



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

Випускна кваліфікаційна робота бакалавра

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

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

Виконавці проекту:

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

Автор звіту:

_____ Ілечко Сергій Сергійович

Одеса – 2025

АНОТАЦІЯ

УДК 004.9

I-43

Ілечко С. С. 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 - 26с.

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:

- Авторизація
- Автентифікація
- Реєстрація
- Створення ієрархії онлайн меню
- Створення страв
- Зберігання загальних налаштувань онлайн меню
- Налаштування та генерація QR коду з посиланням на онлайн меню
- Редагування та зберігання налаштувань зовнішнього вигляду онлайн меню
- Керування зображеннями страв та іконок
- Додавання коментарів до медіафайлів
- Фільтрація страв за певними критеріями

При складанні звіту використано 6 посилань на друковані та електронні джерела інформації.

ЗМІСТ

ВСТУП	5
Технічне завдання до проєкту	6
РОЗДІЛ 1. Загальна характеристика системи	11
РОЗДІЛ 2. Реалізація клієнтської частини	13
РОЗДІЛ 3. Результати випробувань	20
ВИСНОВКИ	25
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ	26

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій цифрові рішення дедалі активніше впроваджуються в різні сфери діяльності, зокрема в індустрію громадського харчування. Використання цифрових інструментів дозволяє закладам не лише оптимізувати внутрішні процеси, а й забезпечити більш зручну та сучасну взаємодію з клієнтами.

Одним із актуальних напрямів цифровізації є впровадження онлайн-меню. На відміну від традиційних паперових варіантів, цифрові меню дають змогу оперативно оновлювати інформацію про страви, ціни та акційні пропозиції, а також адаптувати оформлення під фірмовий стиль закладу. Особливої популярності набули QR-меню - зручний інструмент доступу до інформації як для відвідувачів, так і для персоналу.

У відповідь на ці потреби розроблено веб-застосунок, клієнтська частина якого реалізована за допомогою Angular, що виступає CMS-платформою для керування цифровим меню.

Дана робота присвячений розробці клієнтській CMS частині проєкту та містить опис загальної характеристики системи, реалізацію клієнтської частини та результати випробувань. У фінальному розділі подано висновки щодо функціональності та ефективності запропонованого рішення в контексті підвищення зручності обслуговування клієнтів і цифрової трансформації закладів харчування.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЄКТУ

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 керуванням 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.

Адмін 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 , 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:

- Москва – 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 проєкту). Структура файлів та каталогів обирається на розсуд 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

РОЗДІЛ 1

ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

Розроблена система - це веб-платформа для цифрового представлення та управління меню закладів громадського харчування. Центральним елементом платформи є клієнтська CMS-частина, яка надає адміністраторам закладів інструменти для створення, редагування та кастомізації онлайн-меню без потреби в технічних знаннях. Система функціонує в моделі SaaS (Software as a Service).

Згідно з вимогами технічного завдання, проєкт реалізовано за принципами клієнт-серверної архітектури з чітким розмежуванням відповідальностей між частинами системи. Загалом виділяються такі основні компоненти:

- **Frontend (CMS/лендинг)** - інтерфейс для адміністраторів, що забезпечує керування контентом, структурування меню, стилізацію інтерфейсу та перегляд аналітики.
- **Frontend (онлайн-меню)** - публічна частина, доступна відвідувачам, яка динамічно відображає меню закладу.
- **Backend (CMS)** - обробка запитів з CMS-інтерфейсу, збереження даних у БД, керування автентифікацією та налаштуваннями меню.
- **Backend (меню)** - надання відвідувачам доступу до публічної частини меню без надлишкових даних.

Причини для розділення Frontend:

- Вимоги до стилістичної незалежності між CMS/лендингом та меню.
- Можливість використання різних бібліотек для оптимізації розміру файлів.
- Оптимізація сайтів ресторану, зокрема компіляція розмітки та стилів для зменшення обсягу даних.

Причини для розділення Backend:

- Відокремлення логіки безпеки: CMS працює з конфіденційними даними, а публічне меню лише з відображенням контенту.
- Потенційна можливість керувати декількома ресторанами одним користувачем

Технологічна основа

Клієнтська частина CMS створена з використанням Angular як основного фреймворку для побудови застосунку. Для стилізації інтерфейсу використовуються Tailwind CSS, PrimeNG для UI-компонентів і Iconify для системи іконок,

Передача даних між клієнтською та серверною частинами здійснюється через REST API. Усі дані зберігаються у централізованій базі даних із забезпеченням прав доступу на рівні закладів та користувачів.

Кастомізація і оформлення

Система надає можливість кожному закладу створити власне цифрове меню з унікальним оформленням, що відповідає стилю бренду. Після реєстрації користувач може формувати структуру меню, додавати нові страви, редагувати їхню інформацію, прикріплювати зображення та опис.

Особливістю системи є підтримка широких можливостей візуальної кастомізації. Кожен ресторан може змінювати кольорову палітру, вибирати шрифти, теми та інші елементи дизайну. Усі налаштування зберігаються окремо для кожного користувача та автоматично застосовуються при перегляді меню з боку клієнтів.

Контроль версій та організація розробки

Усі вихідні коди зберігаються у приватних репозиторіях GitHub, окремо для клієнтської та серверної частин: RestaurantSaasBackend та RestaurantSaasFrontend.

Для організації роботи в команді використовується стратегія GitHub Flow із обов'язковим створенням Pull Request, перевіркою коду та документацією змін.

Отже, розроблена система надає закладам харчування можливість швидко створювати і редагувати онлайн-меню, адаптуючи його під власний візуальний стиль, що сприяє підвищенню зручності для відвідувачів та покращенню загального користувацького досвіду.

РОЗДІЛ 2

РЕАЛІЗАЦІЯ КЛІЄНТСЬКОЇ ЧАСТИНИ

Клієнтська частина інформаційної системи управління контентом (CMS) для цифрових меню закладів харчування розроблена з використанням фреймворку Angular, компонентної бібліотеки PrimeNG та утиліти Tailwind CSS для швидкого стилізування інтерфейсів.

Інтерфейс реалізовано із застосуванням готових компонентів PrimeNG, які кастомізувалися за допомогою Tailwind CSS для відповідності загальному стилю проєкту та айдентиці. Використання Tailwind дало можливість гнучко налаштовувати відступи, кольори, типографіку та адаптивність без складних переписувань CSS.

Для роботи з іконками у проєкті використовується бібліотека Iconify, яка забезпечує зручний доступ до понад 200 000 оптимізованих SVG-іконок із сотень популярних наборів. Іконки підключаються автоматично через спеціальний тег `<iconify-icon>`, завантажуються тільки коли потрібні і зберігаються та кешуються у браузері. Це дозволяє уникнути зайвого навантаження на основний бандл і значно пришвидшує розробку.

Нижче наведено приклад використання іконки круасана з бібліотеки Iconify у шаблоні Angular:

```
<iconify-icon icon="mdi:croissant" class="text-yellow-600 w-6 h-6"></iconify-icon>
```

Для моніторингу показників продуктивності (Web Vitals) застосовувалися вбудовані інструменти браузерів (Chrome DevTools), а також зовнішні утиліти аналізу бандлів, зокрема <https://esbuild.github.io/analyze/>. Початковий розмір основного бандлу становив близько 500-600 КБ, що було зменшено за рахунок використання lazy loading та покращення роботи Angular Change Detection з акцентом на використання стратегії OnPush та нових реактивних сигналів для мемоізації результатів.

У межах реалізації клієнтської частини було впроваджено функціональність попереднього перегляду зовнішнього вигляду онлайн-меню ресторану у форматі iframe з можливістю миттєвої кастомізації. Це дозволяє адміністраторам у

реальному часі змінювати стилі (наприклад, кольори, фони, типографіку) та одразу побачити результат на вбудованому сайті, не перезавантажуючи сторінку.

Для забезпечення двосторонньої взаємодії між CMS та iframe використано механізм `postMessage` - стандартний інтерфейс браузера для безпечної передачі повідомлень між вікнами. CMS надсилає повідомлення у форматі `назва змінної:значення` (наприклад: `--color-primary-500:#1A73E8`) у контекст сайту ресторану, який відкрито в `<iframe>`.

На стороні онлайн-меню реалізовано слухач події `message`, який парсить вхідні дані та динамічно встановлює CSS-змінні у кореновому документі (`document.documentElement.style.setProperty(...)`), що дозволяє миттєво змінювати вигляд сайту без повного рендерингу. Приклад надсилання змінної з CMS:

```
setCssVariable(name: ThemeVariableName, value: string) {  
  window.frames[0].postMessage(`${name}:${value}`, this.#accountService.restaurantUrl());  
}
```

Цей підхід дозволив реалізувати живу та інтуїтивно зрозумілу кастомізацію сайту ресторану з попереднім переглядом для трьох типів пристроїв - десктопів, планшетів та мобільних телефонів. Перемикання між цими режимами реалізовано через зміну ширини `iframe` відповідно до обраного форм-фактору.

```
readonly #iframeWidths: Record<IframeModes, number> = {  
  desktop: 1024,  
  tablet: 768,  
  mobile: 320,  
};
```

Кожна зміна кольору в редакторі кастомізації генерує палітру відтінків за допомогою утиліт бібліотеки PrimeNG (`getPalette`) та передається в `iframe` через `postMessage`.

Комунікація з серверною частиною здійснюється за допомогою REST API-запитів. Для цього використовуються вбудовані в Angular механізми HttpClientModule.

Для забезпечення безпеки даних користувача та доступу до функцій CMS, після успішної авторизації користувач отримує куки токен, який зберігається на клієнтській стороні і використовується для автентифікації всіх подальших запитів до захищених ресурсів API.

Типовий підхід до організації взаємодії із Backend-частиною в проєкті - це використання окремих Angular сервісів. Кожен такий сервіс відповідає за HTTP-запити до певного розділу API.

У якості прикладу, приведено сервіс ProductService, який відповідає за всі операції, пов'язані з окремими стравами в меню. Цей сервіс надає функціонал для отримання інформації про страву, а також для її створення, оновлення та видалення.

```
export class ProductService {
  readonly #http = inject(HttpClient);
  readonly #apiUrl = environment.apiUrl + '/products';

  get(id: number) {
    return this.#http.get<Product>(`${this.#apiUrl}/${id}`, { withCredentials: true });
  }

  create(product: NewProduct) {
    // Використання toFormData для коректної відправки файлів разом з іншими даними
    return this.#http.post<Product>(this.#apiUrl, toFormData(product), { withCredentials: true });
  }

  update(updatedProduct: UpdatedProduct) {
    return this.#http.put<void>(this.#apiUrl, toFormData(updatedProduct), { withCredentials: true });
  }

  delete(id: number) {
    return this.#http.delete(`${this.#apiUrl}/${id}`, { withCredentials: true });
  }
}
```

Усі запити до API виконуються з параметром { withCredentials: true }, що забезпечує коректну передачу токена у заголовках Authorization разом із запитом. Це є важливим для належної аутентифікації та авторизації користувача на серверній стороні та доступу до захищених ресурсів.

Для зменшення кількості запитів до сервера та прискорення відкриття сторінки продукту дані про продукт передаються через стан маршруту (route state), якщо вони вже є з попередньої сторінки. Це дозволяє миттєво відобразити інформацію без очікування відповіді від API.

Наприклад, У CategoriesPageComponent при кліку на пункт “редагувати” в контекстному меню страви, система виконує навігацію з передачею об’єкта Product через state:

```
goToEditProductPage() {
  const product = this.selectedProduct();
  if (!product) {
    return;
  }

  this.#router.navigate(['/admin', 'menu', 'edit-product', product.id], {
    state: { product },
  });
}
```

На стороні компонента редагування страви (AddEditProductPageComponent), ці дані приймаються у конструкторі через Router.getCurrentNavigation().extras.state. Якщо об’єкт product присутній, система ініціалізує сторінку одразу, без виконання додаткового HTTP-запиту до серверу:

```
#tryInitEditPage(): boolean {
  const product = this.#router.getCurrentNavigation()?.extras.state?.['product'] as Product;
  if (product) {
    this.#initEditPageWithProduct(product);
    return true;
  }
  ...
}
```

Однак, якщо сторінка редагування відкривається напряму (наприклад, через оновлення сторінки або перехід за прямим посиланням), тоді в route.paramMap зчитується productId, і дані завантажуються з сервера:

```
const productId = Number.parseInt(this.#route.snapshot.paramMap.get('product') ?? "");
```

Завдяки цьому реалізовано гібридний підхід до завантаження даних: якщо інформація про продукт уже є, вона передається через route state і сторінка відкривається миттєво. В іншому випадку дані підвантажуються з API за ідентифікатором продукту. Такий підхід дозволяє забезпечити швидку навігацію та підтримку прямих переходів за посиланням.

Для забезпечення коректності введення даних у формах реалізовано систему валідації, яка поєднує використання вбудованих валідаторів Angular, кастомних валідаторів, а також UI-директиву для відображення помилок. Це дозволяє гарантувати достовірність введених значень і підвищити зручність роботи користувача з інтерфейсом.

Наприклад, у компоненті `AddEditCategoryComponent`, який відповідає за створення або редагування категорій у меню, форма створюється з допомогою `FormGroup` з набором валідаторів:

```
readonly categoryForm = new FormGroup({
  name: new FormControl("", [Validators.required, CustomValidators.noEmptyOrWhiteSpace]),
  description: new FormControl<string | undefined>(undefined),
  isVisible: new FormControl<boolean>(true),
});
```

Поле `name` є обов'язковим і додатково перевіряється за допомогою кастомного валідатора `noEmptyOrWhiteSpace`, який не допускає порожніх або пробільних значень:

```
export class CustomValidators {
  static noEmptyOrWhiteSpace(control: AbstractControl): ValidationErrors | null {
    return (control.value || "").trim().length ? null : { required: true };
  }
}
```

Для того, щоб повідомлення про помилки виводились автоматично, застосовано власну директиву `ControlValidatorDirective`. Вона динамічно реагує на зміну статусу поля (`statusChanges`) та додає до DOM відповідні повідомлення про помилки. Нижче наведено приклад використання директиви у шаблоні:

Приклад використання директиви у шаблоні:

```
<input
  type="text"
  placeholder="Назва категорії"
  controlValidator
  pInputText
  formControlName="name"
/>
```

Перед відправленням даних на сервер форма перевіряється через функцію `verifyForm`, яка перевіряє валідність усіх полів:

```
onCreate() {  
  if (!verifyForm(this.categoryForm)) {  
    return;  
  }  
  // надсилання даних до сервера  
}
```

Таким чином, у проєкті реалізовано уніфікований підхід до валідації форм, який поєднує Angular-валідатори, власні перевірки та кастомну директиву для відображення помилок. Це забезпечує єдиний стиль валідації, зменшує кількість помилок і підвищує зручність користування інтерфейсом.

У проєкті реалізовано сторінку для генерації та налаштування QR-кодів, що дає змогу створювати унікальні коди з посиланням на сайт закладу. Для цього використовується бібліотека `angularx-qrcode`, яка відповідає за відображення коду у вигляді SVG або canvas.

Приклад генерації QR-коду:

```
<qrcode  
  [elementType]="logoPreview !== null && useLogo ? 'canvas' : elementType"  
  [qrcode]="customerSiteUrl()"  
  [colorDark]="foregroundColor"  
  [colorLight]="isTransparent ? '#FFFFFF' : backgroundColor"  
  [width]="qrScale"  
  [margin]="0"  
  [imageSrc]="logoSrc"  
  [imageHeight]="qrScale / 4"  
  [imageWidth]="qrScale / 4">  
</qrcode>
```

Тут параметри `colorDark` та `colorLight` визначають кольори коду, а `imageSrc` - шлях до логотипу, який можна вставити у центр QR-коду. Для кастомізації користувач може вибирати: Колір переднього плану і фону за допомогою `p-colorPicker`. Включати або виключати прозорий фон. Додавати текст зверху та знизу QR-коду з можливістю вибору шрифту, розміру і кольору. Вмикати або вимикати рамку з регульованим кольором і скругленням.

Для збереження результату застосовується бібліотека `dom-to-image`, яка конвертує блок з QR-кодом у файл формату PNG або SVG та запускає завантаження:

```
saveBlock() {  
  const element = this.qrBlock.nativeElement;  
  if (this.selectedFormat === 'PNG') {  
    domtoimage.toPng(element).then(dataUrl => this.downloadDataUrl(dataUrl, 'qr-block.png'));  
  } else if (this.selectedFormat === 'SVG') {  
    domtoimage.toSvg(element).then(dataUrl => this.downloadDataUrl(dataUrl, 'qr-block.svg'));  
  }  
}  
  
private downloadDataUrl(dataUrl: string, filename: string) {  
  const a = document.createElement('a');  
  a.href = dataUrl;  
  a.download = filename;  
  a.click();  
}
```

Таким чином, реалізація забезпечує інтерфейс для створення брендированих QR-кодів із широкими можливостями кастомізації.

РОЗДІЛ 3

РЕЗУЛЬТАТИ ВИПРОБУВАНЬ

Результатом компіляції програмного коду клієнтської частини CMS є готовий веб-застосунок, який надає користувачам інтерфейс для керування інформацією ресторану, меню та дизайном сайту.

Першим кроком користувача є **реєстрація в системі**. Користувачу потрібно ввести такі поля як: назва ресторану, логін, телефон та адресу електронної пошти. Якщо користувач вже зареєстрований, він може увійти до системи, використовуючи вказані під час реєстрації облікові дані.

The image shows two side-by-side web forms. The left form is titled "Створити обліковий запис" (Create account) and the right form is titled "Вхід в обліковий запис" (Login). Both forms have fields for restaurant name, login/email, password, and password confirmation. The right form also has a "Увійти" (Login) button.

Створити обліковий запис
Вже зареєстровані? [Увійти](#)

Назва ресторану
Логін
Телефон
Адреса електронної пошти
Це поле є обов'язковим
Пароль
Це поле є обов'язковим
Підтвердіть пароль
Я приймаю умови використання та прочитав [правила конфіденційності](#)

Вхід в обліковий запис
Ще не зареєстровані? [Створити обліковий запис](#)

Назва ресторану
Адреса електронної пошти або логін
Це поле є обов'язковим
Пароль
Це поле є обов'язковим
[Увійти](#)

Рисунок 1.1 - Реєстрація користувача та вхід

Після авторизації користувач отримує доступ до основних можливостей системи.

Однією з важливих функцій системи є **налаштування QR-коду**, що містить посилання на сайт ресторану. Користувач має можливість налаштувати: розмір QR-коду, кольори переднього плану та фону, додати брендований логотип, вибрати текстові підписи з кастомними шрифтами, розмірами і кольорами, обрати стиль рамки з різними варіантами скруглення і кольорами.

Для збереження QR-коду передбачено експорт у форматах PNG або SVG. Завдяки цьому, власники закладів можуть використовувати брендований QR-код як у друкованих матеріалах, так і в онлайн-рекламі.

QR

Обрати колір
#000000

Фон Прозорий фон

Обрати колір
#ffffff

Текст

Шрифт
Roboto

Обрати колір шрифту
#000000

☒ Текст 1
Проскануй мене! 16 px

☒ Текст 2
Проскануй мене! 32 px

☒ Рамка

Скруглення

Попередній перегляд

Проскануй мене!

Проскануй мене

Висота та ширина, px
128

Виберіть, у якому розмірі ви хочете завантажити налаштований QR-код

Формат файлу
☒ PNG ☐ SVG

Завантажити

Рисунок 1.2 - Сторінка налаштування QR-коду

У розділі **Налаштування** користувач може вказати основну інформацію про заклад, що буде відображатись на сайті та у меню:

- Назва ресторану, його опис.
- Країна, мова інтерфейсу та основна валюта,
- Адреса, електронна пошта та телефонні номери для контакту,
- Посилання на соціальні мережі (YouTube, Telegram, Instagram, Facebook, TikTok),
- Графік роботи закладу з можливістю вказати час роботи по днях тижня.

Налаштування Скасувати Зберегти

Назва та опис
Вкажіть публічну інформацію про свій бізнес, яка слугуватиме основою для опису вашого сайту в пошуку

Назва
Як ваш ресторан називається?

Опис
0/500
Опишіть ваш заклад

Загальна інформація
Загальна інформація про ресторан. Відображатиметься на сайті та може використовуватися в меню. Заповніть усі поля, щоб користування було легшим для ваших клієнтів.

Країна
Україна

Мова
Українська

Основна валюта
UAH - Гривня

Адреса
Введіть адресу

Електронна адреса
poshta@gmail.com

Номер телефону
+380964824178
+380661945126

Рисунок 1.3 - Сторінка налаштувань

Сторінка **Зовнішній вигляд** надає інструменти для візуального кастомізування сайту. Користувач може редагувати кольори:

- Основного фону,
- Другорядного фону,
- Брендowego кольору,
- Контрастного кольору,
- Кольору тексту,

А також обирати шрифти. Вбудований режим попереднього перегляду дозволяє бачити, як сайт виглядатиме на різних пристроях - мобільному телефоні, планшеті або десктопі.

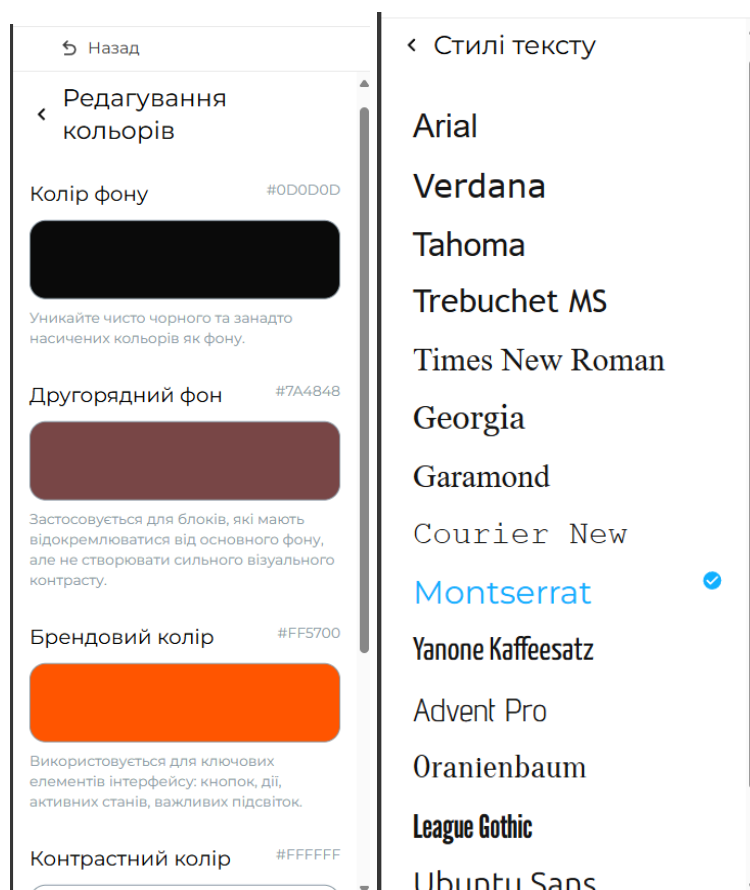


Рисунок 1.4 - Можливості кастомізації меню: редагування кольорів та стилі шрифту.

Розділ **Меню** дозволяє повністю керувати структурою закладу: створювати, редагувати та видаляти розділи меню, категорії у них, а також окремі страви.

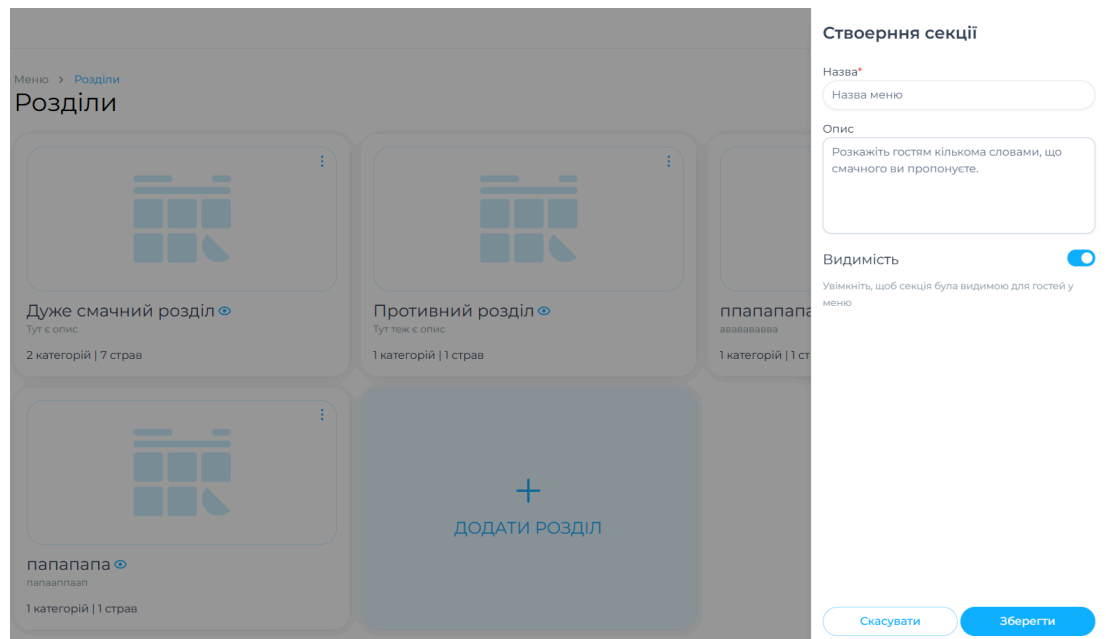


Рисунок 1.5 - Сторінка розділів з відкритим меню створення розділу.

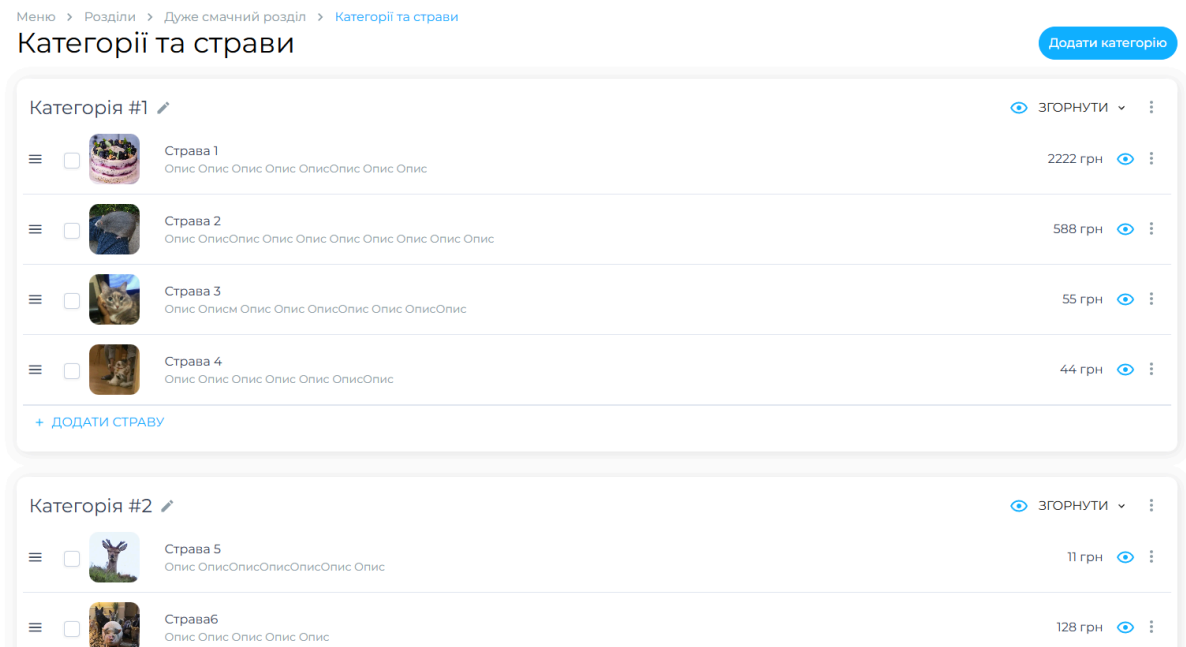


Рисунок 1.6 - Сторінка категорій.

На сторінці **Редагування страви** реалізовані такі поля:

- Назва страви,
- Опис страви,
- Вартість,
- Відсоток знижки,
- Вага/об'єм із вибором одиниці виміру,
- Вміст алкоголю,
- Харчова цінність (калорії, білки, жири, вуглеводи),
- Можливість завантаження до 4 фото,
- Час приготування у хвилинах.

Меню > ... > Категорія #1 > Категорії та страви

Створення страви

[Зберегти](#)

Основне

Назва страви*

Опис страви

Вартість

Вартість* Знижка %

Вага/об'єм страви

Вага/об'єм Вимір у

Вміст алкоголю %

Видимість ☒

Увімкніть, щоб страва була видимою для гостей у меню

Обрати фото



[Обкладинка](#)

Максимальна кількість фото для завантаження - 4
Максимальний розмір файлу — 2 MB
Дозволені лише: png, jpg, jpeg

Час приготування

Час приготування, хвилини

Рисунок 1.7 - Сторінка створення продукту.

Всі описані функції клієнтської CMS частини були реалізовані та протестовані, що дозволяє забезпечити ефективне керування сайтом ресторану, меню та QR-кодами.

ВИСНОВКИ

Проведено огляд цифрових рішень для закладів харчування, виявлено перелік функціональностей, який має найбільшу цінність:

- Наповнення меню стравами з детальною інформацією;
- Персоналізація зовнішнього вигляду сайту з переліком страв;
- Персоналізація та генерація QR коду з посиланням на онлайн меню;
- Інструменти для навігації та пошуку на сторінці зі стравами;
- Актуалізація даних у реальному часі.

Для ефективного досягнення цілей, визначених у технічному завданні, а також для забезпечення структурованої та контрольованої організації робочих процесів, було використано систему Plane.so. Її впровадження дозволило чітко розподілити ролі та зони відповідальності учасників команди, а також здійснювати моніторинг дотримання встановлених термінів виконання завдань.

Проведено розробку та реалізацію клієнтської частини інформаційної системи управління вмістом сайту ресторану (CMS), що дозволяє власникам закладів харчування зручно керувати онлайн-меню та персоналізувати візуальне оформлення.

Систему розгорнуто, протестовано на реальному домені. В результаті тестування підтверджено функціональність реалізованих компонентів. Система складена за розподіленою клієнт-серверною архітектурою та складається з трьох частин, розміщених за адресами:

- Mockup: <https://www.figma.com/file/d9ivM3L0t63EbYJjonjuk2>
- Backend: <https://github.com/BeloMaximka/RestaurantSaasBackend>
- Frontend: <https://github.com/BeloMaximka/RestaurantSaasFrontend>

Перспективи подальшої роботи полягають у розширенні функціональності CMS: локалізація контенту, підтримка декількох мов, реалізація ролей користувачів, міграцію сайту ресторану на SSR-сумісну архітектуру, інтеграція з POS-системами.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

Використані джерела:

1. GitHub Docs – офіційна документація описує простий гілковий робочий процес *GitHub Flow*. <https://docs.github.com/en/get-started/using-github/github-flow>
2. Microsoft Learn – офіційна документація до інструментів Visual Studio Code, TypeScript та інших технологій. <https://learn.microsoft.com>.
3. Angular Documentation – офіційна документація по Angular. <https://angular.io/docs>
4. PrimeNG Documentation – документація по бібліотеці UI-компонентів для Angular [PrimeNG](https://www.primefaces.org/primeng/)
5. Tailwind CSS Docs – документація по утилітарній CSS-бібліотеці. <https://tailwindcss.com/docs>

Використані ресурси:

6. Chat GPT – використовувався як асистент для аналізу варіантів вирішення задач, оптимізації архітектурних рішень, створення описів, а також для написання частини коду. <https://chat.openai.com>
7. Choice QR – сервіс генерації QR-кодів, аналізувався як референс для реалізації функціональності налаштування зовнішнього вигляду QR. <https://choiceqr.com>
8. Figma – інструмент для дизайну інтерфейсів, використовувався для створення мокапів клієнтської частини. <https://www.figma.com>
9. Visual Studio Code – середовище розробки, у якому здійснювалося написання, тестування та дебаг клієнтської частини застосунку. <https://code.visualstudio.com>
10. GitHub – система керування версіями коду, використана для зберігання, спільної роботи та розгортання клієнтської частини. <https://github.com>
11. angularx-qrcode – генерація QR-кодів в Angular-застосунках. <https://github.com/Cordobo/angularx-qrcode>
12. dom-to-image – бібліотека для експорту DOM-елементів у форматах PNG та SVG. <https://github.com/tsayen/dom-to-image>
13. Tailwind CSS – утилітарна CSS-бібліотека для швидкої та адаптивної верстки. <https://tailwindcss.com/docs>
14. PrimeNG – бібліотека UI-компонентів для Angular, використана для реалізації інтерфейсу: кнопки, чекбокси, селектори, колірні підіймачі тощо. <https://primeng.org>
15. Iconify – бібліотека іконок. [Iconify Design: All popular icon sets, one framework.](https://iconify.design/)