

## Zajęcia: Dodajemy Redux do aplikacji pogodowej

### Co mamy obecnie w projekcie

```
src/
  └── App.jsx           ← główny komponent z routingu
  └── main.jsx          ← punkt wejścia aplikacji
  └── Pages/
    ├── HomePage.jsx    ← lista miast + wyszukiwarka
    └── CityDetailPage.jsx  ← szczegóły pogody
  └── components/
    ├── WeatherCard.jsx   ← kartka miasta
    ├── weatherDetails.jsx  ← szczegóły pogody
    └── WeatherIcon.jsx    ← emoji pogody
  └── data/
    └── WeatherData.jsx  ← dane 5 miast
```

**Co już działa:** React Router, wyszukiwarka, hooki (useState, useEffect, useMemo, useCallback)

**Co dziś zrobimy:** 1. ✅ Redux → globalna zmiana jednostek temperatury (°C / °F / K) 2. ✅ Ulubione miasta (gwiazdka) 3. ✅ LocalStorage (żeby ustawienia się zapisywały po odświeżeniu)

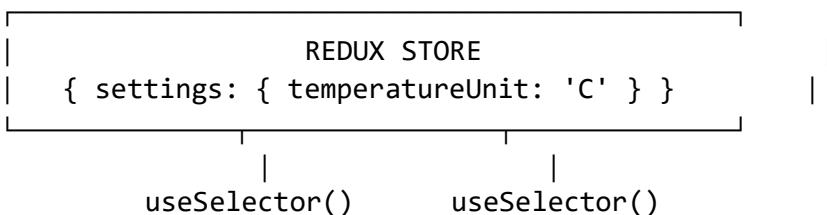
## CZĘŚĆ 1: Instalacja Redux (10 min)

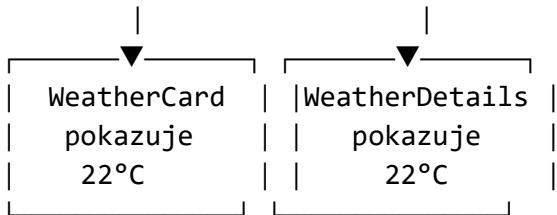
### Po co nam Redux?

Do tej pory stan trzymaliśmy w komponentach przez useState.

**Problem:** Jeśli chcemy mieć wspólny stan dla całej aplikacji (np. jednostkę temperatury), to musielibyśmy przekazywać propsy przez wiele poziomów komponentów. To się nazywa "prop drilling" i jest uciążliwe.

**Rozwiążanie:** Redux daje nam **jeden globalny stan**, do którego każdy komponent może sięgnąć bezpośrednio - bez przekazywania propsów.





## Instalacja

W terminalu, w katalogu projektu:

```
npm install @reduxjs/toolkit react-redux
```

**Co instalujemy:** - `@reduxjs/toolkit` - nowoczesny Redux z uproszczoną składnią - `react-redux` - hooki do łączenia Redux z React: - `useSelector` - odczytuje dane ze store - `useDispatch` - wysyła akcje do store

## CZĘŚĆ 2: Tworzymy Redux Store (25 min)

### Krok 1: Tworzymy strukturę folderów

Utwórzcie nowe foldery w `src/`:

```
mkdir -p src/store/slices
mkdir -p src/utils
```

Po tym struktura wygląda tak:

```
src/
  └── store/
      ├── index.js           ← NOWY FOLDER
      └── slices/
          └── settingsSlice.js ← główny store (za chwilę utworzymy)
  └── utils/
      └── temperature.js   ← NOWY FOLDER
  └── App.jsx
  └── main.jsx
  └── ... reszta plików
```

### Krok 2: Tworzymy slice ustawień

■ **NOWY PLIK: `src/store/slices/settingsSlice.js`**

**Slice** to “kawałek” stanu aplikacji wraz z funkcjami do jego zmiany.

```
// =====
// PLIK: src/store/slices/settingsSlice.js
// OPIS: Slice odpowiedzialny za ustawienia aplikacji (jednostki temperatury)
// =====

import { createSlice } from '@reduxjs/toolkit';
// ↑ createSlice to funkcja z Redux Toolkit, która tworzy slice

// =====

// STAN POCZĄTKOWY
// To jest stan który będzie na starcie aplikacji
// =====

const initialState = {
    temperatureUnit: 'C', // Możliwe wartości: 'C', 'F', 'K'
};

// =====

// TWORZENIE SLICE'A
// =====

const settingsSlice = createSlice({
    name: 'settings', // Nazwa slice'a (widoczna w Redux DevTools)
    initialState, // Stan początkowy (zdefiniowany wyżej)
    reducers: {
        // =====
        // REDUCER: setTemperatureUnit
        // Co robi: Zmienia jednostkę temperatury
        // Kiedy wywoływany: Gdy użytkownik wybierze inną jednostkę w select
        // =====
        setTemperatureUnit: (state, action) => {
            // state = aktualny stan (np. { temperatureUnit: 'C' })
            // action.payload = nowa wartość przekazana przy wywołaniu (np. 'F')
            state.temperatureUnit = action.payload;
        },
    },
});

// =====

// EKSPORTY
// =====

// Eksportujemy AKCJE - użyjemy jej z dispatch() w komponentach
// Przykład użycia: dispatch(setTemperatureUnit('F'))
export const { setTemperatureUnit } = settingsSlice.actions;
```

```
// Eksportujemy REDUCER - potrzebny do konfiguracji store
export default settingsSlice.reducer;
```

### Krok 3: Tworzymy główny store

#### ■ NOWY PLIK: src/store/index.js

```
// _____
// PLIK: src/store/index.js
// OPIS: Główny store Redux - "magazyn" całego stanu aplikacji
// _____

import { configureStore } from '@reduxjs/toolkit';
// ↑ configureStore tworzy store z dobrymi domyślnymi ustawieniami

import settingsReducer from './slices/settingsSlice';
// ↑ Importujemy reducer z naszego slice'a

// _____
// KONFIGURACJA STORE
// _____

const store = configureStore({
  reducer: {
    // Klucz 'settings' = jak będziemy się odwoływać do tego stanu
    // Wartość = reducer który obsługuje ten fragment stanu
    settings: settingsReducer,
    // Później dodamy tutaj:
    // favorites: favoritesReducer,
  },
});
// Stan w STORE będzie wyglądał tak:
// {
//   settings: {
//     temperatureUnit: 'C'
//   }
// }

export default store;
```

## Krok 4: Podłączamy store do aplikacji

### ■ MODYFIKUJEMY: src/main.jsx

```
// =====  
// PLIK: src/main.jsx  
// ZMIANY: Dodajemy Provider z Redux  
// =====  
  
import { StrictMode } from 'react'  
import { createRoot } from 'react-dom/client'  
import { Provider } from 'react-redux' // ← NOWE: Provider z react-redux  
import store from './store' // ← NOWE: nasz store  
import './index.css'  
import App from './App.jsx'  
  
createRoot(document.getElementById('root')).render(  
  <StrictMode>  
    {/* ↓ NOWE: Opakowujemy aplikację w Provider */}  
    {/* Provider udostępnia store wszystkim komponentom w środku */}  
    <Provider store={store}>  
      <App />  
    </Provider>  
  </StrictMode>,  
)
```

Co się zmieniło:

| Linia  | Było    | Jest   |
|--------|---------|--|
| import | -       | import { Provider }<br>from 'react-redux'        |
| import | -       | import store from<br>'./store'                   |
| render | <App /> | <Provider<br>store={store}><App<br>/></Provider> |

## CZĘŚĆ 3: Używamy Redux w komponentach (25 min)

### Krok 1: Funkcje przeliczające temperaturę

### ■ NOWY PLIK: src/utils/temperature.js

```
// =====  
// PLIK: src/utils/temperature.js  
// OPIS: Funkcje pomocnicze do przeliczania temperatury  
// =====
```

```

// -----
// FUNKCJA: convertTemperature
// Co robi: Przelicza temperaturę z Celsjusza na wybraną jednostkę
// Parametry:
//   - celsius: liczba (temperatura w °C)
//   - unit: string ('C', 'F' lub 'K')
// Zwraca: liczbę (przeliczona temperatura)
// -----
export function convertTemperature(celsius, unit) {
  switch (unit) {
    case 'F':
      // Wzór: °F = °C × 9/5 + 32
      return Math.round((celsius * 9/5) + 32);
    case 'K':
      // Wzór: K = °C + 273.15
      return Math.round(celsius + 273.15);
    case 'C':
    default:
      // Celsjusz - zwracamy bez zmian
      return celsius;
  }
}

// -----
// FUNKCJA: getUnitSymbol
// Co robi: Zwraca symbol jednostki do wyświetlenia
// Parametry:
//   - unit: string ('C', 'F' lub 'K')
// Zwraca: string (np. °C, °F, K)
// -----
export function getUnitSymbol(unit) {
  switch (unit) {
    case 'F': return '°F';
    case 'K': return 'K';
    case 'C':
    default: return '°C';
  }
}

```

## Krok 2: Komponent przełącznika jednostek

### NOWY PLIK: src/components/UnitSwitcher.jsx

```

// -----
// PLIK: src/components/UnitSwitcher.jsx
// OPIS: Komponent z selectem do wyboru jednostki temperatury
// -----

```

```
import { useSelector, useDispatch } from 'react-redux';
// ↑ useSelector - odczytuje dane ze store
// ↑ useDispatch - zwraca funkcję do wysyłania akcji

import { setTemperatureUnit } from '../store/slices/settingsSlice';
// ↑ Importujemy akcję którą będziemy wysyłać

function UnitSwitcher() {
  //

  // ODCZYT ZE STORE
  // useSelector przyjmuje funkcję która wybiera kawałek stanu
  // state.settings.temperatureUnit = 'C' (lub 'F' lub 'K')
  //

  const currentUnit = useSelector((state) => state.settings.temperatureUnit);

  //

  // DISPATCH - funkcja do wysyłania akcji
  //

  const dispatch = useDispatch();

  //

  // HANDLER - wywoływany gdy użytkownik zmieni select
  //

  const handleChange = (e) => {
    // e.target.value = nowa wartość wybrana w select (np. 'F')
    // dispatch() wysyła akcję do store
    // setTemperatureUnit('F') tworzy akcję: { type: 'settings/setTemperatureUnit', payload: 'F' }
    dispatch(setTemperatureUnit(e.target.value));
  };

  //

  // RENDER
  //

  return (
    <div className="unit-switcher">
      <label>Jednostka temperatury: </label>
      <select value={currentUnit} onChange={handleChange}>
        <option value="C">Celsjusz (°C)</option>
        <option value="F">Fahrenheit (°F)</option>
        <option value="K">Kelvin (K)</option>
    
```

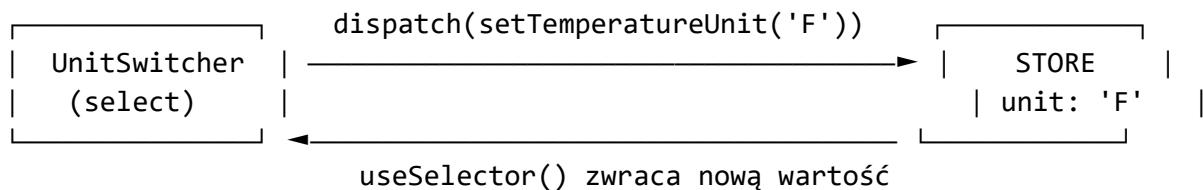
```

        </select>
    </div>
);
}

export default UnitSwitcher;

```

### Schemat działania:



## Krok 3: Dodajemy UnitSwitcher do HomePage

### ■ MODYFIKUJEMY: src/Pages/HomePage.jsx

#### PRZED (początek pliku):

```

import { useState, useMemo, useCallback } from 'react'
import WeatherCard from '../components/WeatherCard'
import WeatherDetails from '../components/WeatherDetails'
import { useNavigate } from 'react-router-dom';

```

#### PO (początek pliku):

```

import { useState, useMemo, useCallback } from 'react'
import WeatherCard from '../components/WeatherCard'
import WeatherDetails from '../components/WeatherDetails'
import { useNavigate } from 'react-router-dom';
import UnitSwitcher from '../components/UnitSwitcher'; // ← NOWE

```

#### PRZED (fragment renderowania):

```

return (
<>
<h1>Pogoda w Polsce</h1>

<div>
<input
  type ="text"
  placeholder="Szukaj miasta..."
  value={searchTerm}
  onChange={(e) => setSearchTerm(e.target.value)}>

```

```
    />
  </div>
```

### PO (fragment renderowania):

```
return (
  <>
  <h1>Pogoda w Polsce</h1>

  {/* ↓↓ NOWE: Dodajemy przełącznik jednostek ↓↓ */}
  <UnitSwitcher />
  {/* ↑↑ KONIEC NOWEGO ↑↑ */}

  <div>
    <input
      type ="text"
      placeholder="Szukaj miasta..."
      value={searchTerm}
      onChange={(e) => setSearchTerm(e.target.value)}
    />
  </div>
```

### Podsumowanie zmian w HomePage.jsx:

| Miejsce              | Co dodajemy   |
|----------------------|---|
| Importy (góra pliku) | <code>import UnitSwitcher from<br/>'../components/UnitSwitcher';</code> |
| W renderze, pod <h1> | <code>&lt;UnitSwitcher /&gt;</code>                                     |

### Krok 4: Aktualizujemy WeatherCard

#### ■ MODYFIKUJEMY: src/components/WeatherCard.jsx

##### PRZED (cały plik):

```
function WeatherCard(props){
  const className = `city-card ${props.selected ? 'selected' : ''}`;
  return(
    <div className={className} onClick={props.onClick} role="button"
    tabIndex={0} onKeyPress={(e)=>{ if(e.key === 'Enter') props.onClick &&
    props.onClick(); }}>
      <h2>{props.miasto}</h2>
      <div className="meta">
        <div className="temp">{props.temperatura ? props.temperatura
        + '°C' : '-'}</div>
        <div className="cond">{props.pogoda || ''}</div>
      </div>
    </div>
  )
```

```
}
```

```
export default WeatherCard;
```

### PO (cały plik z komentarzami):

```
// =====
// PLIK: src/components/WeatherCard.jsx
// ZMIANY: Dodajemy przeliczanie temperatury na podstawie Redux
// =====

// ↓↓↓ NOWE IMPORTY ↓↓↓
import { useSelector } from 'react-redux';
import { convertTemperature, getUnitSymbol } from '../utils/temperature';
// ↑↑↑ KONIEC NOWYCH IMPORTÓW ↑↑↑

function WeatherCard(props){
    // ↓↓↓ NOWE: Odczytujemy jednostkę z Redux ↓↓↓
    const unit = useSelector((state) => state.settings.temperatureUnit);

    // ↓↓↓ NOWE: Przeliczamy temperaturę ↓↓↓
    // props.temperatura to zawsze Celsjusze (z danych)
    // displayTemp to przeliczona wartość (C, F lub K)
    const displayTemp = convertTemperature(props.temperatura, unit);
    const unitSymbol = getUnitSymbol(unit);
    // ↑↑↑ KONIEC NOWEGO ↑↑↑

    const className = `city-card ${props.selected ? 'selected' : ''}`;

    return(
        <div
            className={className}
            onClick={props.onClick}
            role="button"
            tabIndex={0}
            onKeyPress={(e) => { if(e.key === 'Enter') props.onClick && props.onClick(); }}
        >
            <h2>{props.miasto}</h2>
            <div className="meta">
                {/* ↓↓↓ ZMIENIONE: Używamy przeliczonej temperatury ↓↓↓ */}
                <div className="temp">
                    {props.temperatura ? displayTemp + unitSymbol : '-'}
                </div>
                {/* BYŁO: {props.temperatura ? props.temperatura + '°C' : '-' */}
            <div className="cond">{props.pogoda || ''}</div>
        </div>
    )
}
```

```

    }

export default WeatherCard;

```

### Podsumowanie zmian w WeatherCard.jsx:

| Miejsce            | Było                                  | Jest  |
|--------------------|---------------------------------------|---|
| Importy            | brak                                  | <pre>import { useSelector } from 'react-redux'; import { convertTemperature, getUnitSymbol } from '../utils/temperature';</pre> |
| W funkcji          | brak                                  | <pre>const unit = useSelector(...) const displayTemp = convertTemperature(...) const unitSymbol = getUnitSymbol(...)</pre>      |
| Wyświetlanie temp. | <code>props.temperatura + '°C'</code> | <code>displayTemp + unitSymbol</code>   |

### Krok 5: Aktualizujemy WeatherDetails

#### ■ MODYFIKUJEMY: src/components/weatherDetails.jsx

##### PRZED (początek pliku):

```

import WeatherIcon from './WeatherIcon'

function WeatherDetails({ miasto }) {
  if (!miasto) return null;
}

```

##### PO (początek pliku):

```

import WeatherIcon from './WeatherIcon'
// ↓↓ NOWE IMPORTY ↓↓
import { useSelector } from 'react-redux';
import { convertTemperature, getUnitSymbol } from '../utils/temperature';
// ↑↑ KONIEC NOWYCH IMPORTÓW ↑↑

function WeatherDetails({ miasto }) {
  // ↓↓ NOWE: Odczytujemy jednostkę z Redux ↓↓
  // UWAGA: To musi być PRZED if(!miasto) return null;
  const unit = useSelector((state) => state.settings.temperatureUnit);
  const unitSymbol = getUnitSymbol(unit);
  // ↑↑ KONIEC NOWEGO ↑↑

  if (!miasto) return null;
}

```

### **PRZED (wyświetlanie aktualnej temperatury):**

```
<div className="details-item">
  <strong>Temperatura:</strong>
  <div>{miasto.aktualnaTemperatura} °C</div>
</div>
```

### **PO (wyświetlanie aktualnej temperatury):**

```
<div className="details-item">
  <strong>Temperatura:</strong>
  {/* $$$ ZMIENIONE $$$ */}
  <div>{convertTemperature(miasto.aktualnaTemperatura,
  unit)}{unitSymbol}</div>
  {/* BYŁO: <div>{miasto.aktualnaTemperatura} °C</div> */}
</div>
```

### **PRZED (prognoza 5-dniowa):**

```
<div className="details-item">Temperatura: {dzień.temperatura} °C</div>
```

### **PO (prognoza 5-dniowa):**

```
{/* $$$ ZMIENIONE $$$ */}
<div className="details-item">
  Temperatura: {convertTemperature(dzień.temperatura, unit)}{unitSymbol}
</div>
{/* BYŁO: Temperatura: {dzień.temperatura} °C */}
```

### **Podsumowanie zmian w weatherDetails.jsx:**

| Miejsce                     | Co zmieniamy   |
|-----------------------------|--|
| Importy (góra)              | Dodajemy useSelector i funkcje z temperature.js  |
| Początek funkcji (przed if) | Dodajemy const unit = useSelector(...) i const unitSymbol = ...  |
| Aktualna temperatura        | {miasto.aktualnaTemperatura} °C →<br>{convertTemperature(miasto.aktualnaTemperatura,<br>unit)}{unitSymbol} |
| Prognoza 5-dniowa           | {dzień.temperatura} °C →<br>{convertTemperature(dzień.temperatura,<br>unit)}{unitSymbol}                   |

## CHECKPOINT 1 - Sprawdźcie!

W tym momencie powinno działać: 1. Na stronie głównej jest select z jednostkami 2. Zmiana jednostki zmienia temperatury na kartach miast 3. Po wejściu w szczegóły miasta - temperatura też jest przeliczona

## CZĘŚĆ 4: Ulubione miasta (30 min)

### Krok 1: Nowy slice dla ulubionych

#### ■ NOWY PLIK: src/store/slices/favoritesSlice.js

```
// _____  
// PLIK: src/store/slices/favoritesSlice.js  
// OPIS: Slice odpowiedzialny za ulubione miasta  
// _____  
  
import { createSlice } from '@reduxjs/toolkit';  
  
// _____  
// STAN POCZĄTKOWY  
// favoriteIds to tablica z ID ulubionych miast  
// Przykład: [1, 3, 5] = Warszawa, Wrocław i Gdańsk są ulubione  
// _____  
const initialState = {  
    favoriteIds: [],  
};  
  
const favoritesSlice = createSlice({  
    name: 'favorites',  
    initialState,  
    reducers: {  
        // _____  
        // REDUCER: toggleFavorite  
        // Co robi: Dodaje miasto do ulubionych LUB usuwa je  
        //          (jeśli już było ulubione)  
        // Parametr: action.payload = ID miasta (liczba)  
        // _____  
        toggleFavorite: (state, action) => {  
            const cityId = action.payload; // np. 1 (Warszawa)  
  
            // Szukamy czy to miasto już jest w ulubionych  
            const index = state.favoriteIds.indexOf(cityId);  
  
            if (index === -1) {  
                // indexOf zwraca -1 gdy nie znaleziono
```

```

        // Czyli miasto NIE jest w ulubionych - dodajemy je
        state.favoriteIds.push(cityId);
    } else {
        // Miasto JEST w ulubionych - usuwamy je
        state.favoriteIds.splice(index, 1);
    }
},
},
});
};

export const { toggleFavorite } = favoritesSlice.actions;
export default favoritesSlice.reducer;

```

### Jak to działa:

Stan początkowy: { favoriteIds: [] }

dispatch(toggleFavorite(1)) → { favoriteIds: [1] } // Dodano Warszawę

dispatch(toggleFavorite(3)) → { favoriteIds: [1, 3] } // Dodano Wrocław

dispatch(toggleFavorite(1)) → { favoriteIds: [3] } // Usunięto Warszawę

## Krok 2: Dodajemy reducer do store

### MODYFIKUJEMY: src/store/index.js

#### PRZED:

```

import { configureStore } from '@reduxjs/toolkit';
import settingsReducer from './slices/settingsSlice';

const store = configureStore({
    reducer: {
        settings: settingsReducer,
    },
});

export default store;

```

#### PO:

```

import { configureStore } from '@reduxjs/toolkit';
import settingsReducer from './slices/settingsSlice';
import favoritesReducer from './slices/favoritesSlice'; // ← NOWE

const store = configureStore({
    reducer: {

```

```

    settings: settingsReducer,
    favorites: favoritesReducer, // ← NOWE
  },
});

// Teraz stan w STORE wygląda tak:
// {
//   settings: { temperatureUnit: 'C' },
//   favorites: { favoriteIds: [] }
// }

export default store;

```

### Krok 3: Komponent gwiazdki

#### ■ NOWY PLIK: src/components/FavoriteButton.jsx

```

// =====
// PLIK: src/components/FavoriteButton.jsx
// OPIS: Przycisk gwiazdki do dodawania/usuwania z ulubionych
// =====

import { useSelector, useDispatch } from 'react-redux';
import { toggleFavorite } from '../store/slices/favoritesSlice';

function FavoriteButton({ cityId }) {
  // cityId to props - ID miasta (np. 1 dla Warszawy)

  const dispatch = useDispatch();

  // Pobieramy tablicę ulubionych z Redux
  const favoriteIds = useSelector((state) => state.favorites.favoriteIds);

  // Sprawdzamy czy TO miasto jest w ulubionych
  const isFavorite = favoriteIds.includes(cityId);

  const handleClick = (e) => {
    // -----
    // WAŻNE: stopPropagation()
    // Bez tego kliknięcie w gwiazdkę kliknęłoby też w całą kartę
    // i przeszłabyśmy do szczegółów miasta
    // -----
    e.stopPropagation();

    // Wysyłamy akcję toggle - doda lub usunie z ulubionych
    dispatch(toggleFavorite(cityId));
  };
}
```

```

    return (
      <button
        onClick={handleClick}
        style={{
          background: 'none',
          border: 'none',
          fontSize: '1.5rem',
          cursor: 'pointer',
          padding: '0.25rem'
        }}
        title={isFavorite ? 'Usuń z ulubionych' : 'Dodaj do ulubionych'}
      >
        {/* Pełna gwiazdka gdy ulubione, pusta gdy nie */}
        {isFavorite ? '★' : '☆'}
      </button>
    );
}

export default FavoriteButton;

```

## Krok 4: Dodajemy gwiazdkę do WeatherCard

■ **MODYFIKUJEMY:** `src/components/WeatherCard.jsx`

**Dodajemy import na górze (do istniejących importów):**

```
import FavoriteButton from './FavoriteButton'; // ← NOWE
```

**PRZED (fragment renderowania):**

```

return(
  <div
    className={className}
    onClick={props.onClick}
    // ... atrybuty ...
  >
    <h2>{props.miasto}</h2>
    <div className="meta">

```

**PÓŁ (fragment renderowania):**

```

return(
  <div
    className={className}
    onClick={props.onClick}
    // ... atrybuty ...
  >
    /* !!! ZMIENIONE: Dodajemy kontener z flexbox i gwiazdkę !!! */
    <div style={{ display: 'flex', justifyContent: 'space-between',
```

```

alignItems: 'center' }}>
    <h2>{props.miasto}</h2>
    <FavoriteButton cityId={props.cityId} />
</div>
/* BYŁO: <h2>{props.miasto}</h2> */
/* ↑↑↑ KONIEC ZMIAN ↑↑↑ */

<div className="meta">

```

### Podsumowanie zmian w WeatherCard.jsx:

| Miejsce  | Było                    | Jest  |
|----------|-------------------------|---|
| Import   | -                       | import FavoriteButton from './FavoriteButton';              |
| Nagłówek | <h2>{props.miasto}</h2> | <div style={{...}}><h2>...</h2><FavoriteButton ... /></div> |

### Krok 5: Przekazujemy cityId w HomePage

■ MODYFIKUJEMY: [src/Pages/HomePage.jsx](#)

Szukamy w pliku tego fragmentu:

```

{filteredMiasta.map((dane) => (
  <WeatherCard
    key={dane.id}
    miasto={dane.miasto}
    temperatura={dane.aktualnaTemperatura}
    onClick={() => handleClick(dane)}
  />
))
}
```

Zmieniamy na:

```

{filteredMiasta.map((dane) => (
  <WeatherCard
    key={dane.id}
    cityId={dane.id}           // ← NOWE: przekazujemy ID miasta
    miasto={dane.miasto}
    temperatura={dane.aktualnaTemperatura}
    onClick={() => handleClick(dane)}
  />
))
}
```

## Krok 6: Strona ulubionych miast

■ NOWY PLIK: src/Pages/FavoritesPage.jsx

```
// =====
// PLIK: src/Pages/FavoritesPage.jsx
// OPIS: Strona wyświetlająca tylko ulubione miasta
// =====

import { useSelector } from 'react-redux';
import { useNavigate } from 'react-router-dom';
import WeatherCard from '../components/WeatherCard';
import UnitSwitcher from '../components/UnitSwitcher';

function FavoritesPage({ miasta }) {
    // miasta = wszystkie miasta (props z App.jsx)

    const navigate = useNavigate();

    // Pobieramy tablicę ID ulubionych
    const favoriteIds = useSelector((state) => state.favorites.favoriteIds);

    // Filtrujemy - zostawiamy tylko te miasta, których ID jest w ulubionych
    // Przykład: favoriteIds = [1, 3]
    // miasta = [{id:1, ...}, {id:2, ...}, {id:3, ...}, ...]
    // favoriteCities = [{id:1, ...}, {id:3, ...}]
    const favoriteCities = miasta.filter(m => favoriteIds.includes(m.id));

    return (
        <div>
            <h1>★ Ulubione miasta</h1>

            <UnitSwitcher />

            <button
                onClick={() => navigate('/')}
                style={{ margin: '1rem 0' }}
            >
                ← Powrót do listy
            </button>

            {favoriteCities.length === 0 ? (
                // Gdy nie ma ulubionych - pokazujemy komunikat
                <p>Nie masz jeszcze ulubionych miast. Kliknij ★ przy mieście, żeby dodać.</p>
            ) : (
                // Gdy są ulubione - pokazujemy karty
                <div className="cities-container">
                    {favoriteCities.map((dane) => (
                        <WeatherCard
```

```

        key={dane.id}
        cityId={dane.id}
        miasto={dane.miasto}
        temperatura={dane.aktualnaTemperatura}
        onClick={() => navigate(`/${miasto}/${dane.id}`)}
      />
    ))
</div>
)
</div>
);
}

export default FavoritesPage;

```

## Krok 7: Dodajemy route i nawigację

■ MODYFIKUJEMY: `src/App.jsx`

**PRZED (importy):**

```

import HomePage from './Pages/HomePage'
import CityDetailPage from './Pages/CityDetailPage'

```

**PO (importy):**

```

import HomePage from './Pages/HomePage'
import CityDetailPage from './Pages/CityDetailPage'
import FavoritesPage from './Pages/FavoritesPage' // ← NOWE

```

**PRZED (Routes):**

```

<Routes>
  <Route path="/" element={<HomePage miasta={miasta}>} />
  <Route path="/miasto/:cityId" element={<CityDetailPage miasta={miasta}>} />
</Routes>

```

**PO (Routes):**

```

<Routes>
  <Route path="/" element={<HomePage miasta={miasta}>} />
  <Route path="/miasto/:cityId" element={<CityDetailPage miasta={miasta}>} />
  {/* ↓↓↓ NOWE: Route do strony ulubionych ↓↓↓ */}
  <Route path="/ulubione" element={<FavoritesPage miasta={miasta}>} />
</Routes>

```

## ■ MODYFIKUJEMY: src/Pages/HomePage.jsx - dodajemy przycisk

Szukamy (pod wyszukiwarką):

```
</div>
<div className="cities-container">
```

Dodajemy między nimi przycisk:

```
</div>

/* ↓↓↓ NOWE: Przycisk do ulubionych ↓↓↓ */
<button
  onClick={() => navigate('/ulubione')}
  style={{ margin: '1rem 0' }}
>
  ★ Ulubione miasta
</button>
/* ↑↑↑ KONIEC NOWEGO ↑↑↑ */

<div className="cities-container">
```

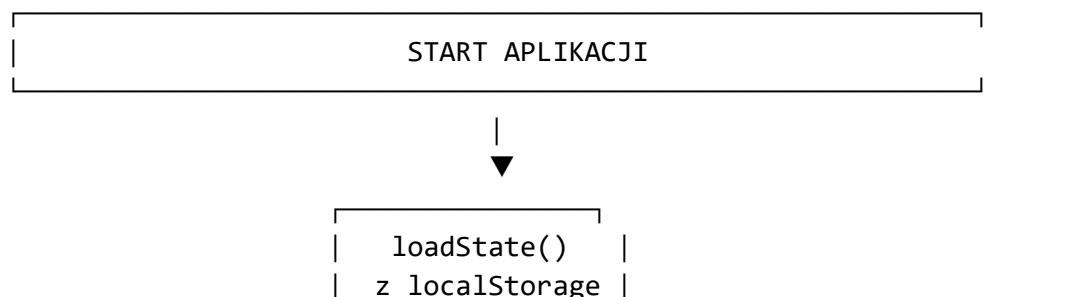
### 🛠 CHECKPOINT 2 - Sprawdźcie!

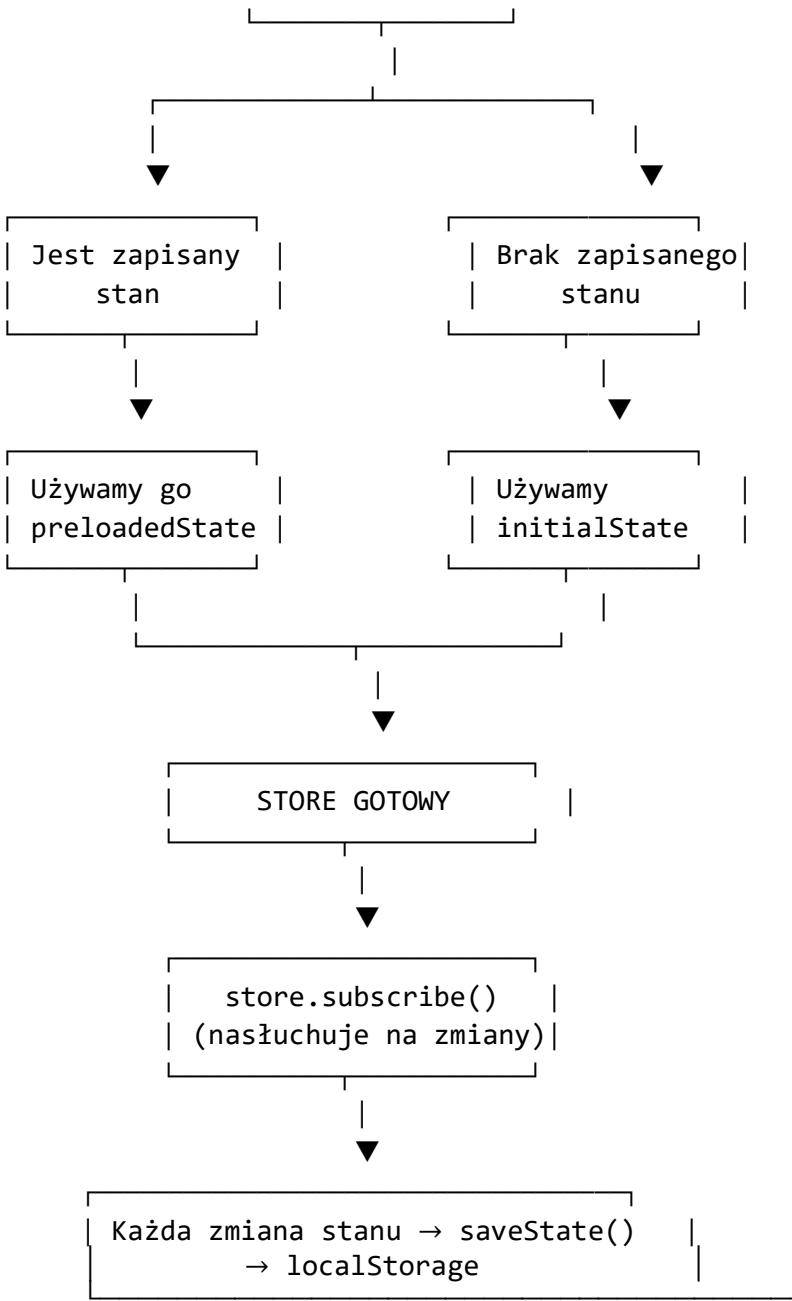
- Przy każdym mieście jest gwiazdka (★)
- Kliknięcie gwiazdki zmienia ją na ★ (i odwrotnie)
- Przycisk “Ulubione miasta” prowadzi do /ulubione
- Na stronie ulubionych widać tylko oznaczone miasta

## CZĘŚĆ 5: LocalStorage (20 min)

Chcemy żeby ustawienia zapisywały się w przeglądarce i przetrwały odświeżenie strony.

Jak to działa - schemat





## Krok 1: Modyfikujemy store

■ **MODYFIKUJEMY: src/store/index.js (zastępujemy CAŁY PLIK)**

```
// _____
// PLIK: src/store/index.js
// ZMIANY: Dodajemy zapis/odczyt z localStorage
// _____
```

```

import { configureStore } from '@reduxjs/toolkit';
import settingsReducer from './slices/settingsSlice';
import favoritesReducer from './slices/favoritesSlice';

// -----
// FUNKCJA: LoadState
// Co robi: ładuje zapisany stan z localStorage
// Kiedy wywoływana: Raz, przy starcie aplikacji
// Zwraca: Obiekt ze stanem LUB undefined (gdy brak zapisanego)
// -----
const loadState = () => {
  try {
    // Pobieramy string z localStorage
    const serializedState = localStorage.getItem('weatherAppState');

    if (serializedState === null) {
      // Nie ma zapisanego stanu - zwracamy undefined
      // Redux używa wtedy initialState z slice'ów
      return undefined;
    }

    // Parsujemy JSON na obiekt
    return JSON.parse(serializedState);
  } catch (err) {
    // Błąd (np. uszkodzony JSON) - logujemy i zwracamy undefined
    console.error('Błąd ładowania stanu z localStorage:', err);
    return undefined;
  }
};

// -----
// FUNKCJA: saveState
// Co robi: Zapisuje aktualny stan do localStorage
// Kiedy wywoływana: Za każdym razem gdy zmieni się stan
// -----
const saveState = (state) => {
  try {
    // Tworzymy obiekt tylko z tym co chcemy zapisać
    const stateToSave = {
      settings: state.settings,
      favorites: state.favorites,
    };

    // Zamieniamy na string JSON i zapisujemy
    const serializedState = JSON.stringify(stateToSave);
    localStorage.setItem('weatherAppState', serializedState);
  } catch (err) {
    console.error('Błąd zapisywania stanu do localStorage:', err);
  }
};

```

```

};

// -----
// ŁADOWANIE STANU PRZY STARCIE
// -----
const preloadedState = loadState();

// -----
// KONFIGURACJA STORE
// -----
const store = configureStore({
  reducer: {
    settings: settingsReducer,
    favorites: favoritesReducer,
  },
  preloadedState, // ← Używamy załadowanego stanu (jeśli istnieje)
});

// -----
// SUBSKRYPCJA - ZAPISYWANIE PRZY ZMIANACH
// store.subscribe() wywołuje funkcję za każdym razem
// gdy stan się zmieni (po każdej akcji)
// -----
store.subscribe(() => {
  saveState(store.getState());
});

export default store;

```

**Co dodaliśmy:**

| Element           | Co robi  |
|-------------------|--|
| loadState()       | Przy starcie ładuje stan z localStorage          |
| saveState()       | Zapisuje stan do localStorage                    |
| preloadedState    | Przekazuje załadowany stan do store              |
| store.subscribe() | Za każdym razem gdy stan się zmieni, zapisuje go |

## 🛠 GOTOWE! Sprawdźcie całość

- Zmieńcie jednostkę na Fahrenheit
- Dodajcie 2-3 miasta do ulubionych (kliknijcie gwiazdki)

- **Odświeżcie stronę (F5)**
  - ✓ Jednostka powinna zostać na Fahrenheit
  - ✓ Ulubione miasta powinny być zapamiętane

**Bonus:** Otwórzcie DevTools → Application → Local Storage → localhost i zobaczcie zapisany JSON!

## Podsumowanie zajęć

## Nowe pliki które utworzyliście:

| Plik                               | Co robi                           |
|------------------------------------|-----------------------------------|
| src/store/index.js                 | Główny store Redux + localStorage |
| src/store/slices/settingsSlice.js  | Stan jednostki temperatury        |
| src/store/slices/favoritesSlice.js | Stan ulubionych miast             |
| src/utils/temperature.js           | Funkcje przeliczające temperaturę |
| src/components/UnitSwitcher.jsx    | Select do zmiany jednostek        |
| src/components/FavoriteButton.jsx  | Przycisk gwiazdki                 |
| src/Pages/FavoritesPage.jsx        | Strona z ulubionymi miastami      |

## Zmodyfikowane pliki:

| Plik                              | Co zmieniliście  |
|-----------------------------------|--|
| src/main.jsx                      | Dodaliście <Provider>  |
| src/App.jsx                       | Dodaliście route /ulubione                                     |
| src/Pages/HomePage.jsx            | Dodaliście UnitSwitcher, przycisk ulubionych, cityId w kartach |
| src/components/WeatherCard.jsx    | Przeliczanie temp., gwiazdka                                   |
| src/components/weatherDetails.jsx | Przeliczanie temp.   |

## Struktura po zajęciach:

```
src/
  └── store/
    ├── index.js
    └── slices/
      ├── settingsSlice.js
      └── favoritesSlice.js
  └── utils/
    └── temperature.js
  └── components/
    ├── WeatherCard.jsx
    └── weatherDetails.jsx
```

```
|   ├── WeatherIcon.jsx           ← NOWY
|   ├── UnitSwitcher.jsx         ← NOWY
|   └── FavoriteButton.jsx
|   Pages/
|   ├── HomePage.jsx            ← ZMODYFIKOWANY
|   ├── CityDetailPage.jsx
|   └── FavoritesPage.jsx       ← NOWY
|   App.jsx                     ← ZMODYFIKOWANY
|   main.jsx                    ← ZMODYFIKOWANY
|   ...

```

### Co macie teraz:

| Wymaganie                | Status |
|--------------------------|--------|
| Redux - zmiana jednostek | ✓      |
| Ulubione miasta          | ✓      |
| LocalStorage             | ✓      |

🎉 Macie aplikację na ocenę 4.5!

Do oceny 5.0 brakuje integracji z API OpenWeatherMap - to zrobimy na kolejnych zajęciach.