

# Звіт по компонентах маршрутизації та сторінки деталей контейнера

## 1. Роутер (AppRouter.tsx)

**Призначення:** Компонент відповідає за конфігурацію всієї системи маршрутизації додатку, визначає доступність сторінок для різних категорій користувачів.

### Основна структура:

- Використовує React Router v6 для організації навігації
- Розділяє маршрути на публічні та захищені через компонент ProtectedRoute
- Підтримує українську локалізацію в коментарях

### Категорії маршрутів:

#### *Публічні маршрути:*

- / - головна сторінка (доступна всім)
- /listing/:id - перегляд оголошення (доступний всім)
- /container/:id/details - деталі контейнера (доступні всім)

#### *Маршрути автентифікації:*

- /login та /register - доступні лише неавторизованим користувачам
- Авторизовані користувачі автоматично перенаправляються на головну сторінку

#### *Захищені маршрути користувача:*

- /myaccount/\* - особистий кабінет з підрозділами:
  - listings/active - активні оголошення
  - listings/pending - оголошення в очікуванні
  - listings/rejected - відхилені оголошення
  - listings/deleted - видалені оголошення
- /create-listing - створення нового оголошення
- /listing/:id/edit - редагування оголошення

#### *Адміністративні маршрути:*

- /admin/\* - панель адміністратора з підрозділами:
  - listings - управління оголошеннями
  - parser - парсер даних
  - users - управління користувачами
  - logs - системні логи

### Особливості реалізації:

- Використовує хук `useAuth()` для перевірки стану авторизації
- Реалізована fallback-сторінка 404 для неіснуючих маршрутів
- Компонент розділено на `AppRouterContent` та `AppRouter` для можливого розширення функціоналу

## 2. Захищені роути (`ProtectedRoute.tsx`)

**Призначення:** Компонент-обгортка для контролю доступу до захищених сторінок на основі статусу авторизації та ролі користувача.

### Параметри компонента:

- `children` - дочірні компоненти для відображення
- `requiredRoles` - масив ролей, які мають доступ до маршруту
- `requireAuth` - прапорець необхідності авторизації (за замовчуванням `true`)

### Логіка перевірок:

#### Етап 1 - Завантаження:

```
if (isLoading) {  
  // Відображення спінера з текстом "Завантаження..."  
}
```

#### Етап 2 - Перевірка автентифікації:

```
if (requireAuth && !isAuthenticated) {  
  // Перенаправлення на /login зі збереженням  
  попереднього URL  
}
```

#### Етап 3 - Перевірка ролі:

```
if (requiredRoles.length > 0 &&  
    !requiredRoles.includes(user.role) && user.role !==  
    "admin") {
```

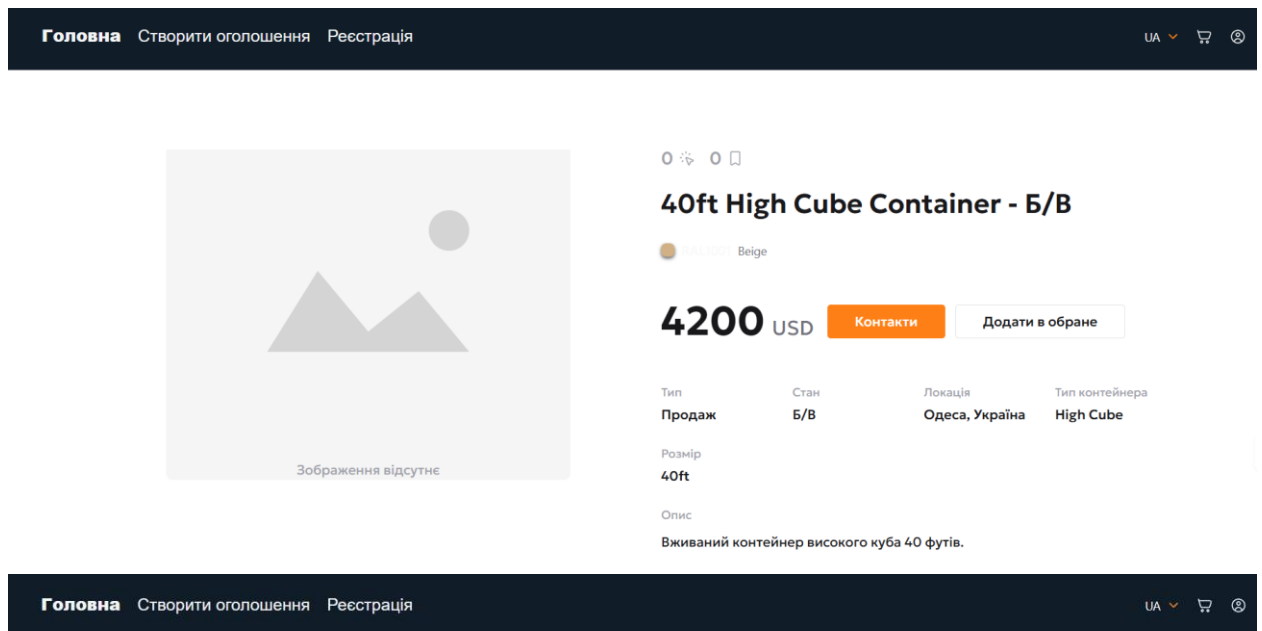
```
// Відображення повідомлення про відсутність доступу  
}
```

## UI елементи:

- Спіннер завантаження з українським текстом
- Повідомлення про відсутність доступу з кнопкою повернення назад
- Використання власних CSS класів замість Chakra UI

## 3. Сторінка деталей контейнера (ContainerDetailsPage.tsx)

**Призначення:** Відображення детальної інформації про конкретний контейнер з можливістю перегляду фотографій та рекомендованих аналогічних контейнерів.



*Отримання даних:*

```
const { data: allListingsData, isLoading, isFetching }
= useListings({
  page: 1,
  page_size: 20,
  status: 'active',
});
```

*Компоненти сторінки:*

1. **ImageCarousel** - карусель зображень контейнера
2. **ContainerDescription** - детальний опис характеристик
3. **RecommendedSection** - секція рекомендованих контейнерів (до 4 шт.)

**Стани завантаження:**

- Спіннер під час isLoading або isFetching
- Повідомлення про відсутність контейнера
- Адаптивний layout для різних розмірів екрану

**Responsive дизайн:**

```
<Flex
  direction={{ base: 'column', lg: 'row' }}
  px={{ base: '16px', md: '128px' }}
>
  <Box flex="1" maxW={{ lg: '600px' }}>
    <ImageCarousel />
  </Box>
  <Box flex="1" maxW={{ lg: '600px' }}>
    <ContainerDescription />
  </Box>
</Flex>
```

**Допоміжні компоненти**

**ImageCarousel.tsx:**

- Підтримка навігації між зображеннями
- Превью-мініатюри з підсвічуванням активного зображення
- Placeholder для відсутніх зображень
- Адаптивне позиціонування кнопок навігації
- Анімації при наведенні

**ContainerCard.tsx:**

- Компактне відображення інформації про контейнер
- Відображення основних характеристик:
  - Назва
  - Ціна з валютою
  - Тип (з використанням listingTypes map)
  - Стан (з використанням conditionMap)
  - RAL колір через компонент RalColorBox
- Вертикальний роздільник між картками
- Fallback зображення для відсутніх фото

#### **RecommendedSection.tsx:**

- Grid layout з адаптивною кількістю колонок (1-4)
- Сірий фон (#FAFAFA) для візуального відділення
- Умовний рендеринг (не показується якщо немає рекомендацій)

#### **Загальні особливості:**

- Консистентне використання Chakra UI
- Типізація через TypeScript інтерфейси
- Українська локалізація всіх текстів
- Єдиний дизайн-стиль з шрифтом 'Geologica Variable'
- Брендний колір акцентів #FD7F16

## **4. Перемикач мови (LanguageSwitcher.tsx)**

**Призначення:** Компонент для зміни мови інтерфейсу додатку з підтримкою української та англійської мов.

#### **Технологічний стек:**

- Використовує бібліотеку react-i18next для інтернаціоналізації
- Хук useTranslation() для доступу до об'єкта i18n
- Власні CSS стилі через окремий файл LanguageSwitcher.css

#### **Структура даних мов:**

```
const languages = [
  { code: 'ua', label: 'Українська', short: 'УКР' },
  { code: 'en', label: 'English', short: 'ENG' }
];
```

#### **Функціонал:**

### *Відображення поточної мови:*

- Використовується значення `i18n.language` для визначення активної мови
- В селекторі відображаються скорочені назви мов (UKR/ENG)

### *Зміна мови:*

```
const changeLanguage = (event:
React.ChangeEvent<HTMLSelectElement>) => {
    i18n.changeLanguage(event.target.value);
};
```

### **UI компоненти:**

- `<select>` елемент для вибору мови
- Атрибут `aria-label="Select language"` для доступності
- Кастомна стрілка ▼ для стилізації селектора
- Опції формуються динамічно через `map()` по масиву мов

### **Особливості реалізації:**

#### *Переваги:*

- Простий та зрозумілий інтерфейс
- Підтримка accessibility через `aria-label`
- Розширюваність - легко додати нові мови до масиву `languages`
- Відокремлені стилі в CSS файлі

#### *Потенційні вдосконалення:*

1. Код мови для української використовує `'ua'` замість стандартного ISO `'uk'`
2. Відсутня типізація для масиву мов (можна створити `interface Language`)
3. Немає збереження вибраної мови в `localStorage`
4. Використання HTML `select` замість кастомного `dropdown` (може бути обмеженням в стилізації)

### **Інтеграція з додатком:**

- Компонент зазвичай розміщується в `header` або `navigation bar`
- Зміна мови миттєво перекладає весь інтерфейс завдяки `react-i18next`
- Для коректної роботи потрібна попередня конфігурація `i18n` з файлами перекладів

### Приклад використання:

```
import LanguageSwitcher from
'./components/LanguageSwitcher';

function Header() {
  return (
    <header>
      <nav>
        { /* інші елементи навігації */ }
        <LanguageSwitcher />
      </nav>
    </header>
  );
}
```



