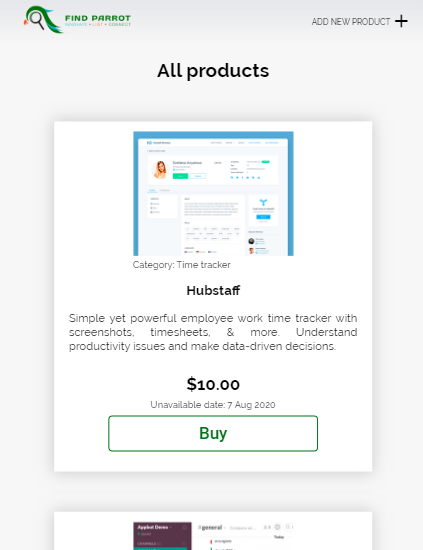


Рисунок 10. Сторінка зі всіма продуктами в телефонному режимі.

**Другий крок – створення Angular проекту.**

Для початку потрібно встановити утиліту @angular/cli для створення структур проекту. Для цього в консолі в папці проекту вводиться команда npm install –g @angular/cli.

Далі за допомогою команди ng new project створюємо новий проект.

Після встановлення, у створеній теці можна побачити наступну структуру:

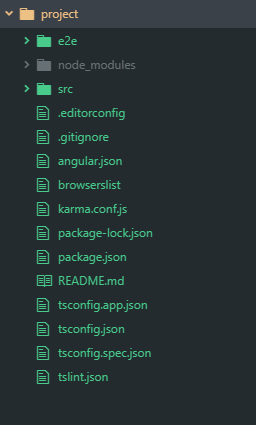


Рисунок 11. Проект Angular.

Файл tslint.json – конфігураційний файл для статичного аналізатора синаксису TypeScript. Файл tsconfig.json – конфігураційний файл компілятора TypeScript. Окрім усього іншого, в ньому зазначено версію, в яку буде перетворено код TypeScript при викнанні збірки проекту. Файл package.json зберігає всі NPM пакети, необхідні для проекту. Ці пакети буде завантажено до теки node\_modules. Під час створення нового проекту усі необхідні модулі буде завантажено автоматично завдяки Angular CLI. Тека e2e призначена для тестування. І нарешті, у теці src знаходяться головні файли нашого проекту.

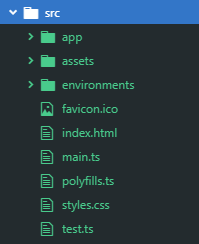


Рисунок 12. Головні файли проекту.

Завантаження застосунку починається з файлу main.ts. У ньому імпортуються усі необхідні файли. Саме в цьому файлі розташована функція, яка безпосередньо запускає застосунок.

У файлі index.html знаходиться спеціальний тег <app-root></app-root>, в якому відображається все, що генерує фреймворк.

У файлі style.css розміщуються глобальні стилі, які застосовуватимуться на всьому проекті.

У теці assets зберігаємо всі необхідні зображення для веб застосунку.

У теці app зосереджено програмний код застосунку.

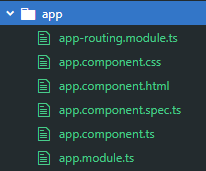


Рисунок 13. Програмний код застосунку.

Файл app.module.ts – кореневий модуль нашого застосунку. У ньому, за допомогою імпортів, приєднано необхідні функціональні елементи. За допомогою декоратора @NgModule, ми оголошуємо кореневий модуль.

Перший компонент має назву app.component.ts. У ньому є декоратор та клас самого компоненту. Декоратор вміщує в себе:

* selector – HTML селектор, який влаштовується в шаблон. На його місці визначається сам компонент в селектор <app-root></app-root>.
* tamolateUrl – шаблон, який належить цьому компоненту;
* styleUrl – таблиця стилів, що належить даному компоненту.

У теці розташований також роутинг модуль, app-routing.module.ts, в якому вказються маршрути, які є у застосунку та яким чином ними слідувати.

Для того, щоб запустити процес збірки, у папці проекту в консолі запускаємо сервер за допомогою команди ng serve. В результаті, проект буде запущений за адресою localhost:4200.

Згідно з завданням, аплікація має дві основні сторінки: домашня сторінка та сторінка з продуктами. Для цього в консолі вводимо команду ng generate component home (для домашньої сторінки) ng g c all-products (для сторінки з продуктами).

Оскільки домашня сторінка складається з п’яти частин, необхідно створити п’ять компонент відповідно. Для цього у папці home генеруємо компоненти.

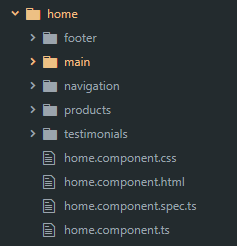


Рисунок 14. Компонента домашньої сторінки.

Далі – наповнюю компоненти відповідним кодом та стилями. В компоненті home.html об’єднуємо компоненти основної сторінки за допомогою селекторів <app-navigation></app-navigation> <app-main></app-main> <app-products></app-products> <app-testimonials></app-testimonials> <app-footer></app-footer>.

Створюю компонент all-products і заповнюєю його контентом.

У файлі app.routing.module.ts створюємо route для навігації між сторінками. Пустий шлях за замовчуванням відкриває основну сторінку. Шлях /products перенаправляє на сторінку з продуктами. Шлях /\*\*/ означає будь який інший шлях, який не є прописаний в Routes і він повинен або відправляти на головну сторінку, або на сторінку з відповідним повідомленням.

Для цього генерую новий компонент page-not-found, заповнюю його заголовком, що повідомляє користувача, що він заблудився, і кнопкою, яка повертає його на основну сторінку.

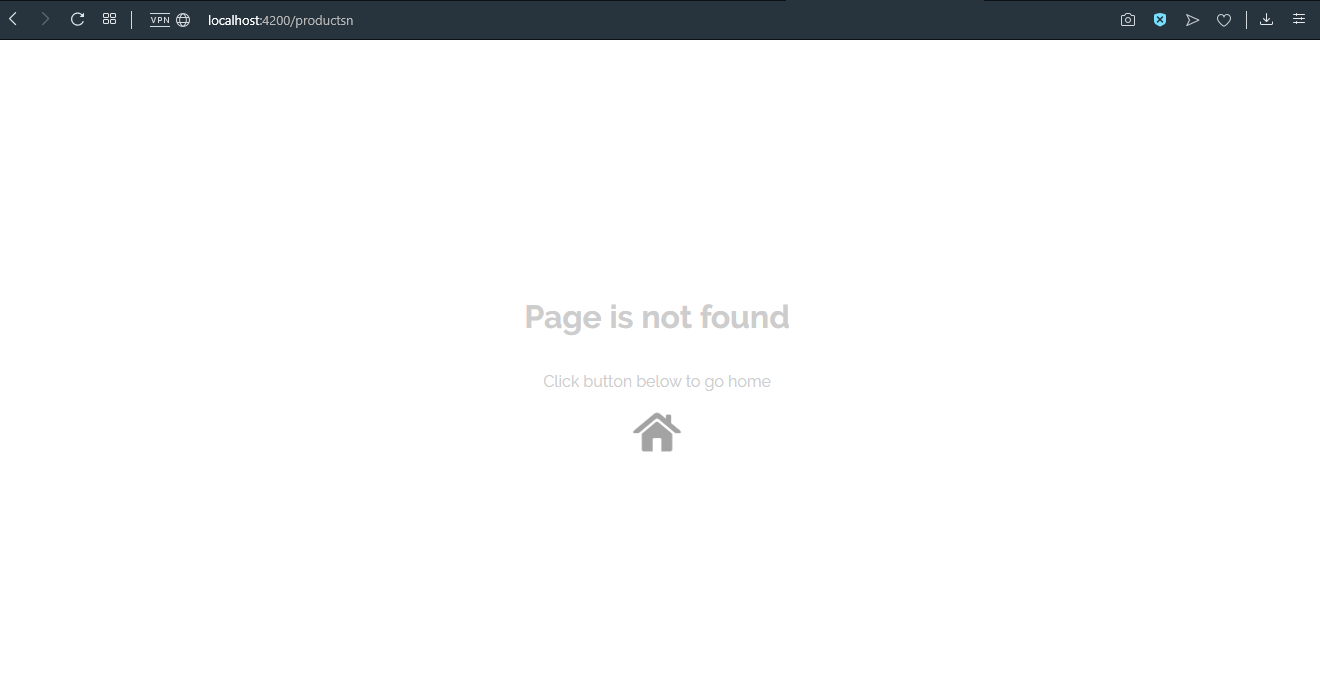


Рисунок 15. Сторінка з невідомим шляхом.

Наступний крок – створення форм для реєстрації, авторизації та додавання продукту.

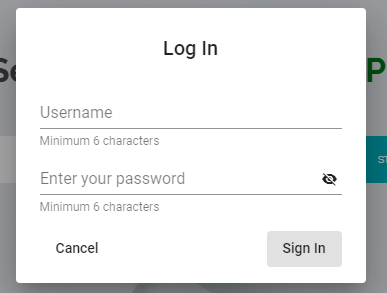
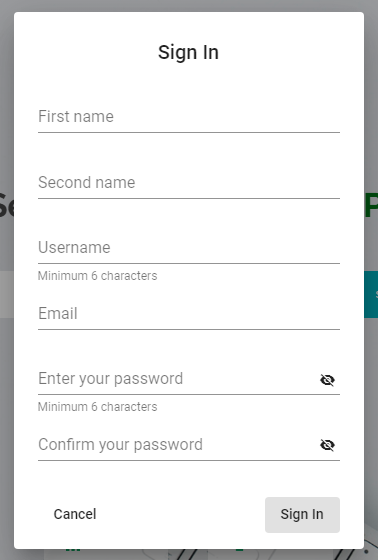
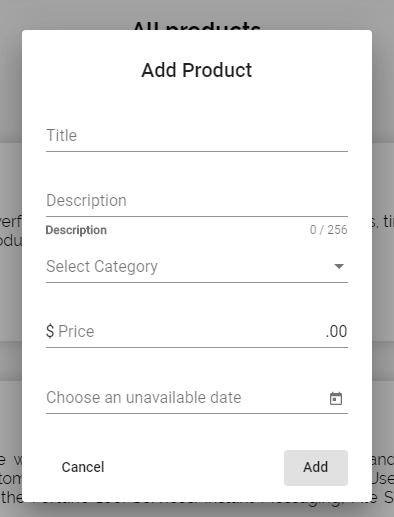
 

Рисунок 16. Модальні вікна.

Форми відкриватимуться за допомогою модульного вікна, тож непотрібно створювати зайвих шляхів у роуті.

Для того, щоб користуватись ангулярівськими модульними вікнами, необхідно встановити модуль Angular Material. За допомогою сервісу MatDialog відкриваються модальні вікна. В компоненти вписуємо функції, які відкривають в модальному вікні необхідні компоненти.

Для верстки форм використовуються встроєні директиви, які мають власні стилі, які імпортуємо в основний css-файл проекту. Для випадаючих списків, кнопок, іконок та ін. використовуємо готові директиви Angular Material.

Для валідації введення у форми використовуємо модуль FormsModule та Validators. У компоненті пишемо функції котрі провіряють правильність введення та динамічно відображають помилки, які відображаються за допомогою директиви mat-error. Перевірка помилок відбуваються за допомогою функцій модуля Validators, котрі перевіряють кількість символів, правильність введення електронної адреси, тощо.

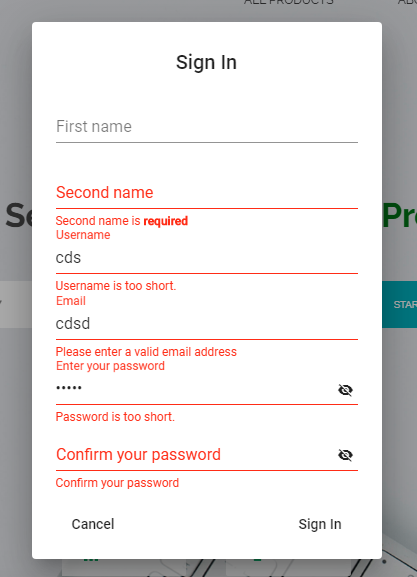
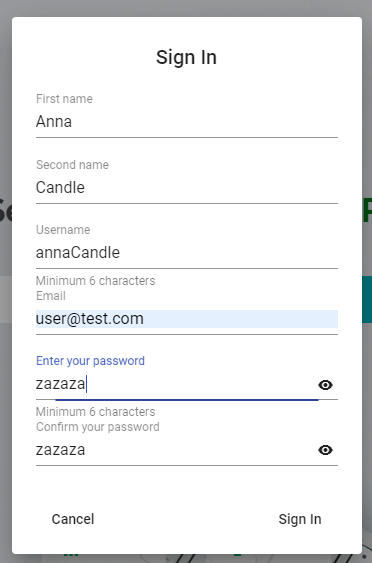
 

Рисунок 17. Валідація форм.

**Висновки**

Під час проходження проектно-технологічної практики я ознайомилась зі структурою, штатом і спеціалізацією підприємства і отримала практичні навички по роботі з програмним забезпеченням.

На практиці я закріпила свої практичні навички зі створення веб-сторінок, навчилась верстати сторінки з responsive layout. Освоїла фреймворк Angular та навчилась робити динамічні сторінки.

У результаті проектно-технологічно практики було створено веб аплікацію максимально схожою до дизайну і згідно з отриманим завданням.

**Список використаної літератури**

1. <https://codeguida.com/post/1556>
2. <https://habr.com/ru/post/348818/>
3. <https://itcluster.lviv.ua/vitayemo-link-studio-u-lviv-it-cluster/>
4. <https://material.angular.io/components/dialog/>
5. <https://material.angular.io>
6. <https://angular.io>
7. <https://ru.wikipedia.org/wiki/Angular_(фреймворк)>