



Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна роботи бакалавра

**Інформаційна система управління підрозділами розподілених
комерційних проєктів Mozaik (на прикладі ChoiceQR)**

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проєкту:

№	П.І.Б.	Група
1	Ілечко Сергій Сергійович	КН-П-213
2	Бєлова Алена Ігорівна	КН-Д-211
3	Бєлов Максим Ігорович	КН-П-213
4	Дубська Олена Владиславівна	КН-Д-211
5	Лічкун Микита Вікторович	КН-П-213

Автор звіту:

_____ Лічкун Микита Вікторович

Експерт	П.І.Б.
Науковий керівник	доц. Самойленко Д. М.

Одеса – 2025

АНОТАЦІЯ

УДК 004.9

Л-96

Лічкун М. В. Інформаційна система управління контентом закладів харчування: Автореферат випускної кваліфікаційної роботи на здобуття бакалаврського рівня вищої освіти зі спеціальності «122 Комп'ютерні науки». - Одеса: ОТУШ, 2025 - 26 с.

Бакалаврська робота присвячена дослідженню та розробці інформаційної системи, головне призначення якої полягає в тому, щоб дозволити закладам харчування створювати, редагувати та актуалізувати цифрові меню в зручному форматі. Основним завданням системи є забезпечення гнучкого інструменту для створення, структурування та оновлення електронних меню, доступних для редагування адміністрацією закладу та перегляду відвідувачами через QR-коди та посилання.

Актуальність теми обумовлена зростаючою потребою у цифровізації обслуговування клієнтів у сфері харчування, а також необхідністю оперативного оновлення меню без залучення сторонніх спеціалістів. Запропоноване рішення дозволяє автоматизувати процес керування асортиментом страв, спрощує роботу персоналу та покращує зручність взаємодії кінцевого користувача з меню.

Платформа складається з наступних частин, кожна з яких представляє певну функціональність:

- Автентифікація
- Реєстрація та авторизація користувачів (адміністраторів ресторанів);
- Створення та редагування меню з розділами та стравами;
- Створення страв
- Зберігання загальних налаштувань онлайн меню
- Налаштування та генерація QR коду з посиланням на онлайн меню
- Можливість налаштування зовнішнього вигляду меню (теми, кольори);
- Керування зображеннями страв та іконок
- Додавання коментарів до медіафайлів
- Фільтрація страв за певними критеріями

ЗМІСТ

Технічне завдання до проекту_____	4
ВСТУП _____	8
РОЗДІЛ 1. Загальна характеристика системи_____	10
1.1. Архітектурна структура та розділення компонентів	
1.2. Технологічна основа	
1.3. Кастомізація та унікальність онлайн-меню	
1.4. Система контролю версій та організація розробки	
РОЗДІЛ 2. Лендинг та його реалізація_____	12
2.1. Опис лендінгові сторінки	
2.2. Структура та основні блоки лендингу	
2.3. Технології та інструменти для розробки лендингу	
2.4. Особливості адаптивного дизайну та візуального оформлення	
РОЗДІЛ 3. Реалізація онлайн-меню_____	19
3.1. Архітектура та структура онлайн-меню	
3.2. Реалізація фільтрації та пошуку	
3.3. Реалізація роутінгу та навігації	
3.4. Кастомізація зовнішнього вигляду меню	
ВИСНОВКИ_____	24
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ_____	26

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЕКТУ

Загальний опис.

Програмне забезпечення (ПЗ) призначене для створення незалежних інформаційних системи, яка забезпечує цифрове управління меню в закладах харчування.

Головна особливість системи полягає у поєднанні трьох незалежних частин:

- Мобільного цифрового меню (для гостей закладу);
- Адміністративної панелі (для власника або адміністратора закладу, що дозволяє керувати усім вмістом меню);
- Лендінга – сторінки, яка закликає власника харчового закладу спробувати продукт.

Призначення:

ПЗ орієнтується на забезпечення зручного механізму взаємодії між кінцевим споживачем (відвідувачем закладу) та адміністративною частиною (представником закладу) шляхом візуалізації та керування інформаційними одиницями – категоріями, позиціями меню, модифікаторами, описами, інгредієнтами тощо. Сам процес створення та керування ресторану повинен бути швидким та інтуїтивним

Вимоги до функціональності:

Реєстрація користувачів:

Функціонал реєстрації користувача покликаний забезпечити зручний та інтуїтивно зрозумілий спосіб створення облікового запису для взаємодії з CMS-системою. На цьому етапі користувач має ввести такі дані, як:

- Назва ресторану – що дозволяє ідентифікувати об'єкт обслуговування та використовується для створення піддомену сайту ресторану;
- Логін – унікальний ідентифікатор користувача в рамках ресторану;
- Телефон – як засіб оперативного контакту;
- Електронна пошта – для можливості підтвердження дій та відновлення доступу;
- Пароль – обов'язково складний, для забезпечення належного рівня безпеки.

Вхід в систему:

Для входу у систему користувач вводить свої дані у відповідну форму. Необхідно зазначити:

- Піддомен ресторану;
- Логін або електронну пошту;
- Пароль.

Лендінг (Початкова сторінка)

Головна сторінка системи Mosaic орієнтована переважно на бізнес-користувачів – власників закладів харчування, і має за мету ознайомлення з сервісом, підвищення зацікавленості та швидку навігацію. Основна інформація представлена в мультирівневому форматі з якорями та кнопками в хедері. Передбачена можливість, ознайомлення з ключовими перевагами системи.

Адміністративна панель власника ресторану (CMS):

Основне робоче місце для налаштування ресторану та керування його наповненням. Повинна забезпечити усі необхідні засоби для

- Керування ієрархією меню, а саме:
 - Розділами в межах ресторану, що дозволяє:
 - Змінювати назву розділу;
 - Додавати або редагувати його опис;
 - Керувати його видимістю (публічна/прихована);
 - Категоріями в межах розділу, що передбачає:
 - Зміну назви категорії;
 - Керування її видимістю для клієнтів;
- Керування страв в межах категорії з можливостями вказати/додати:
 - Фото (до 4 штук, вага до 2 МБ, можливість вибору обкладинки);
 - Назву, опис, калорійність, БЖВ;
 - Вагу або об'єм в альтернативних одиницях (грам, мілілітр тощо);
 - Ціну та знижку;
 - Видимість страви у клієнтському інтерфейсі.
- Налаштування зовнішнього вигляду онлайн меню
- Керування загальною інформацією про ресторан, зокрема:
 - Назва, опис, адреса закладу;
 - Основна валюта та мова обслуговування;
 - Контактна пошта та телефони;
 - Favicon сайту;
 - Посилання на соціальні мережі;
 - Графік роботи по кожному дню тижня.
- Створення та кастомізація зовнішнього вигляду QR-коду:
 - Стиль маркерів, колір, фон, прозорість;
 - Текст над або під кодом;
 - Рамка, логотип, розмір.
 - Зберігання налаштувань

Перегляд меню

Місце, з яким взаємодіють усі відвідувачі харчового закладу через посилання або QR-код. Має реалізовувати можливість:

- Фільтрацію за ціною, типом приготування, часом, вагою;
- Пошук страв;
- Вибір розділу та категорії;
- Перегляд детальної інформації про страви, яка включає:
 - Назву, фото, вагу, опис, склад, час приготування;
 - Рекомендації щодо схожих страв.
- Отримання необхідної інформації про заклад. В цей перелік входять:
 - Контактні дані (телефон та соцмережі),
 - Години роботи.
 - Адресу закладу.

Вимоги до функціональності. Опціональна частина

Мітки для відображення особливостей страв. Користувач має змогу створювати власні бейджі відповідно до концепції закладу.

Можливість вказання часу роботи розділу — наприклад, для сніданків або ланчів.

Встановлення іконок категорії.

Підтримка різних мов інтерфейсу.

Відновлення паролю

Додаткова інформація про страви. Поглиблені опції:

- Харчові уподобання,
- Наявність алергенів,
- Спосіб приготування;
- Присвоєння бейджів для фільтрації (наприклад, "Веганське" або "Гостре").

Деталізація оформлення елементів та стилів. Поглиблені можливості редагування:

- Елементів меню, карток, категорій;
- Розширене редагування кнопок;
- Налаштування фону (включно з .svg паттернами);
- Вибір готових тем, їх кастомізація та скидання;
- Стани (hover, default, pressed).

Перемикання теми CMS (світла/темна).

Функціональність для перегляду обраних страв (Кошик), їх редагування та видалення. Показує:

- Рекомендовані страви;
- Загальну вартість;
- Кнопку "Замовити".

Оформлення замовлення, де користувач обирає:

- Тип замовлення (доставка, бронювання, самовивіз);

- Персональні дані (ПІБ, телефон), адреса, час кількість персон, побажання;
- Окремі форми збору інформації для різних способів отримання замовлення;
- Спосіб оплати (готівка або картка);
- Суму замовлення;.

Архітектура

ПЗ має бути спроектоване за клієнт-серверною архітектурою з розподіленими вузлами:

- Mockup – візуальна модель інтерфейсу користувача із зазначенням зв'язків і переходів між різними частинами інтерфейсу.
- Backend – серверне програмне забезпечення, що забезпечує збереження та оброблення поточних даних, взаємодію з базою даних. Надає програмний інтерфейс взаємодії (API) з іншими вузлами
- Frontend – додаток, доступний з веб-простору (сайт), що забезпечує візуальний інтерфейс користувачів

Усі модулі ПЗ повинні мати засоби

- тестування функціональності,
- розгортання на новому пристрої (інсталяції)

Технології

Зберігання вихідних кодів ПЗ повинно здійснюватись за допомогою системи контролю версій (вибір конкретної системи узгоджується групою проекту). Структура файлів та каталогів обирається на розгуд команди, але повинна максимально відбивати архітектуру ПЗ

Для кожного з модулів рекомендується (з можливістю часткової зміни) наступні технології створення:

- Backend – .NET або Java, або NodeJS технології з хмарним розміщенням (Azure, AWS, Google Cloud),
- Frontend – серверні технології (ASP, Razor Pages, JSP) або React чи Angular
- Mockup - визначається ментором з дизайну

ВСТУП

Сучасна сфера громадського харчування активно впроваджує цифрові технології для підвищення якості обслуговування клієнтів та оптимізації внутрішніх процесів. Одним із ключових напрямків цифровізації є використання онлайн-меню, які дозволяють закладам швидко оновлювати інформацію про страви, ціни та акції без необхідності друкувати нові паперові версії.

В умовах зростаючої конкуренції та підвищених вимог до гнучкості бізнесу, заклади харчування потребують інструментів, що забезпечують просте та ефективне керування контентом меню, а також можливість адаптації зовнішнього вигляду під власний фірмовий стиль. Важливою складовою сучасних сервісів є також зручна презентація продукту для потенційних клієнтів через лендингову сторінку.

Дана робота присвячена розробці веб-платформи для створення та адміністрування цифрових меню закладів харчування. У межах проекту реалізовано лендинг для презентації сервісу, функціонал онлайн-меню, яке може мати різний зовнішній вигляд завдяки налаштуванням кольорів, шрифтів та тем оформлення. Окрім цього, розроблена CMS-система для зручного управління стравами та налаштуваннями меню, що дозволяє адміністраторам закладів оперативно додавати, редагувати та структурувати інформацію.

Актуальність теми обумовлена необхідністю оперативного оновлення інформації, підвищення зручності для клієнтів та автоматизації роботи персоналу. Запропоноване рішення сприяє підвищенню ефективності роботи закладів харчування та покращенню взаємодії з клієнтами.

Загальна характеристика системи

Розроблена система є сучасною веб-платформою для автоматизації роботи закладів харчування, яка дозволяє створювати, редагувати та публікувати онлайн-меню для ресторанів у форматі SaaS (Software as a Service). Архітектура проекту побудована за принципом SPA (Single Page Application) з розподілом на окремі застосунки для різних ролей користувачів.

1.1. Архітектурна структура та розділення компонентів

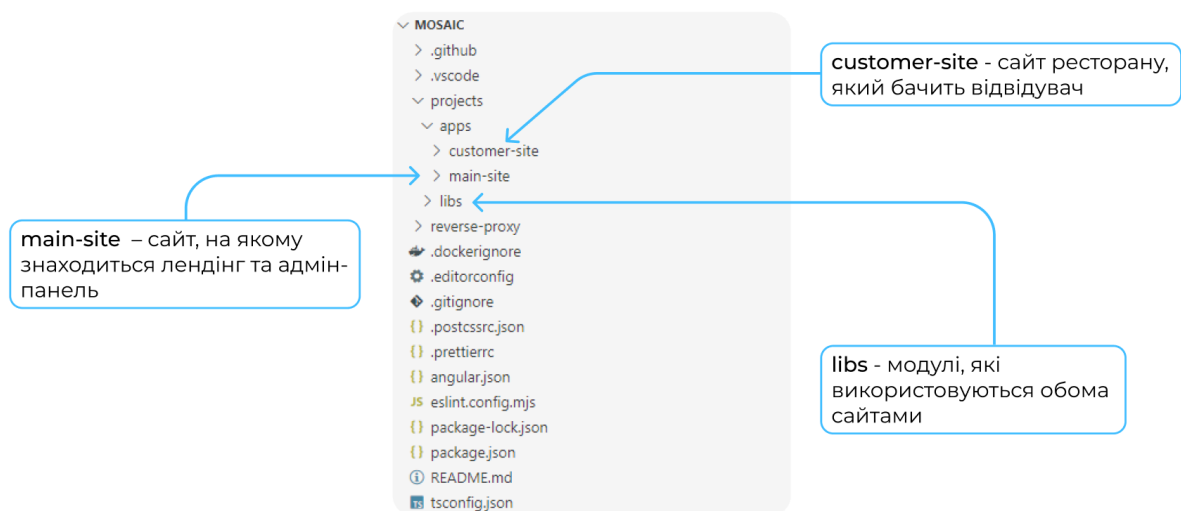


Рисунок 1.1 - Архітектурна структура проектів та розподілення

Відповідно до технічного завдання система спроектована за клієнт-серверною розподіленою архітектурою. У складі системи умовно виділено такі архітектурні частини(рис 1.1):

- Frontend (ресторан) — інтерфейс онлайн-меню, який бачить відвідувач ресторану.
- Backend (ресторан) — серверна частина, що обробляє запити онлайн-меню.
- Frontend (лендінг та CMS) — інтерфейс для адміністраторів закладів та лендинг для презентації сервісу.
- Backend (CMS) — серверна частина для керування контентом та налаштуваннями меню.

Система розбита на дві пари (Frontend та Backend), які відповідають частинам платформи управління контентом (CMS) та онлайн-меню.

Причини для розділення Frontend:

- Підтримка різних стилів: лендинг та CMS мають фірмовий стиль, а онлайн-меню — гнучку кастомізацію для кожного ресторану.
- Можливість використання різних бібліотек для оптимізації розміру файлів.

- Оптимізація сайтів ресторану, зокрема компіляція розмітки та стилів для зменшення обсягу даних.

Причини для розділення Backend:

- Можливість керування кількома ресторанами одним користувачем.
- Реплікація ресторанної частини без дублювання чутливих даних.

Загальна причина для обох частин:

Розмежування аутентифікації та авторизації: адміністратор ресторану та відвідувач мають різні права доступу, що підвищує безпеку та спрощує підтримку коду.

1.2. Технологічна основа

Для розробки клієнтської частини використано Angular, Tailwind CSS, PrimeNG, Lottie (анімації), Iconify (іконки).

Всі дані про меню, налаштування та користувачів зберігаються у централізованій базі даних, а взаємодія між клієнтською та серверною частинами здійснюється через RESTful API.

1.3. Кастомізація та унікальність онлайн-меню

Кожен власник ресторану після реєстрації отримує доступ до особистого кабінету, де може створювати ієрархію меню, додавати та редагувати страви, завантажувати зображення, налаштовувати зовнішній вигляд онлайн-меню. Система підтримує кастомізацію: власник може обирати кольорову схему, шрифти, теми оформлення, що дозволяє зробити меню унікальним для кожного закладу. Всі ці налаштування зберігаються окремо для кожного користувача і застосовуються автоматично при відображенні онлайн-меню для відвідувачів.

1.4. Система контролю версій та організація розробки

Для контролю версій використовується Git, код розміщено у двох приватних репозиторіях на GitHub — RestaurantSaasBackend та RestaurantSaasFrontend. Обрана стратегія двох монорепозиторіїв дозволяє розмежувати зони відповідальності та використовувати спільні модулі (libs) для обох частин, дотримуючись принципу DRY.

Використовується стратегія GitHub Flow, затверджені вимоги до Pull Requests та процедури code review, що забезпечує якість і підтримуваність коду.

Таким чином, система дозволяє закладам харчування оперативно створювати та оновлювати онлайн-меню, забезпечує індивідуальний стиль для кожного ресторану та підвищує зручність для кінцевих користувачів.

Лендинг та його реалізація

2.1. Опис лендінгові сторінки

Лендинг є вітриною платформи, що знайомить потенційних користувачів із можливостями сервісу, перевагами цифрового меню та мотивує до реєстрації. Сторінка містить сучасний дизайн, анімації, інтерактивні елементи та адаптивну верстку для різних пристроїв. Особливу увагу приділено зручності навігації, швидкому доступу до основної інформації та простоті взаємодії з формами реєстрації й авторизації.

2.2. Структура та основні блоки лендингу

Лендинг побудований із застосуванням сучасних технологій, має адаптивний дизайн, інтерактивні анімації та зручні форми для залучення нових користувачів. Кожен блок реалізовано як окремий Angular-компонент, що забезпечує гнучкість, масштабованість та легкість підтримки коду.

Лендинг складається з таких основних компонентів(рис 2.2.1):

- Header — верхня панель із логотипом, навігацією та бургер-меню для мобільних пристроїв.
- Hero-block — головний екран із заголовком, підзаголовком, кнопкою реєстрації та паралакс-ефектом.
- Possibilities — блок із переліком можливостей платформи, кожна з яких супроводжується Lottie-анімацією.
- Pricing-plan — секція з тарифними планами.
- FAQ — блок із відповідями на поширені питання.
- Footer — нижня частина з контактами, посиланнями та формою підписки.

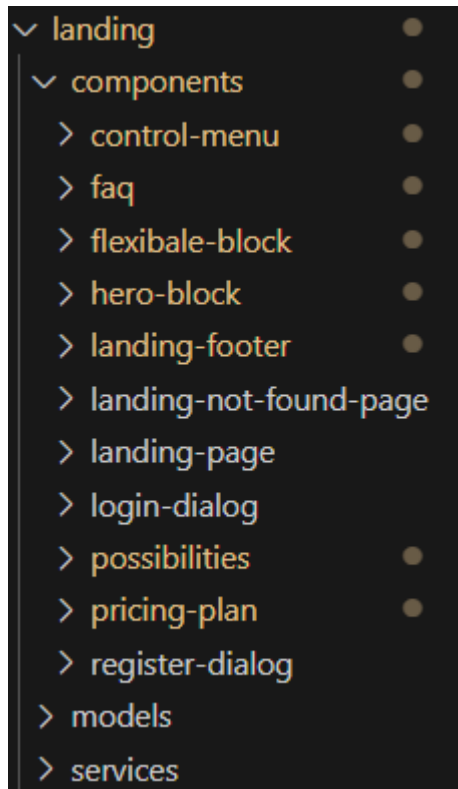


Рисунок 2.2.1 - Структура компонентів у проекті

2.3. Технології та інструменти для розробки лендингу

Для розробки лендингу використано сучасний стек:

- Angular — основний фреймворк для побудови SPA.
- Tailwind CSS — для швидкої та адаптивної верстки.
- PrimeNG — бібліотека UI-компонентів (кнопки, діалоги, форми).
- Lottie (ngx-lottie) — для інтеграції SVG-анімацій у блоках можливостей.
- Iconify — для роботи з іконками.
- ESLint, Prettier — для підтримки якості та стилю коду.

2.4. Особливості адаптивного дизайну та візуального оформлення

Адаптивність

Всі блоки лендингу розроблені з урахуванням різних розмірів екранів (мобільний, планшет, десктоп). Tailwind CSS дозволяє швидко задавати breakpoints та стилі для кожного розміру.

Приклад адаптивного бургер-меню:

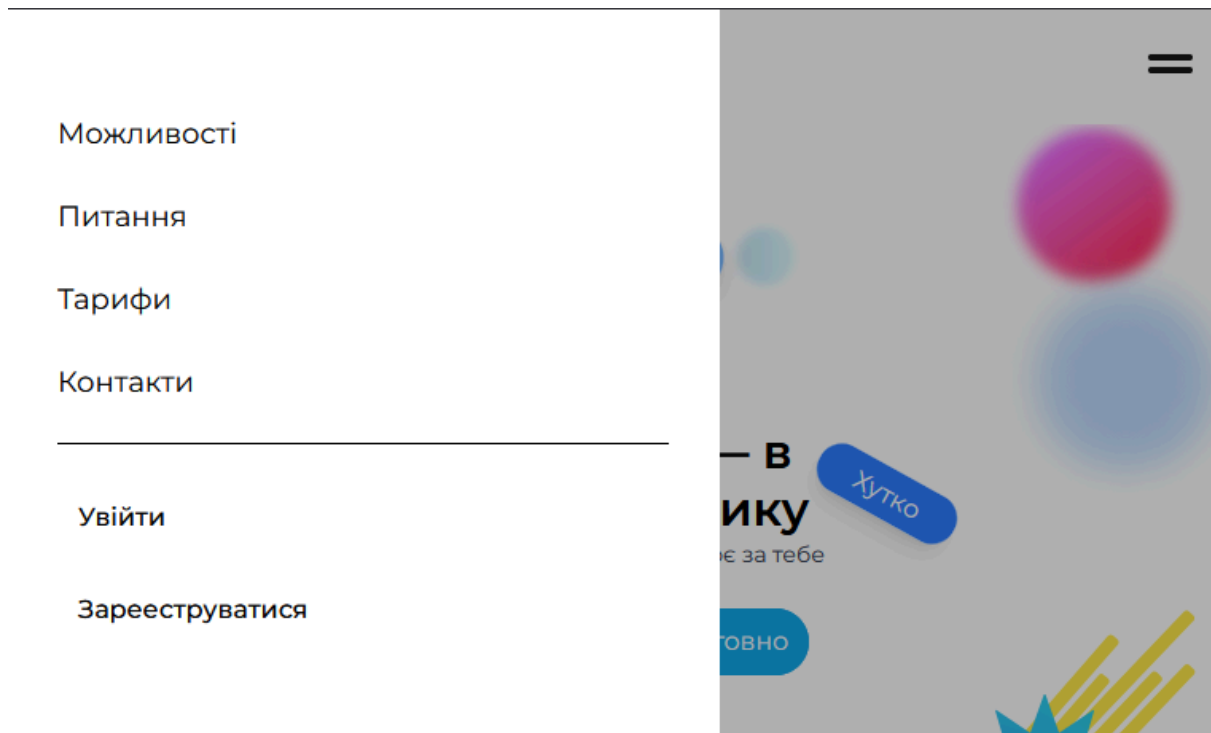


Рисунок 2.4.1 - Адаптивне бургер-меню

Параллакс-ефект

У головному блоці (Hero-block) реалізовано параллакс-ефект, який створює відчуття глибини та динаміки інтерфейсу. Суть паралаксу полягає в тому, що окремі декоративні елементи зміщуються відносно руху миші користувача, імітуючи багатошаровий простір.

Логіка роботи методу ``getParallaxStyle()``

1. Відстеження координат миші:

У компоненті зберігаються координати центру блоку та поточне положення курсора (``this.#mouseX``, ``this.#mouseY``, ``this.#centerX``, ``this.#centerY``).

2. Обчислення зміщення:

Функція приймає параметри: початкові координати елемента (x, y), кут повороту, коефіцієнт впливу (``factor``), множники по осях (``xMul``, ``yMul``) та сторону розміщення (``side``).

Далі розраховується зміщення по X та Y:

```
const dx = ((this.#mouseX - this.#centerX) * factor * xMul) / rem;  
const dy = ((this.#mouseY - this.#centerY) * factor * yMul) / rem;
```

Рисунок 2.4.1 - обчислення зміщення елементів по осях X та Y

Це дозволяє кожному елементу рухатися з різною швидкістю та напрямком, створюючи ефект багат шаровості.

3. Формування стилю:

Функція повертає об'єкт стилів для абсолютного позиціонування елемента з урахуванням обчисленого зміщення та повороту:

```
return {  
  position: 'absolute',  
  [side]: `calc(${x}rem + ${dx}rem)`,  
  top: `calc(${4 + y}rem + ${dy}rem)`,  
  transform: `rotate(${rotate}deg)`,  
};
```

Рисунок 2.4.2 - формування об'єкта стилів для позиціонування та обертання елемента

Таким чином, при русі миші елементи плавно зміщуються, створюючи сучасний інтерактивний ефект.

Ось детальний опис, як саме працює Lottie у твоєму проекті:

Як працює Lottie-анімація в Angular-проекти

Lottie — це бібліотека, яка дозволяє відтворювати складні анімації, створені у After Effects та експортовані у форматі JSON через Bodymovin. Lottie-анімації відображаються у браузері як SVG, Canvas або HTML, займають мало місця та не втрачають якості при масштабуванні.

Основні етапи роботи Lottie у лендингу:

1. Підключення бібліотеки

Для Angular використовується пакет [ngx-lottie](<https://github.com/ngx-lottie/ngx-lottie>), який обгортає оригінальну бібліотеку Lottie для зручної інтеграції з Angular-компонентами.

2. Опис анімації у компоненті

Для кожного блоку можливостей у масиві зберігається шлях до JSON-файлу анімації та налаштування:

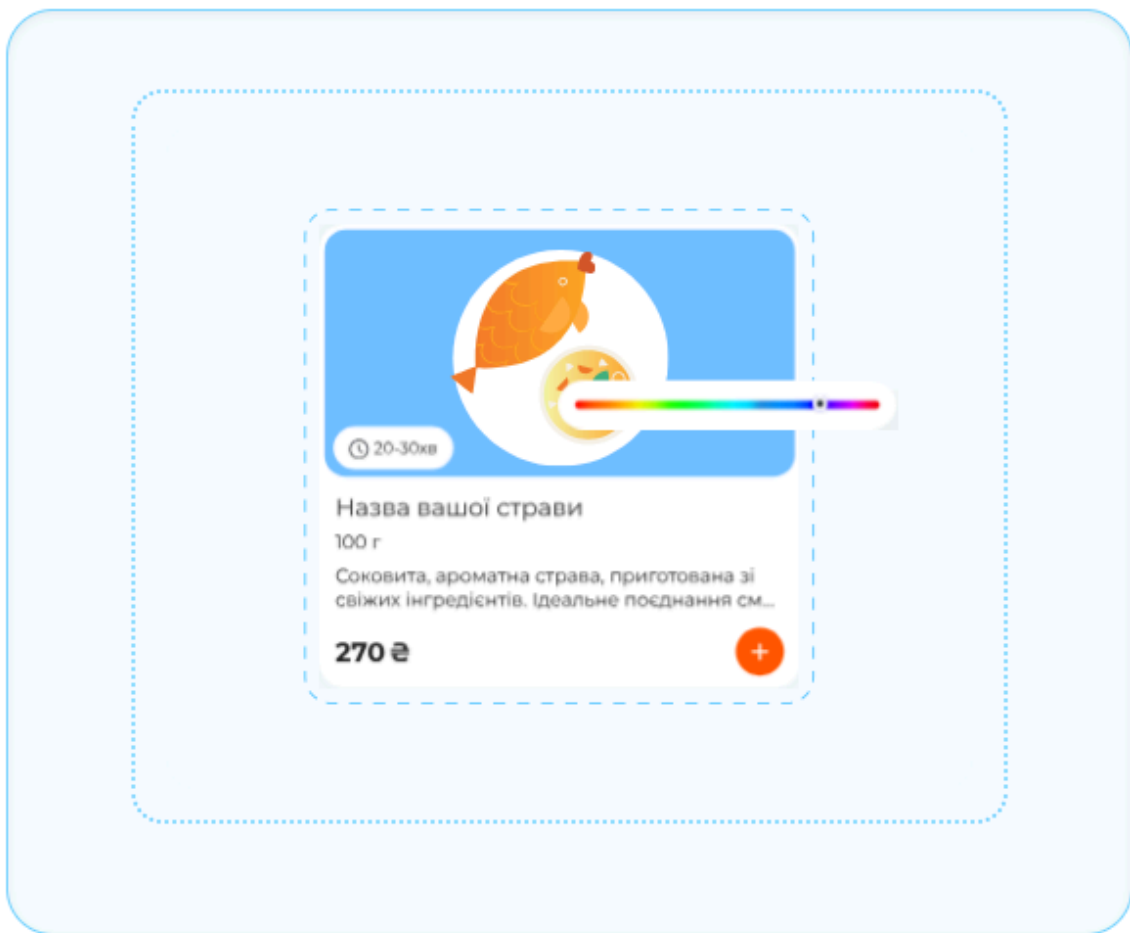


Рисунок 2.4.3 - lottie компонент

3. Відображення у шаблоні

Для відображення анімації використовується компонент ``ng-lottie``, якому передаються параметри з `options`:

```
<ng-lottie [options]="item.options"></ng-lottie>
```

Рисунок 2.4.4 - Angular-компонент `<ng-lottie>`

Lottie самостійно завантажує JSON-файл, парсить його та відтворює анімацію у SVG-контейнері.

4. Життєвий цикл та взаємодія

- Анімація автоматично стартує при появі на сторінці (autoplay).
- Якщо вказано `loop: true`, анімація повторюється нескінченно.
- Можна керувати анімацією через API (наприклад, запускати, зупиняти, змінювати швидкість або реагувати на події).
- Lottie оптимізовано для продуктивності: анімації не гальмують навіть на мобільних пристроях.

Форми реєстрації та авторизації

Для реєстрації та входу реалізовано окремі модальні діалоги з валідацією даних, використовуючи Reactive Forms та PrimeNG.

```
readonly newAccountForm = new FormGroup({
  restaurantName: new FormControl('', [Validators.required, Validators.maxLength(60)]),
  login: new FormControl('', [Validators.required]),
  phone: new FormControl('', [Validators.required]),
  email: new FormControl('', [Validators.required, Validators.email]),
  password: new FormControl('', [
    Validators.required,
    Validators.pattern(/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[A-Za-z0-9]).{6,}$/),
  ]),
  confirmPassword: new FormControl('', [
    (control: AbstractControl) => {
      return this.newAccountForm?.value.password === control.value ? null : { passwordMismatch: true };
    },
  ]),
  termsAndConditions: new FormControl(false, [Validators.requiredTrue]),
});
```

Рисунок 2.4.5 - форма реєстрації нового акаунта в Angular з валідацією полів,

Пояснення:

- Всі поля мають валідацію (обов'язковість, email, складність паролю, підтвердження паролю, згода з умовами).
- Після успішної реєстрації користувач автоматично перенаправляється до адмін-панелі.

Сервіс для роботи з акаунтами

Для обробки реєстрації, авторизації, отримання та оновлення інформації про акаунт на лендингу використовується сервіс `AccountService`. Він відповідає за взаємодію з бекендом через HTTP-запити, зберігання стану користувача, кастомізацію теми ресторану та безпечну роботу з URL. `AccountService` забезпечує всю бізнес-логіку для роботи з акаунтами на лендингу, підтримує

кастомізацію зовнішнього вигляду онлайн-меню та гарантує безпечну взаємодію з бекендом і клієнтською частиною.

Основні функції сервісу:

- Отримання інформації про поточний акаунт (`getMy`)
- Створення нового акаунта (`create`)
- Оновлення теми ресторану (`updateTheme`)
- Авторизація (`signIn`) та вихід із системи (`signOut`)
- Формування та соматизація URL ресторану для iframe

Ключові фрагменти коду та пояснення:

```
@Injectable({ providedIn: 'root' })
export class AccountService {
  readonly #apiUrl = environment.apiUrl + '/accounts';
  readonly #http = inject(HttpClient);
  readonly #domSanitizer = inject(DomSanitizer);
  readonly #currentAccount = signal<AccountInfo | null>(null);

  readonly currentAccount = this.#currentAccount.asReadonly();
  readonly restaurantUrl = computed(() =>
    environment.customerSiteOriginUrl.replace('*', this.currentAccount()?.restaurant.domain || ''),
  );
  readonly safeRestaurantUrl = computed(() => this.#domSanitizer.bypassSecurityTrustResourceUrl(this.res
```

Рисунок 2.4.6 - безпечне формування URL ресторану на основі поточного облікового запису

- Зберігає поточний акаунт у реактивному signal.
- Формує URL ресторану для перегляду онлайн-меню та санітизує його для безпечного використання в iframe.

```
getMy(): Observable<AccountInfo> {
  const account = this.#currentAccount();
  if (account) {
    return of(account);
  }
  return this.#http.get<AccountInfo>(`${this.#apiUrl}/my`, { withCredentials: true }).pipe(
    map((account) => {
      account.restaurant.themeVariables = { ...defaultTheme, ...account.restaurant.themeVariables };
      return account;
    }),
    tap((account) => this.#currentAccount.set(account)),
  );
}
```

Рисунок 2.4.6 - Метод getMy() повертає обліковий запис з кешу або виконує HTTP-запит

- Отримує інформацію про поточного користувача з сервера або повертає збережену у signal.
- Додає дефолтні змінні теми до налаштувань ресторану.

```
create(account: NewAccount) {
  return this.#http.post<CreatedAccount>(this.#apiUrl, account, { withCredentials: true });
}
```

Рисунок 2.4.8 - Реєструє нового користувача (відправляє дані форми на сервер)

```
updateTheme(variables: Partial<ThemeVariables>) {
  return this.#http.patch(`${environment.apiUrl}/themes`, variables, { withCredentials: true }).pipe(
    tap(() => {
      this.#currentAccount.update((account) => {
        if (account) {
          account.restaurant.themeVariables = { ...account?.restaurant.themeVariables, ...variables };
        }
        return account;
      }),
    ),
  );
}
```

Рисунок 2.4.9 - Метод оновлення та миттєвого застосування змін теми

- Оновлює тему ресторану (кольори, шрифти тощо) і одразу застосовує зміни у локальному стані.

```
signIn(request: SignInRequest) {
  return this.#http.post(`${this.#apiUrl}/sign-in`, request, { withCredentials: true });
}

signOut() {
  return this.#http.post(`${this.#apiUrl}/sign-out`, null, { withCredentials: true });
}
```

Рисунок 2.4.10 - методи signIn() та signOut()

- Реалізує вхід та вихід користувача із системи.

- Сервіс інjektується у компоненти реєстрації, авторизації, налаштувань теми.
 - Дозволяє централізовано керувати станом акаунта, темою ресторану та безпечно працювати з URL.

Реалізація онлайн-меню

3.1. Архітектура та структура онлайн-меню

Онлайн-меню реалізовано як окремий SPA-застосунок із чіткою модульною структурою, що забезпечує масштабованість, підтримуваність і зручність розробки. Вся логіка побудована на реактивних сигналах Angular, що забезпечує швидке оновлення інтерфейсу при зміні даних. Основні елементи структури:

Основні UI-компоненти для відображення меню:

- **menu-page** — головна сторінка меню, де відображаються секції, категорії та список страв.

- **product-card** — окремий компонент для відображення картки страви у списку.

- **product-filter** — компонент для фільтрації страв за різними критеріями (ціна, вага, категорія, теги тощо).

- **product-grid** — компонент для організації сітки карток страв.

- **product-page** — модуль для детального перегляду страви, який містить:

- **product-details** — компонент для відображення повної інформації про страву.

Сервіси для роботи з даними:

- **product-filtering.service** — логіка фільтрації страв.

- **product.service** — отримання та обробка даних про страви.

- **sections.service** — отримання структури меню (секції, категорії).

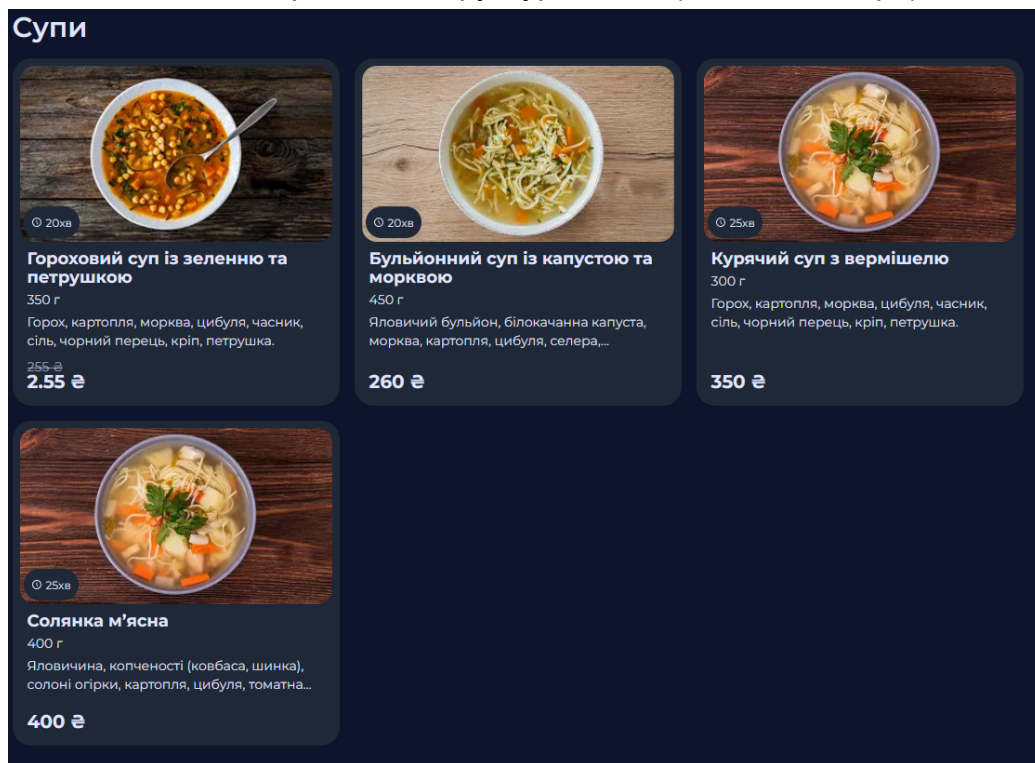


Рисунок 3.1.1- отримання страв у компоненті product-grid

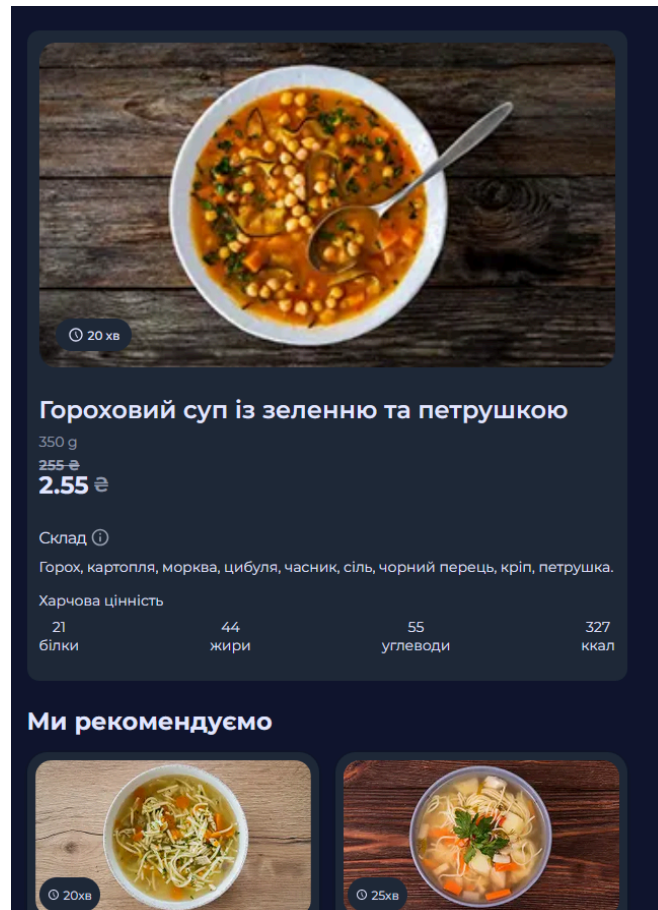


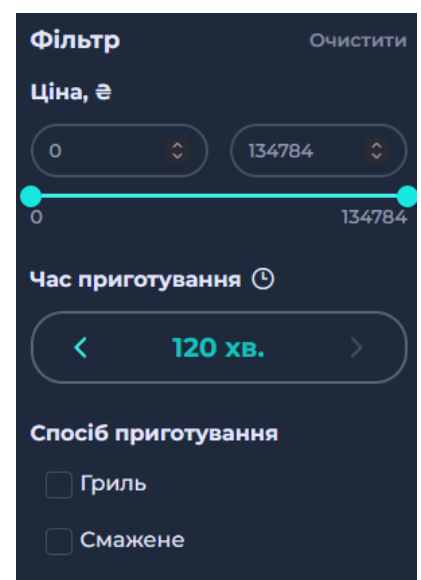
Рисунок 3.1.2- Відображення повної інформації страви

Загальна логіка роботи:

- При завантаженні сторінки через сервіси отримується структура меню (секції, категорії, страви).
- Дані передаються у відповідні компоненти для відображення.
- Користувач може фільтрувати страви за допомогою `product-filter`, а результати одразу відображаються у `product-grid`.
- При виборі страви відкривається детальна сторінка через `product-page` та `product-details`.

3.2. Реалізація фільтрації та пошуку

Для зручності користувачів реалізовано багаторівневу фільтрацію страв за ціною, вагою, часом приготування, способом приготування, дієтичними вподобаннями та тегами. Всі фільтри інтегровані у компонент `ProductFilterComponent`, який взаємодіє з сервісом `ProductFilteringService`. Фільтрація відбувається на клієнті, що дозволяє миттєво оновлювати список страв без додаткових запитів до серверу.



```
readonly filteredSections = computed(() =>
  this.#sectionsService
    .sections()
    .map((section) => ({
      ...section,
      categories: this.#filterCategoriesAndProducts(section),
    }))
    .filter((section) => section.categories.length > 0),
);
```

Рисунок 3.2.1- Метод значення обчислення, яке повертає список секцій меню, що пройшли фільтр

Рисунок 3.2.2- Фільтрація в певному онлайн-меню

3.3. Реалізація роутінгу та навігації

Роутінг

Роутінг організовано таким чином, щоб кожна сторінка меню, категорії чи продукту мала унікальний URL.

Використовується Angular Router з lazy loading компонентів для оптимізації швидкості завантаження. Передбачено обробку неіснуючих сторінок (404) та автоматичне перенаправлення залежно від стану ресторану.

```
{
  path: 'menu/:section',
  loadComponent: () =>
    import('../features/menu/components/menu-page/menu-page.component').then((m) =>
    },
  {
    path: 'menu',
    loadComponent: () =>
      import('../features/menu/components/menu-page/menu-page.component').then((m) =>
    },
  { path: '', pathMatch: 'full', redirectTo: 'menu' },
  {
    path: '**',
    loadComponent: () =>
      import('../features/global/components/item-not-found/item-not-found.component').
      (m) => m.ItemNotFoundComponent,
    ),
  },
}
```

Рисунок 3.3.1- Налаштування роутінгу Angular для онлайн-меню ресторану.

Логіка субдоменів

Кожен ресторан має власну унікальну сторінку, яка доступна за окремим субдоменом (наприклад, `subdomain.mozaik.rest`).

Це реалізовано за допомогою функції

```
export function resolveDomainFromFirstPathParam(): string {
  return window.location.pathname.split('/').filter(Boolean)[0];
}

export function resolveDomainFromSubdomain(): string {
  const parts = window.location.hostname.split('.');
  return parts.length > 2 ? parts[0] : '';
}

export const resolveSubdomain = environment.production ? resolveDomainFromSubdomain : res
```

Рисунок 3.2.2- Метод обчислення субдоменів від URL клієнта

- У продакшн-режимі визначення домену відбувається через субдомен (перший елемент у `hostname`)
- У дев-режимі — через перший параметр у шляху (для зручності локальної розробки)
-

Ця логіка дозволяє кожному власнику ресторану мати власну унікальну сторінку, а система автоматично підтягує дані саме для цього ресторану.

3.4. Кастомізація зовнішнього вигляду меню

Кожен ресторан має власні налаштування теми (кольори, шрифти, темна/світла тема), які зберігаються у базі даних і підтягуються через сервіс `RestaurantStateService`.

Після ініціалізації стану ресторану ці налаштування застосовуються до `root`-елемента сторінки через змінні CSS, що дозволяє динамічно змінювати вигляд меню для кожного користувача.

```
#applyTheme(theme: ThemeVariables) {
  const root = this.#document.documentElement;
  for (const [key, value] of Object.entries(theme)) {
    root.style.setProperty(key, value);
  }
  if (this.#isDarkThemeColor(theme['--color-background-500'])) {
    this.#document.documentElement.classList.add('dark-mode');
  }
}
```

Рисунок 3.4.1- Метод застосування теми оформлення для онлайн-меню ресторану.

- Змінні теми застосовуються до root-елемента через CSS custom properties
- Це дозволяє динамічно змінювати вигляд меню для кожного ресторану без перезавантаження сторінки

ВИСНОВКИ

Проведено огляд сучасних фронтенд-технологій для цифрових рішень у сфері громадського харчування, визначено перелік функціональних можливостей, які мають найбільшу цінність для кінцевого користувача:

- Адаптивний інтерфейс для різних пристроїв;
- Інтерактивний лендінг із сучасними анімаціями та зручною навігацією;
- Можливість реєстрації та авторизації через модальні діалоги;
- Персоналізація зовнішнього вигляду онлайн-меню для кожного ресторану;
- Генерація унікальних сторінок меню за допомогою субдоменів;
- Інструменти для фільтрації, пошуку та швидкої навігації по стравах;
- Миттєве оновлення даних без перезавантаження сторінки.

Для досягнення результату, визначеного у технічному завданні, та організації ефективного робочого процесу було використано сучасні інструменти керування проектом, що дозволило розподілити ролі, відповідальність та контролювати строки виконання задач.

Були прийняті архітектурні рішення, які дозволили власникам закладів харчування створювати власні онлайн-меню з мінімальними зусиллями:

- SPA-архітектура для забезпечення швидкої взаємодії з інтерфейсом;
- Розділення застосунків на лендинг/адмін-панель та клієнтську частину меню;
- Використання субдоменів для ідентифікації кожного ресторану та підвантаження відповідних налаштувань і тем оформлення;
- Централізоване керування станом та темою меню через реактивні сервіси.

Враховано можливості для масштабування та подальшого розвитку:

- Винесення великих закладів на окремі вузли;
- Перехід на SSR для покращення SEO та швидкості завантаження;
- Додавання багатомовності, розширення функціоналу для адміністраторів, інтеграція з зовнішніми сервісами.

Спроековано, реалізовано та випробувано фронтенд-частину інформаційної системи управління контентом закладів харчування. Система розгорнута на реальному домені та протестована як членами команди, так і добровольцями. Результати випробування підтвердили загальну працездатність і зручність створеної системи.

Перспективи подальших досліджень полягають у впровадженні SSR, розгляд переносу архітектури на gRPC для оптимізації трафіку та синхронізації контрактів розширенні можливостей кастомізації, підвищенні доступності інтерфейсу, а також інтеграції з новими сервісами для підвищення цінності платформи для кінцевих користувачів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

Використані джерела

1. Angular Documentation – офіційна документація по Angular. <https://angular.io/docs>
2. PrimeNG Documentation – документація по бібліотеці UI-компонентів для Angular PrimeNG. <https://primeng.org>
3. Tailwind CSS Docs – документація по утилітарній CSS-бібліотеці. <https://tailwindcss.com/docs>
4. ngx-lottie Documentation – документація по інтеграції Lottie-анімацій в Angular. <https://github.com/ngx-lottie/ngx-lottie>
5. Iconify Documentation – документація по бібліотеці іконок. <https://docs.iconify.design/>
6. TypeScript Handbook – офіційна документація по мові TypeScript. <https://www.typescriptlang.org/docs>
7. GitHub Docs – офіційна документація по GitHub Flow. <https://docs.github.com/en/get-started/using-github/github-flow>
8. Microsoft Learn – офіційна документація до інструментів Visual Studio Code, TypeScript та інших технологій. <https://learn.microsoft.com>

Використані ресурси

1. ChatGPT – асистент для аналізу рішень, оптимізації архітектури, створення описів та написання частини коду. <https://chat.openai.com>
2. Figma – інструмент для дизайну інтерфейсів, створення макетів клієнтської частини. <https://www.figma.com>
3. Visual Studio Code – середовище розробки, у якому здійснювалося написання, тестування та дебаг клієнтської частини. <https://code.visualstudio.com>
4. GitHub – система керування версіями коду, використана для зберігання, спільної роботи та розгортання клієнтської частини. <https://github.com>
5. angularx-qrcode – генерація QR-кодів в Angular-застосунках. <https://github.com/Cordobo/angularx-qrcode>
6. dom-to-image – бібліотека для експорту DOM-елементів у форматах PNG та SVG. <https://github.com/tsayen/dom-to-image>
7. ChoiceQR – сервіс генерації QR-кодів, аналізувався як референс для реалізації функціональності налаштування зовнішнього вигляду QR. <https://choiceqr.com>
