

Aplicație de vreme și de știri

ABSOLVENT

Borze Andrei

2023

CUPRINS

1. INTRODUCERE	3
2. DESCRIEREA APLICAȚIEI	4
2.1 App.tsx	4
2.2 WeatherPage.tsx	6
2.3 NewsPage.tsx.....	10
2.3 Store și Api	15
CONCLUZII.....	23
IMAGINI CU APLICAȚIA	24

1. INTRODUCERE

Redux este o librărie de management al stării aplicației care poate fi utilizată împreună cu React sau cu orice altă librărie de interfață utilizată pentru construirea interfețelor utilizator (UI). Redux gestionează starea aplicației într-un singur obiect numit store și oferă un set de metode standardizate pentru a actualiza starea din store. Utilizarea Redux poate simplifica managementul stării, împiedicând situații de cod duplicat și menținând o arhitectură scalabilă și ușor de menținut în timp.

Redux oferă o serie de avantaje importante în dezvoltarea unei aplicații web, cum ar fi:

- **Managementul stării:** Redux oferă un mecanism centralizat pentru gestionarea stării unei aplicații, ceea ce face mai ușor de urmărit și de întreținut. Starea poate fi gestionată într-un singur loc, numit magazin (store), făcând aplicația mai predictibilă și mai ușor de testat.
- **Scalabilitate:** Redux este proiectat să funcționeze bine cu aplicații mari, care au multe componente și fluxuri de date complexe. Datorită gestionării centrale a stării, este mai ușor să se adauge și să se întrețină noi funcționalități, fără a fi necesar să se schimbe structura existentă a aplicației.
- **DevTools:** Redux vine cu un set de DevTools care oferă o modalitate convenabilă de a urmări și de a depana starea aplicației. Aceste instrumente sunt foarte utile în timpul dezvoltării și depanării și pot economisi mult timp.
- **Compatibilitate:** Redux poate fi integrat cu o varietate de alte tehnologii și biblioteci, cum ar fi React, Angular, Vue. Aceasta face posibilă dezvoltarea unei aplicații folosind o combinație de tehnologii care să corespundă nevoilor specifice ale proiectului.

Având în vedere faptul că Redux oferă un mod mai structurat și mai ușor de urmărit pentru gestionarea stării unei aplicații web și este mai ușor de întreținut pe termen lung, s-a decis implementarea aplicației cu această tehnologie.

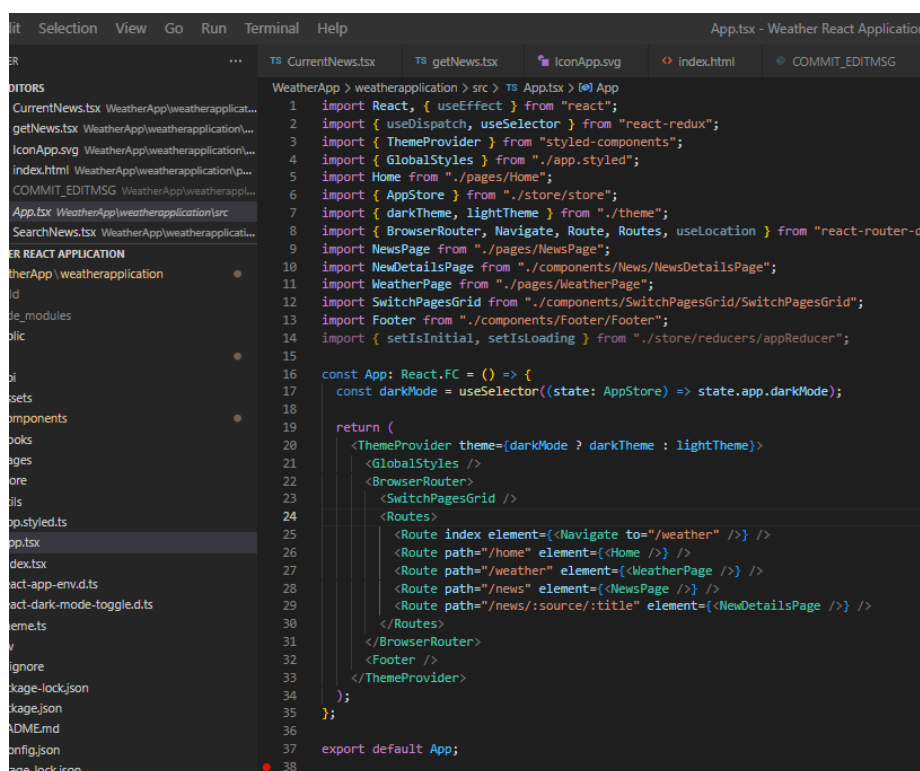
2. DESCRIEREA APLICAȚIEI

2.1 App.tsx

Acest cod reprezintă componenta principală a aplicației React, care utilizează diverse alte componente, biblioteci și tehnologii pentru a crea o interfață web dinamică și eficientă.

Componenta importă React, useEffect, useDispatch, useSelector, ThemeProvider, GlobalStyles, BrowserRouter, Navigate, Route, Routes, useLocation și mai multe alte componente și pagini.

Aceasta definește o componentă funcțională numită App care returnează o expresie JSX care se randează în DOM. Componenta App este încapsulată într-un ThemeProvider de la styled-components, care furnizează un aspect dinamic în funcție de preferința utilizatorului pentru modul întunecat sau luminos. Componenta utilizează, de asemenea, BrowserRouter și Routes de la react-router-dom pentru a defini sistemul de rutare al aplicației și pentru a afișa diferite pagini în funcție de calea URL. Componenta utilizează diverse alte componente și pagini pentru a afișa conținutul necesar, cum ar fi Home, NewsPage, NewDetailsPage și WeatherPage.



```
1 import React, { useEffect } from "react";
2 import { useDispatch, useSelector } from "react-redux";
3 import { ThemeProvider } from "styled-components";
4 import { GlobalStyles } from "../app.styled";
5 import Home from "../pages/Home";
6 import { AppStore } from "../store/store";
7 import { darkTheme, lightTheme } from "../theme";
8 import { BrowserRouter, Navigate, Route, Routes, useLocation } from "react-router-dom";
9 import NewsPage from "../pages/NewsPage";
10 import NewDetailsPage from "../components/News/NewsDetailsPage";
11 import WeatherPage from "../pages/WeatherPage";
12 import SwitchPagesGrid from "../components/SwitchPagesGrid/SwitchPagesGrid";
13 import Footer from "../components/Footer/Footer";
14 import { setIsInitial, setIsLoading } from "../store/reducers/appReducer";
15
16 const App: React.FC = () => {
17   const darkMode = useSelector((state: AppStore) => state.app.darkMode);
18
19   return (
20     <ThemeProvider theme={darkMode ? darkTheme : lightTheme}>
21       <GlobalStyles />
22       <BrowserRouter>
23         <SwitchPagesGrid />
24         <Routes>
25           <Route index element={<Navigate to="/weather" /> } />
26           <Route path="/home" element={<Home /> } />
27           <Route path="/weather" element={<WeatherPage /> } />
28           <Route path="/news" element={<NewsPage /> } />
29           <Route path="/news/:source/:title" element={<NewDetailsPage /> } />
30         </Routes>
31       </BrowserRouter>
32       <Footer />
33     </ThemeProvider>
34   );
35 }
36
37 export default App;
```

Figura nr. 2.1 App.tsx

Interfață temei din figura 2.2 definește proprietățile și aspectul componentelor vizuale ale aplicației, în funcție de tema aleasă de utilizator (light sau dark). Aceste proprietăți includ culoarea de fundal și gradientul, culorile textului și a iconițelor, culorile butoanelor și a intrării de căutare. Interfața este aplicată la nivelul întregii aplicații și poate fi utilizată pentru a schimba aspectul vizual al acesteia în funcție de preferințele utilizatorilor.

```
WeatherApp > weatherapplication > src > TS theme.ts > +o Theme >
1  import lightBg from './assets/bg.svg';
2  import darkBg from './assets/darkBg.svg';
3
4  export interface Theme {
5    backgroundImage: string;
6    backgroundGradient: {
7      color1: string;
8      color2: string;
9    };
10   temperatureSwitch: {
11     backgroundColor: string;
12     sliderColor: string;
13     textColor: string;
14   };
15   searchSuggestion: {
16     textColor: string;
17     backgroundColor: string;
18     hoverBackgroundColor: string;
19     separatorLineColor: string;
20   };
21
22   header: {
23     backgroundColor: string;
24     titleColor: string;
25     iconColor: string;
26   };
27   search: {
28     backgroundColor: string;
29     iconColor: string;
30     searchInput: {
31       color: string;
32       placeholderColor: string;
33     },
34     button: {
35       color: string;
36       placeholderColor: string;
37       hoverColor: string;
38     }
39   };
40   CurrentWeather: {
41     containerBgColor: string;
42     textColor: string;
43
44     cityNameContainer: {
45       containerBgColor: string;
46       textColor: string;
47     },
48     temperatureContainer: {
49       containerBgColor: string;
50       textColor: string;
51     }
52   };
53   Forecast: {
54     containerBgColor: string;
55     textColor: string;
56   },
57   CurrentNewsContainer: {
58     containerBgColor: string;
59     textColor: string;
60   };
61
62   ArticleNewContainer: {
63     containerBgColor: string;
64     textColor: string;
65   };
66   SwitchPagesGrid: {
67     containerBgColor: string;
68     textColor: string;
69   };
70 }
71
```

Figura nr. 2.2 theme.ts

2.2 WeatherPage.tsx

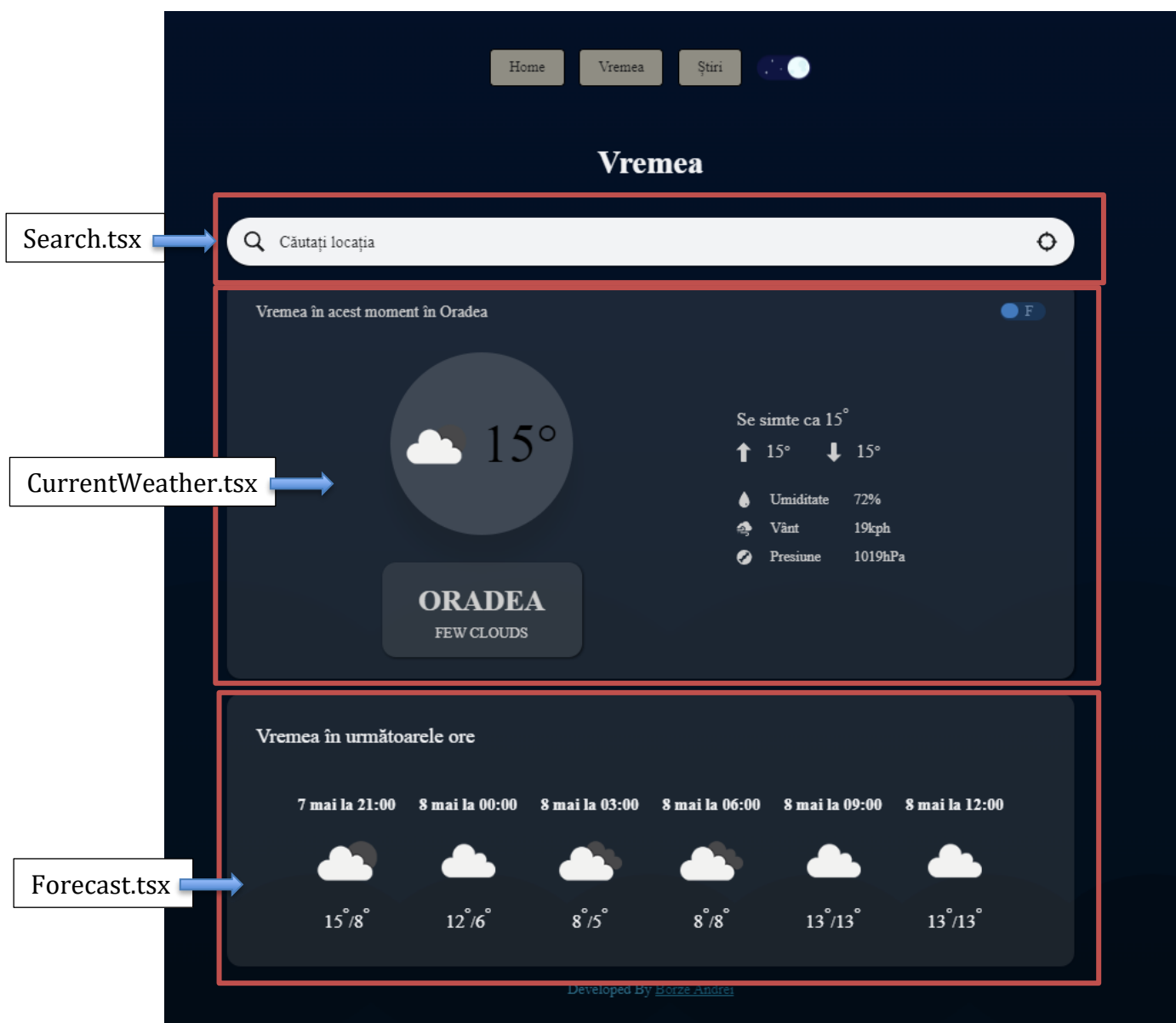
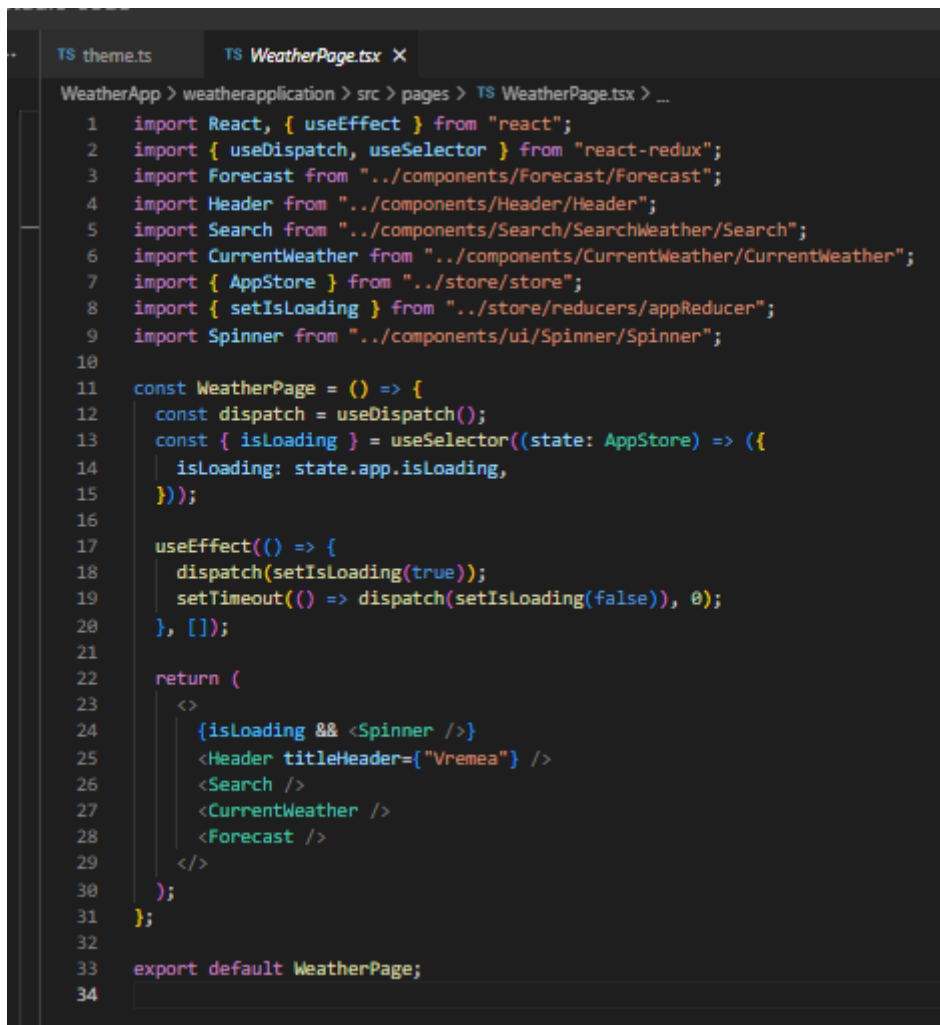


Figura nr. 2.3 WeatherPage.tsx

Componenta WeatherPage reprezintă pagina o a aplicației, care afișează informații despre vreme. Componenta conține un antet, o bară de căutare a locației dorite, informații despre vremea curentă și o prognoză pentru vremea în orele următoare în locația aleasă.

De asemenea, componenta are un spinner care este afișat atunci când se încarcă informațiile despre vreme. Această componentă este conectată la store-ul Redux al aplicației, folosind hook-urile “useDispatch” și “useSelector”, pentru a prelua și a actualiza starea aplicației.

În “useEffect”, se inițializează starea de încărcare cu “setIsLoading(true)”, după care se așteaptă 0 milisecunde și se schimbă starea de încărcare cu “setIsLoading(false)”.



```
1 import React, { useEffect } from "react";
2 import { useDispatch, useSelector } from "react-redux";
3 import Forecast from "../components/Forecast/Forecast";
4 import Header from "../components/Header/Header";
5 import Search from "../components/Search/SearchWeather/Search";
6 import CurrentWeather from "../components/CurrentWeather/CurrentWeather";
7 import { AppState } from "../store/store";
8 import { setIsLoading } from "../store/reducers/appReducer";
9 import Spinner from "../components/ui/Spinner/Spinner";
10
11 const WeatherPage = () => {
12   const dispatch = useDispatch();
13   const { isLoading } = useSelector((state: AppState) => ({
14     isLoading: state.app.isLoading,
15   }));
16
17   useEffect(() => {
18     dispatch(setIsLoading(true));
19     setTimeout(() => dispatch(setIsLoading(false)), 0);
20   }, []);
21
22   return (
23     <>
24       {isLoading && <Spinner />}
25       <Header titleHeader={"Vremea"} />
26       <Search />
27       <CurrentWeather />
28       <Forecast />
29     </>
30   );
31 };
32
33 export default WeatherPage;
34
```

Figura nr. 2.4 WeatherPage.tsx

Componenta Search din figura nr. 2.4 reprezintă un câmp de căutare pentru locația utilizatorului, cu opțiunea de a utiliza geolocația pentru a obține informații despre vremea din zona curentă. În plus, componenta afișează sugestii de orașe în funcție de termenul de căutare introdus de utilizator. Când utilizatorul introduce un termen de căutare, sugestiile sunt afișate într-o listă cu opțiuni de selectare.

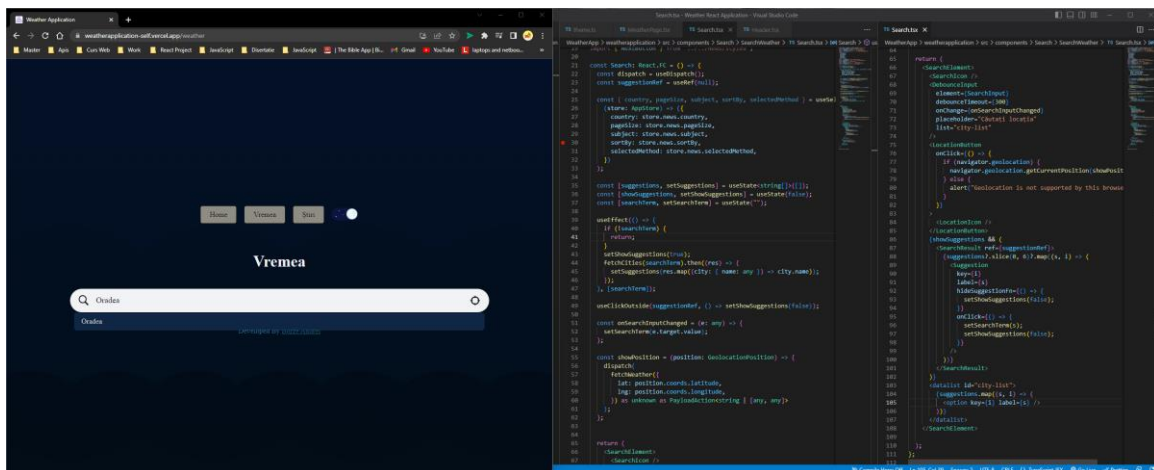


Figura nr. 2.4 Search.tsx

CurrentWeather din figura 2.5 afișează informații despre vremea curentă pentru o anumită locație, folosind date de la API-ul OpenWeatherMap. Componenta include informații despre temperatura curentă, temperatura maximă și minimă, umiditate, vânt și presiune. De asemenea, include și un comutator pentru a schimba unitatea de măsură a temperaturii între Celsius și Fahrenheit. Dacă este o problemă cu încărcarea datelor meteo pentru locație, se afișează un mesaj de eroare.

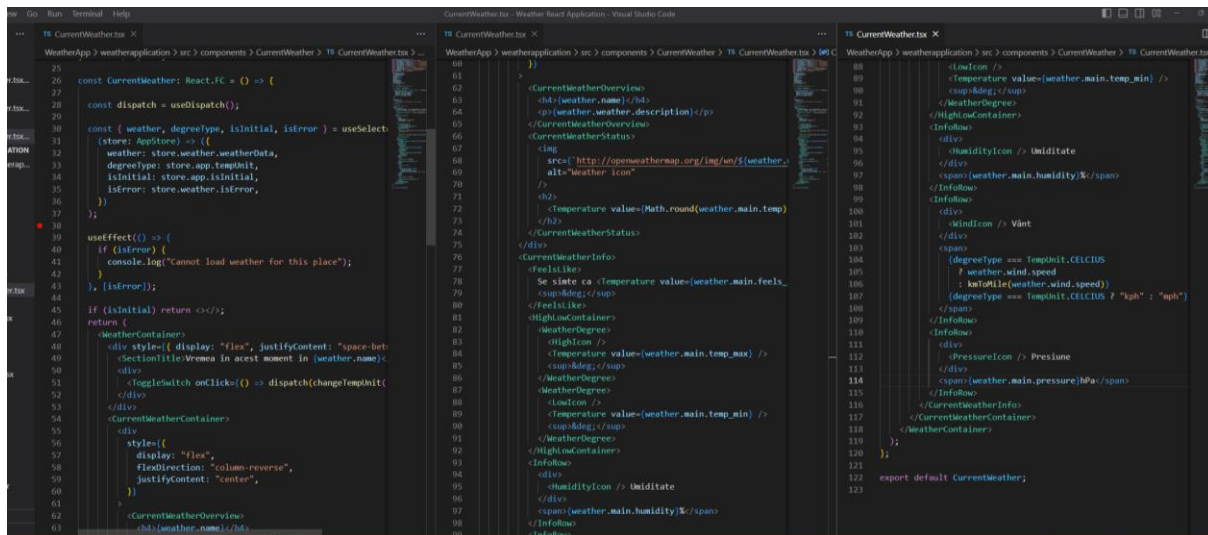
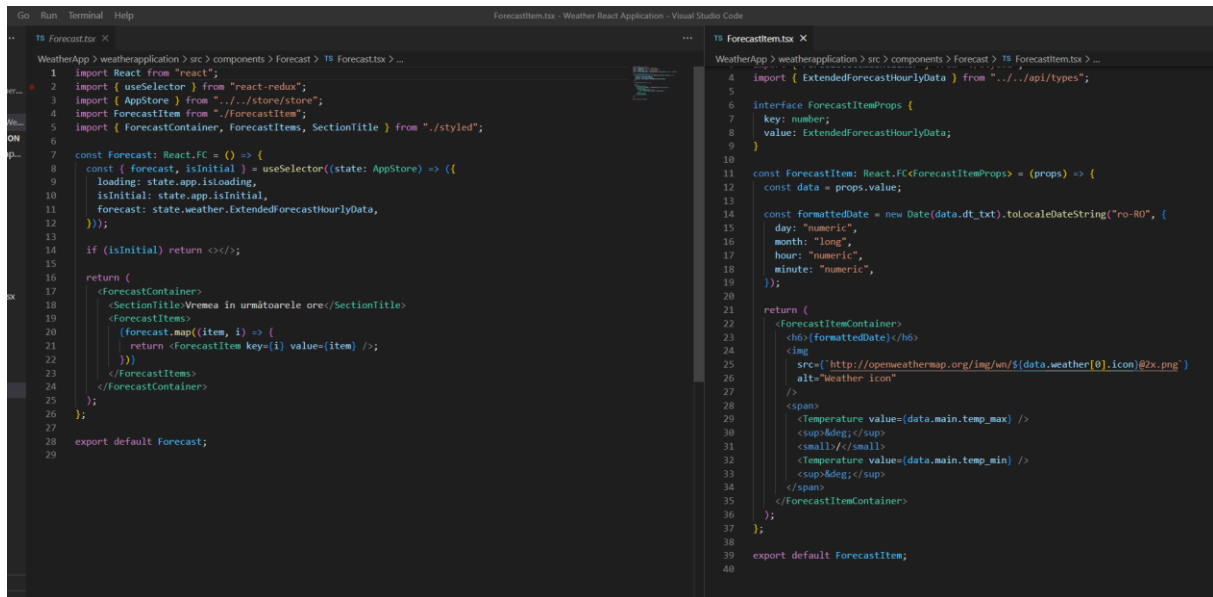


Figura nr. 2.5 CurrentWeather.tsx

Forecast.tsx (figura 2.6) este o componentă React care afișează prognoza meteo pentru următoarele ore. Utilizează hook-ul “useSelector” pentru a prelua datele din store-ul Redux al aplicației. Dacă “isInitial” este adevărat, atunci componenta returnează un element gol. În caz contrar, se afișează prognoza meteo într-un container cu un titlul și o listă de elemente de

prognoză. Fiecare element de prognoză este un componentă separată (“ForecastItem.tsx”), care este reutilizată în lista de elemente de prognoză și primește ca proprietate “value” datele de prognoză.

Componenta “ForecastItem.tsx” afișează un element de prognoză pentru o oră specifică. Acesta primește ca proprietate “value” datele de prognoză și afișează informații precum data, o imagine cu iconița vremii și temperaturile minime și maxime pentru acea oră.



```
WeatherApp > weatherapplication > src > components > Forecast > TS Forecast.tsx > ...
1 import React from "react";
2 import { useSelector } from "react-redux";
3 import { AppStore } from "../../store/store";
4 import ForecastItem from "../ForecastItem";
5 import { ForecastContainer, ForecastItems, SectionTitle } from "../styled";
6
7 const Forecast: React.FC = () => {
8   const { forecast, isInitial } = useSelector((state: AppStore) => ({
9     loading: state.app.isLoading,
10    isInitial: state.app.isInitial,
11    forecast: state.weather.ExtendedForecastHourlyData,
12  }));
13
14   if (isInitial) return <</>;
15
16   return (
17     <ForecastContainer>
18       <SectionTitle>Vremea în următoarele ore</SectionTitle>
19       <ForecastItems>
20         {forecast.map((item, i) => {
21           return <ForecastItem key={i} value={item} />;
22         })}
23       </ForecastItems>
24     </ForecastContainer>
25   );
26 };
27
28 export default Forecast;
```

```
WeatherApp > weatherapplication > src > components > Forecast > TS ForecastItem.tsx > ...
4 import { ExtendedForecastHourlyData } from "../../api/types";
5
6 interface ForecastItemProps {
7   key: number;
8   value: ExtendedForecastHourlyData;
9 }
10
11 const ForecastItem: React.FC<ForecastItemProps> = (props) => {
12   const data = props.value;
13
14   const formattedDate = new Date(data.dt_txt).toLocaleDateString("ro-RO", {
15     day: "numeric",
16     month: "long",
17     hour: "numeric",
18     minute: "numeric",
19   });
20
21   return (
22     <ForecastItemContainer>
23       <h3>{formattedDate}</h3>
24       <img
25         src={`http://openweathermap.org/img/wn/${data.weather[0].icon}@2x.png`}
26         alt="Weather icon"
27       />
28       <span>
29         <Temperature value={data.main.temp_max} />
30         <sup>°deg</sup></sup>
31         <small>/</small>
32         <Temperature value={data.main.temp_min} />
33         <sup>°deg</sup></sup>
34       </span>
35     </ForecastItemContainer>
36   );
37 };
38
39 export default ForecastItem;
```

Figura nr. 2.6 Forecast.tsx și ForecastItem.tsx

2.3 NewsPage.tsx

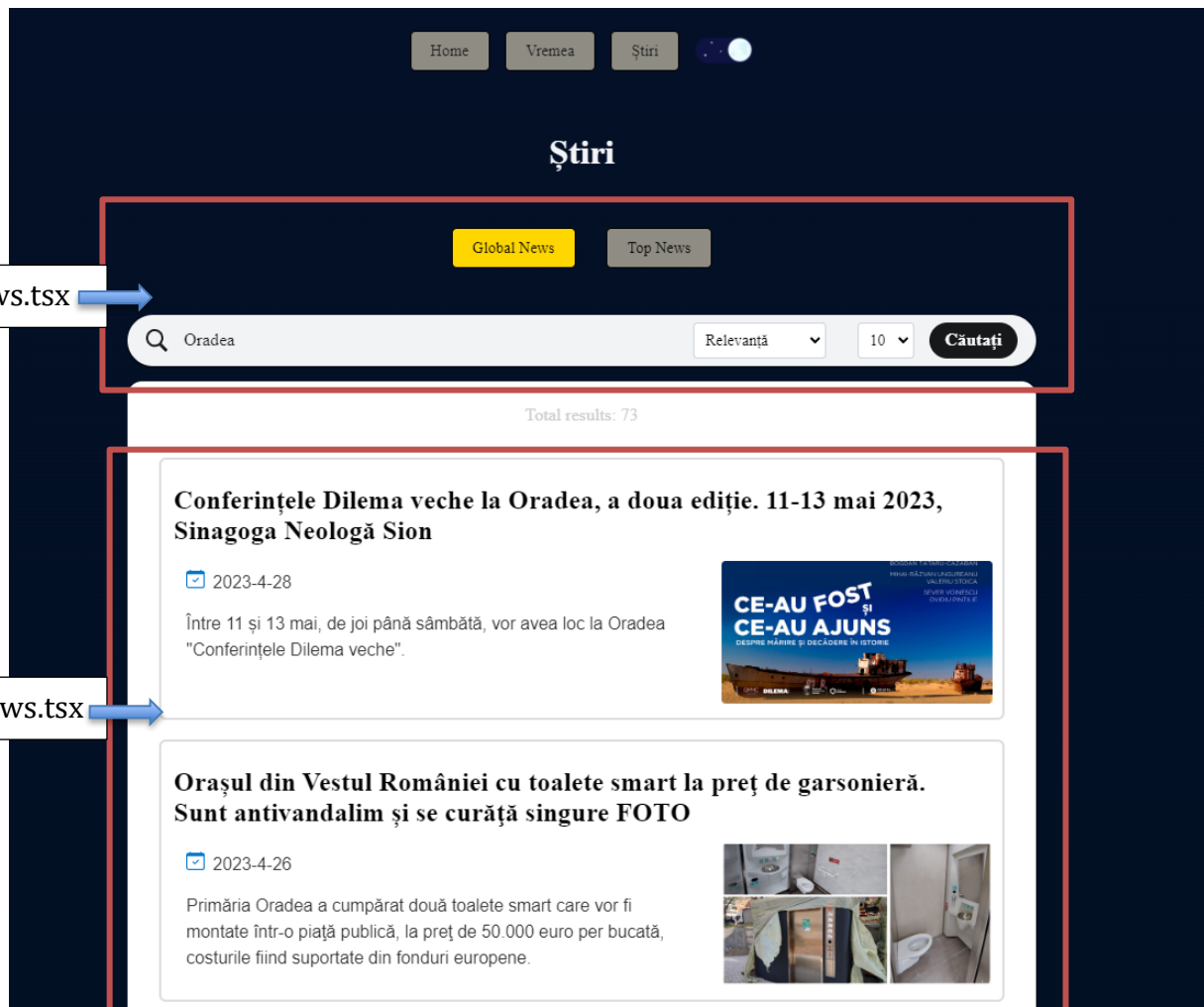


Figura nr. 2.7 NewsPage.tsx

NewsPage reprezintă pagina de știri a aplicației web. Aceasta include un antet, o componentă pentru căutarea de știri și o componentă care afișează știrile curente. În plus, componenta utilizează un spinner pentru a indica încărcarea datelor și stochează starea de încărcare în store-ul Redux pentru a fi accesibilă altor componente.

```

1 import React, { useEffect } from "react";
2 import { useDispatch, useSelector } from "react-redux";
3 import Header from "../components/Header/Header";
4 import { AppState } from "../store/store";
5
6 import CurrentNews from "../components/News/CurrentNews";
7 import { setIsLoading } from "../store/reducers/appReducer";
8 import Spinner from "../components/ui/Spinner/Spinner";
9 import SearchNews from "../components/Search/SearchNews/SearchNews";
10
11 const NewsPage = () => {
12
13   const dispatch = useDispatch();
14   const { isLoading } = useSelector((state: AppState) => ({
15     isLoading: state.app.isLoading,
16   }));
17
18   useEffect(() => {
19     dispatch(setIsLoading(true));
20     setTimeout(() => dispatch(setIsLoading(false)), 0);
21   }, []);
22
23   return (
24     <>
25       {isLoading && <Spinner />}
26       <Header titleHeader="Știri" />
27       <SearchNews />
28       <CurrentNews />
29     </>
30   );
31 };
32
33 export default NewsPage;
34

```

Figura nr. 2.7 Code NewsPage.tsx

SearchNews.tsx este o componentă React care permite utilizatorului să caute articole de știri după subiect, țară și opțiuni de sortare. Utilizatorul poate selecta numărul de articole pe pagină și să caute știri globale sau știri dintr-o anumită țară. Componenta trimite o acțiune `fetchNews` pentru a prelua articole dintr-un API.

Componenta definește patru variabile de stare folosind hook-ul `useState`: `subiect`, `sortBy`, `pageSize` și `country`. De asemenea, definește o variabilă de stare `selectMethod` (figura nr. 2.9), care este utilizată pentru a determina dacă să preia știri globale sau știri dintr-o anumită țară. Componenta definește trei componente funcționale imbricate: `Select`, `SelectNumber` și `SelectCountry`. Aceste componente redă meniuri derulante care permit utilizatorului să selecteze opțiunile de sortare, numărul de articole pe pagină și țara.

Componenta folosește, de asemenea, componenta `DebounceInput` din biblioteca `react-debounce-input` pentru a elimina intrarea utilizatorului în câmpul de intrare de căutare. Acest lucru ajută la reducerea solicitărilor API inutile.

Componenta definește o funcție `onSearchInputChanged` care actualizează variabila de stare a subiectului atunci când utilizatorul introduce în câmpul de introducere a căutării. De

asemenea, definește o funcție `onGetNewsBtnClick` care trimite acțiunea `fetchNews` atunci când utilizatorul face click pe butonul „Căutați”.

```

20 const [sortBy, setSortBy] = useState("relevancy");
21 const [pageSize, setPageSize] = useState(10);
22 const [country, setCountry] = useState("");
23 const [selectedMethod, setSelectedMethod] = useState("Global News");
24
25 const onSearchInputChanged = (e: React.ChangeEvent) => {
26   setSelectedMethod(e.target.value);
27 };
28
29 const onGetNewsBtnClick = (e: React.FormEvent<HTMLFormElement>) => {
30   e.preventDefault(); // prevent form submission
31   dispatch(
32     fetchNews({
33       // as unknown as PayloadAction<string | any, any>
34     })
35   );
36 };
37
38 type SelectProps = {
39   value: string;
40   onChange: (e: React.ChangeEvent<HTMLSelectElement>) => void;
41 };
42
43 const Select = ({ value, onChange }: SelectProps) => {
44   return (
45     <select
46       value={value}
47       onChange={onChange}
48     />
49   );
50 };
51
52 const SelectNumber = ({ value, onChange }: SelectProps) => {
53   return (
54     <input
55       type="text"
56       value={value}
57       onChange={onChange}
58     />
59   );
60 };
61
62 const SelectCountry = ({ value, onChange }: SelectProps) => {
63   return (
64     <select
65       value={value}
66       onChange={onChange}
67     />
68   );
69 };
70
71 return (
72   <div className="SearchNewsContainer">
73     <div>
74       <MethodButtons
75         selectedMethod={selectedMethod}
76         setSelectedMethod={setSelectedMethod}
77       />
78       <SearchForm
79         onSearchInputChanged={onSearchInputChanged}
80         onGetNewsBtnClick={onGetNewsBtnClick}
81       />
82     </div>
83   </div>
84 );

```

Figura nr. 2.8 SearchNews.tsx

```

1 import React from "react";
2 import { ActiveButton, ButtonContainer, ButtonsContainer } from "../styled";
3
4
5 const MethodButtons = ({
6   selectedMethod,
7   setSelectedMethod,
8 }) => {
9   return (
10     <ButtonsContainer>
11       {selectedMethod === "Global News" ? (
12         <ActiveButton
13           onClick={() => setSelectedMethod("Global News")}
14         />
15       ) : (
16         <ButtonContainer
17           onClick={() => setSelectedMethod("Global News")}
18         />
19       )}
20       {selectedMethod === "Top News" ? (
21         <ActiveButton
22           onClick={() => setSelectedMethod("Top News")}
23         />
24       ) : (
25         <ButtonContainer
26           onClick={() => setSelectedMethod("Top News")}
27         />
28       )}
29     </ButtonsContainer>
30   );
31 };
32
33 export default MethodButtons;

```

Figura nr. 2.9 MethodButtons.tsx

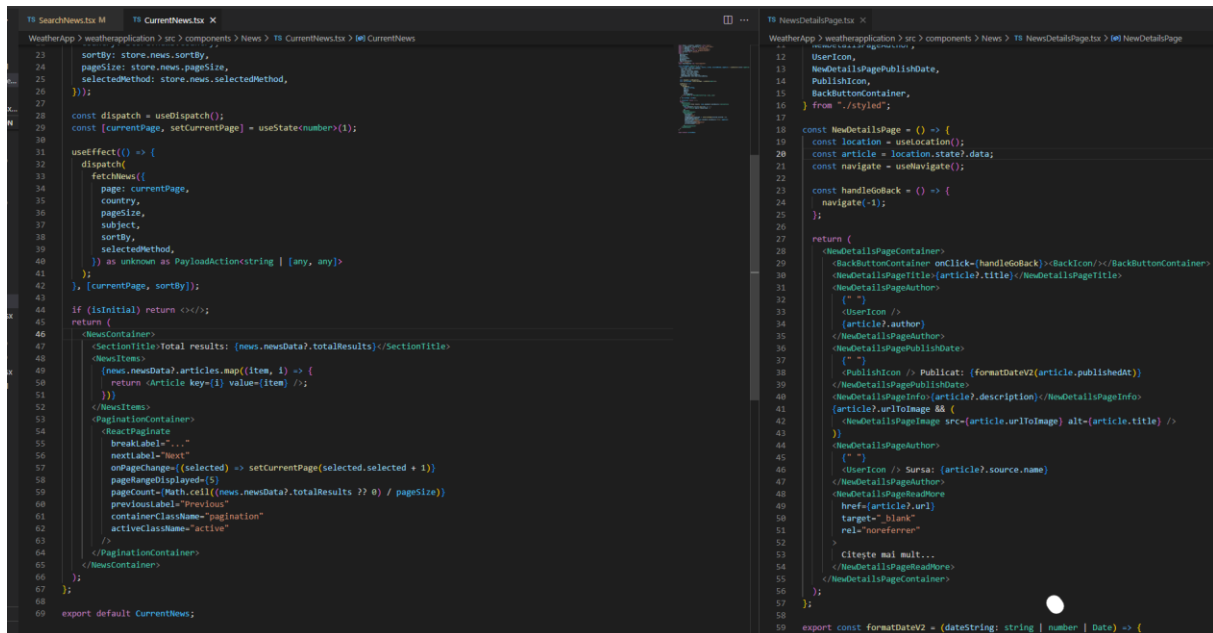


Figura nr. 2.10 CurrentNews.tsx și NewsDetailsPage.tsx

CurrentNews afișează știrile actuale, preluate dintr-un “magazin” de stocare a datelor numit “store”. Componenta utilizează două hook-uri: “useState” și “useEffect”.

Componenta primește date din store prin intermediul selectorului “useSelector”, care extrage valorile relevante pentru subiectul, țara, metoda de sortare și numărul de elemente pe pagină, precum și starea inițială a aplicației.

În componenta CurrentNews, este definită o variabilă “dispatch” prin intermediul căreia sunt apelate acțiuni Redux. În plus, se utilizează o paginare React, implementată prin intermediul pachetului “ReactPaginate”.

În cadrul hook-ului “useEffect”, se face o cerere HTTP pentru a prelua știrile curente în funcție de parametrii specificați (pagina curentă, țara, subiectul, metoda de sortare și numărul de elemente pe pagină). Dacă starea inițială a aplicației este adevărată, nu se afișează nimic.

În caz contrar, se afișează componenta NewsContainer, care conține un titlu al numărului total de rezultate și o listă cu elementele “Article”, care reprezintă articolele de știri propriu-zise.

Codul din figura nr. 2.11 reprezintă o pagină cu detaliile unei știri selectate. Aceasta afișează informații precum titlul, autorul, data publicării și descrierea știrii, împreună cu o imagine (dacă există) și un link către sursa originală a știrii. De asemenea, aceasta oferă o funcționalitate de navigare înapoi la pagina anterioară prin intermediul unui buton cu o săgeată și utilizează biblioteca React Router pentru a gestiona această funcționalitate. Funcția “formatDateV2” este utilizată pentru a formata data într-un format ușor de citit.



Conferințele Dilema veche la Oradea, a doua ediție. 11-13 mai 2023, Sinagoga Neologă Sion

ADH

Publicat: 04/28/2023 16:26

Între 11 și 13 mai, de joi până sâmbătă, vor avea loc la Oradea "Conferințele Dilema veche".



Sursa: Adevarul.ro

[Citește mai mult...](#)

Figura nr. 2.11 NewsDetailsPage.tsx

2.3 Store și Api

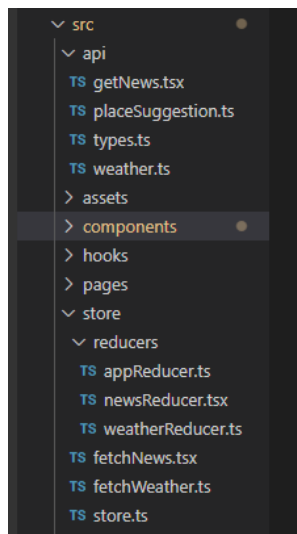


Figura nr. 2.12 Store

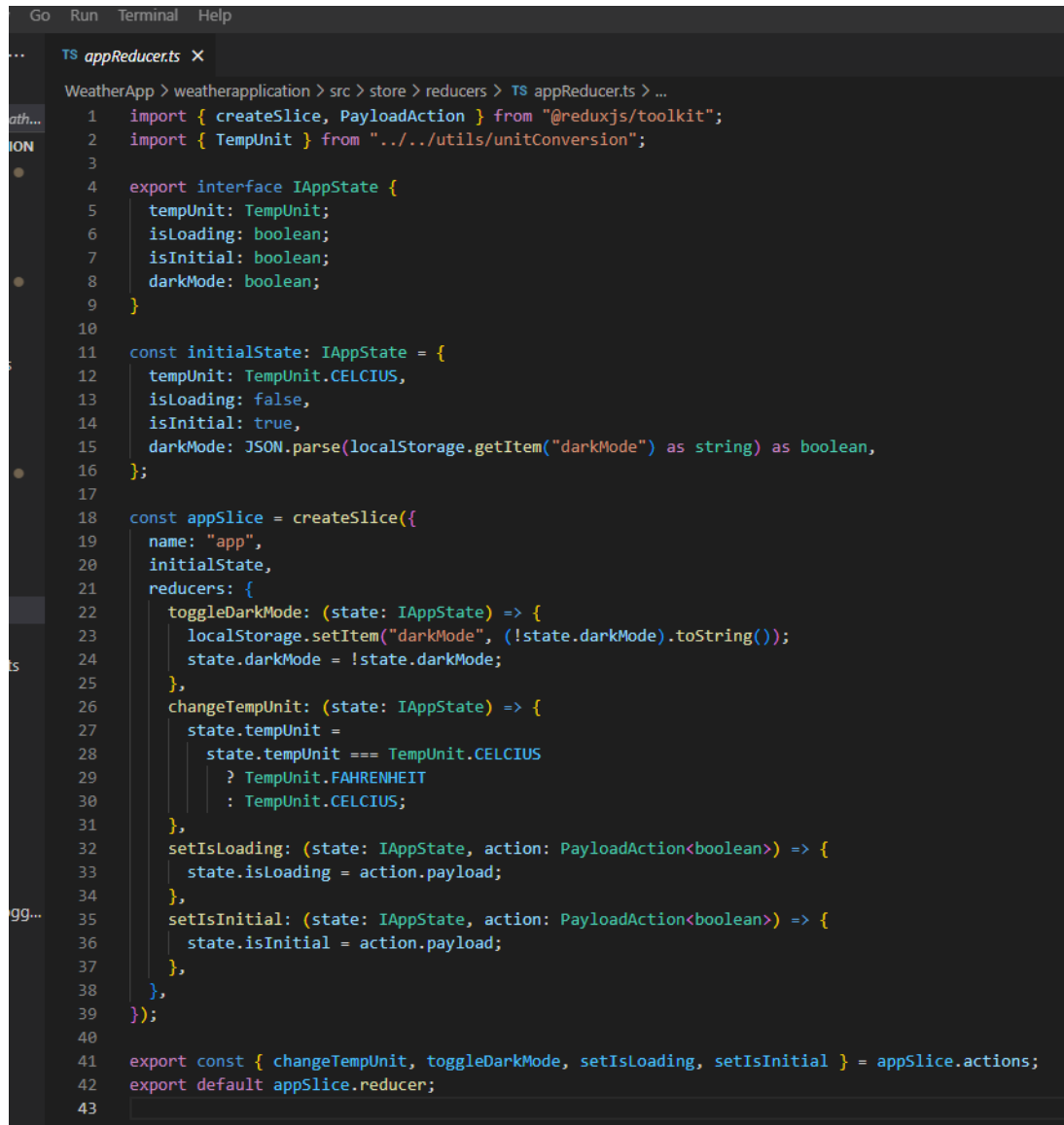
Store.ts este compomenta de configurare al magazinului de stocare Redux din aplicație. Magazinul de stocare Redux este o bibliotecă utilizată în aplicațiile React pentru a gestiona starea aplicației și a permite comunicarea între componente.

În acest fișier, se importă funcția “configureStore” din pachetul “@reduxjs/toolkit”. Această funcție este utilizată pentru a crea un magazin de stocare Redux. Funcția “configureStore” primește un obiect “reducer”. Reducerii sunt definiți ca “appReducer”, “weatherReducer” și “newsReducer”. Acești reduceri sunt responsabili de gestionarea diferitelor părți ale stării aplicației, cum ar fi starea generală a aplicației, vremea și știrile.

```
View Go Run Terminal Help
... TS store.ts X
WeatherApp > weatherapplication > src > store > TS store.ts > ...
1 import { configureStore } from '@reduxjs/toolkit';
2 import appReducer from './reducers/appReducer';
3 import weatherReducer from './reducers/weatherReducer';
4 import newsReducer from './reducers/newsReducer';
5
6 export const store = configureStore({
7   reducer: {
8     app: appReducer,
9     weather: weatherReducer,
10    news: newsReducer,
11  },
12 });
13
14 export type AppState = ReturnType<typeof store.getState>;
15
16
```

Figura nr. 2.12 Store.ts

AppReducer este responsabil pentru definirea logicii de gestionare a stării legate de starea globală a aplicației folosind Redux Toolkit. Obiectul “initialState” definește valorile implicite pentru proprietățile de stare a aplicației, inclusiv “tempUnit”, “isLoading”, “isInitial” și “darkMode”.



```
Go Run Terminal Help
TS appReducer.ts x
WeatherApp > weatherapplication > src > store > reducers > TS appReducer.ts > ...
1 import { createSlice, PayloadAction } from "@reduxjs/toolkit";
2 import { TempUnit } from "../../utils/unitConversion";
3
4 export interface IAppState {
5   tempUnit: TempUnit;
6   isLoading: boolean;
7   isInitial: boolean;
8   darkMode: boolean;
9 }
10
11 const initialState: IAppState = {
12   tempUnit: TempUnit.CELCIUS,
13   isLoading: false,
14   isInitial: true,
15   darkMode: JSON.parse(localStorage.getItem("darkMode") as string) as boolean,
16 };
17
18 const appSlice = createSlice({
19   name: "app",
20   initialState,
21   reducers: {
22     toggleDarkMode: (state: IAppState) => {
23       localStorage.setItem("darkMode", (!state.darkMode).toString());
24       state.darkMode = !state.darkMode;
25     },
26     changeTempUnit: (state: IAppState) => {
27       state.tempUnit =
28         state.tempUnit === TempUnit.CELCIUS
29           ? TempUnit.FAHRENHEIT
30           : TempUnit.CELCIUS;
31     },
32     setIsLoading: (state: IAppState, action: PayloadAction<boolean>) => {
33       state.isLoading = action.payload;
34     },
35     setIsInitial: (state: IAppState, action: PayloadAction<boolean>) => {
36       state.isInitial = action.payload;
37     },
38   },
39 });
40
41 export const { changeTempUnit, toggleDarkMode, setIsLoading, setIsInitial } = appSlice.actions;
42 export default appSlice.reducer;
43
```

Figura nr. 2.13 appReducer.ts

Codul din figura nr 2.14 afișează vremea pentru un anumit oraș. Codul definește o interfață `WeatherState` care reprezintă forma state-ului. Starea inițială este definită ca un obiect cu trei proprietăți: `“weatherData”`, `“ExtendedForecastHourlyData”` și `“isError”`.

`“weatherSlice”` este definit folosind `“createSlice”` din Redux Toolkit. Acesta definește doi „extraReducers” pentru a gestiona acțiunea `„fetchWeather”`: unul pentru când acțiunea este îndeplinită și altul pentru când este respinsă. Când acțiunea este îndeplinită, apelează `“transformWeatherData”` pentru a transforma datele de răspuns și pentru a actualiza starea. Când acțiunea este respinsă, setează indicatorul `“isError”` în stare la `“true”`.

Acțiunea `“fetchWeather”` este definită folosind `“createAsyncThunk”`. Este nevoie de un parametru `“city”` și `“setIsLoading”` pentru a indica faptul că cererea este în curs. Apoi apelează `“fetchWeatherData”` și `“fetchExtendedForecastData”` pentru a prelua datele meteo și, respectiv, datele de prognoză extinse. Dacă cererea are succes, trimite `“setIsInitial”` pentru a indica faptul că aplicația nu mai este în starea inițială și returnează datele răspunsului. Dacă cererea eșuează, returnează o promisiune respinsă cu un mesaj de eroare.

Funcția `“transformWeatherData”` preia datele răspunsului și le transformă într-un obiect cu două proprietăți: `“weather”` și `“forecast”`. Folosește `“kelvinToCelcius”` pentru a converti temperatura de la Kelvin la Celsius și actualizează obiectele `“vremei”` și `“prognoză”` cu datele transformate.

Funcțiile `“fetchWeatherData”` și `“fetchExtendedForecastData”` sunt utilizate pentru a prelua datele meteo și, respectiv, datele de prognoză extinsă. Aceste funcții construiesc adresa URL pentru cererea API și folosesc `“fetch”` pentru a face cererea.

```

TS weatherReducer.ts X
WeatherApp > weatherapplication > src > store > reducers > TS weatherReducer.ts > initialState >
11 const initialState: WeatherState = {
12   weatherData: {
13     main: {
14       feels_like: 0,
15       humidity: 0,
16       pressure: 0,
17       temp: 0,
18       temp_max: 0,
19       temp_min: 0,
20     },
21     name: '',
22     sys: {
23       country: '',
24       sunrise: 0,
25       sunset: 0,
26     },
27   },
28   weather: {
29     id: 200,
30     main: '',
31     description: '',
32     icon: '',
33   },
34   wind: {
35     deg: 0,
36     speed: 0,
37   },
38   ExtendedForecastHourlyData: [],
39   isError: false,
40 };
41
42 const weatherSlice = createSlice({
43   name: 'weather',
44   initialState,
45   reducers: {},
46   extraReducers: (builder) => {
47     builder
48       .addCase(fetchWeather.fulfilled, (state, action) => {
49         const res = transformWeatherData(action.payload);
50         state.weatherData = res.weather;
51         state.ExtendedForecastHourlyData = res.forecast;
52       })
53       .addCase(fetchWeather.rejected, (state, action) => {
54         state.isError = true;
55       });
56   },
57 });
58

```

Figura nr. 2.14 weatherReducer.ts

```

export const fetchWeather = createAsyncThunk(
  'weather/fetchWeather',
  async (city: string | { lat: number; lng: number }, { dispatch, rejectWithValue }) => {

    dispatch(setIsLoading(true));

    try {
      const res = await Promise.all([fetchWeatherData(city), fetchExtendedForecastData(city)]);
      setTimeout(() => dispatch(setIsLoading(false)), 0);
      if (res[0].cod === 200) {
        dispatch(setIsInitial(false));
        return res;
      }
      return rejectWithValue(res[0].message);
    } catch {
      setTimeout(() => dispatch(setIsLoading(false)), 0);
      return rejectWithValue('Error');
    }
  }
);

export const transformWeatherData = (res: any): { weather: WeatherData; forecast: ExtendedFore

const weather = res[0] as WeatherData;
const forecast: ExtendedForecastHourlyData[] = [];

weather.weather.id = res[0].weather[0].id;
weather.weather.icon = res[0].weather[0].icon;
weather.weather.description = res[0].weather[0].description;
weather.weather.main = res[0].weather[0].main;
weather.main = {
  ...weather.main,
  temp: kelvinToCelcius(weather.main.temp),
  feels_like: kelvinToCelcius(weather.main.feels_like),
  temp_max: kelvinToCelcius(weather.main.temp_max),
  temp_min: kelvinToCelcius(weather.main.temp_min),
};
weather.wind.speed = Math.round(weather.wind.speed * 3.6);
weather.wind.deg = weather.wind.deg;

res[1].list.forEach((i: any, index: number) => {
  forecast.push({
    dt: i.dt,
    main: {
      ...i.main,
      temp: kelvinToCelcius(i.main.temp),
      feels_like: kelvinToCelcius(i.main.feels_like),
      temp_min: kelvinToCelcius(i.main.temp_min),
    },
  });
});

```

Figura nr. 2.15 fetchWeather.ts

```

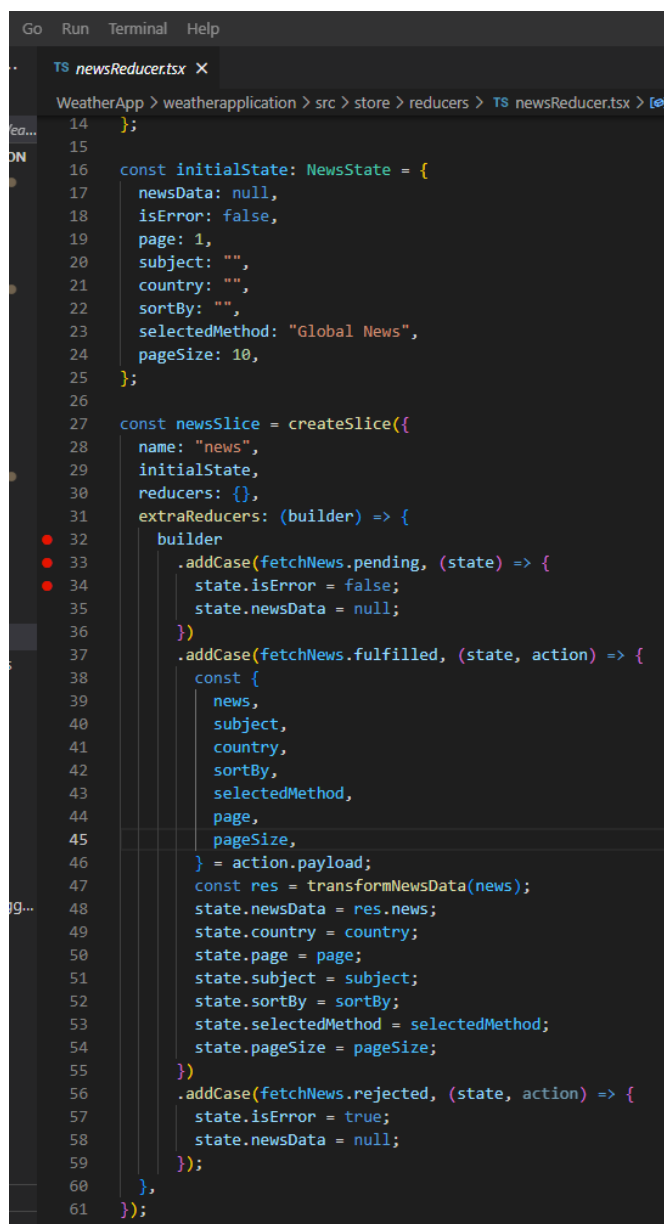
weatherReducer.ts - Weather React Application - Visual Studio Code

TS weather.ts X
WeatherApp > weatherapplication > src > api > TS weather.ts > ...
1  const baseUrl = "https://api.openweathermap.org/data/2.5";
2
3  export const fetchWeatherData = async (city: string | { lat: number; lng: number }) => {
4
5    const apiKey = process.env.REACT_APP_REACT_WEATHER_API_KEY ?? "default-api-key";
6    let url = `${baseUrl}/weather?q=${city}&appid=${apiKey}`;
7
8    if (typeof city === "object") {
9      url = `${baseUrl}/weather?lat=${city.lat}&lon=${city.lng}&appid=${apiKey}`;
10   }
11   return await (await fetch(url)).json();
12 };
13
14 export const fetchExtendedForecastData = async (city: string | { lat: number; lng: number }) => {
15
16   const apiKey = process.env.REACT_APP_REACT_WEATHER_API_KEY ?? "default-api-key";
17   let url = `${baseUrl}/forecast?q=${city}&cnt=6&appid=${apiKey}`;
18
19   if (typeof city === "object") {
20     url = `${baseUrl}/forecast?lat=${city.lat}&lon=${city.lng}&cnt=6&appid=${apiKey}`;
21   }
22   return await (await fetch(url)).json();
23 };
24

```

Figura nr. 2.16 weather.ts

NewsReducer.tsx definește o porțiune a stării de aplicație în cadrul store Redux.



```
14 };
15
16 const initialState: NewsState = {
17   newsData: null,
18   isError: false,
19   page: 1,
20   subject: "",
21   country: "",
22   sortBy: "",
23   selectedMethod: "Global News",
24   pageSize: 10,
25 };
26
27 const newsSlice = createSlice({
28   name: "news",
29   initialState,
30   reducers: {},
31   extraReducers: (builder) => {
32     builder
33       .addCase(fetchNews.pending, (state) => {
34         state.isError = false;
35         state.newsData = null;
36       })
37       .addCase(fetchNews.fulfilled, (state, action) => {
38         const {
39           news,
40           subject,
41           country,
42           sortBy,
43           selectedMethod,
44           page,
45           pageSize,
46         } = action.payload;
47         const res = transformNewsData(news);
48         state.newsData = res.news;
49         state.country = country;
50         state.page = page;
51         state.subject = subject;
52         state.sortBy = sortBy;
53         state.selectedMethod = selectedMethod;
54         state.pageSize = pageSize;
55       })
56       .addCase(fetchNews.rejected, (state, action) => {
57         state.isError = true;
58         state.newsData = null;
59       });
60   },
61 });
```

Figura nr. 2.17 newsReducer.tsx

Componenta utilizează funcția “createSlice” din toolkit-ul Redux pentru a defini slice-ul, starea inițială și reducerii care gestionează schimbările de stare generate de acțiunile “fetchNews” (figura nr. 2.18) (o acțiune asincronă care descarcă știrile din API și le transformă într-un format mai ușor de utilizat). Acești reducerii actualizează starea slice-ului cu datele descărcate și cu setările corespunzătoare, și setează variabila “isError” în cazul unei erori în timpul descărcării datelor.

```

TS fetchNews.tsx x
WeatherApp > weatherapplication > src > store > TS fetchNews.tsx > [0] fetchNews > createAsyncThunk("news/fetchNews") callback
1 import { createAsyncThunk } from "@reduxjs/toolkit";
2 import { NewsData } from "../api/types";
3 import { getNews } from "../api/getNews";
4 import { setIsInitial, setIsLoading } from "../reducers/appReducer";
5
6 export const fetchNews = createAsyncThunk(
7   "news/fetchNews",
8   async (
9     {
10       subject,
11       country,
12       sortBy,
13       selectedMethod,
14       page,
15       pageSize,
16     }
17   ): { ...
23   },
24   { dispatch, rejectWithValue }
25 ) => {
26   dispatch(setIsLoading(true));
27
28   try {
29     const res = await getNews(
36     );
37     setTimeout(() => dispatch(setIsLoading(false)), 0);
38     const { news } = transformNewsData(res);
39     dispatch(setIsInitial(false));
40     return { news, subject, country, sortBy, selectedMethod, page, pageSize };
41   } catch (error) {
42     setTimeout(() => dispatch(setIsLoading(false)), 0);
43     return rejectWithValue(
44       (error as { response: { data: { message: string } } }).response.data
45       .message
46     );
47   }
48 }
49 );
50
51 export const transformNewsData = (res: any): { news: NewsData } => {
52   const news: NewsData = {
53     status: res.status,
54     totalResults: res.totalResults,
55     articles: res.articles.map((article: any) => ({
56       source: {
57         id: article.source.id,
58         name: article.source.name,
59       },
60       author: article.author,
61       title: article.title

```

Figura nr. 2.18 fetchNews.tsx

Funcția “getNews” (Figura nr. 2.19) din cod face apelul la API și primește datele de la server. Apoi, aceste date sunt prelucrate în funcția “transformNewsData”, care transformă datele primite în formatul specificat de obiectul “NewsData”. Funcția “fetchNews” este o funcție asincronă creată folosind “createAsyncThunk” din Redux Toolkit și apelează “getNews” pentru a obține datele. Dacă apelul este reușit, funcția “setIsInitial” este apelată cu parametrul “false” și datele obținute sunt returnate împreună cu alte informații relevante. Dacă

apelul eșuează, funcția “setIsLoading” este apelată cu parametrul “false”, iar eroarea este pasată prin “rejectWithValue” pentru a fi gestionată ulterior.



```
TS getNews.tsx X
WeatherApp > weatherapplication > src > api > TS getNews.tsx > [?] getNews
1 export const getNews = async (
2   search: string,
3   country: string,
4   sortBy: string,
5   selectedMethod: string,
6   page: number,
7   pageSize: number
8 ) => {
9   let method = "";
10  const searchSubject = search || "volvo";
11  const baseUrl = "https://newsapi.org/v2/";
12  const globalOp = "everything?q=";
13  const topOp = "top-headlines?country=";
14  const apiKey = process.env.REACT_APP_REACT_NEWS_API_KEY ?? "default-api-key";
15
16  switch (selectedMethod) {
17    case "Top News":
18      method = topOp + country;
19      break;
20    case "Global News":
21      method = globalOp + searchSubject;
22      break;
23    default:
24      method = globalOp;
25      break;
26  }
27
28  try {
29    const date = getFormattedDateXMonthsAgo();
30    const headers: HeadersInit = {
31      "X-API-Key": apiKey,
32    };
33
34    const response = await fetch(
35      `${baseUrl}${method}&sortBy=${sortBy}&from=${date}&page=${page}&pageSize=${pageSize}`,
36      // `http://localhost:3002/newsapi.org/v2/everything`,
37      {
38        method: "GET",
39        headers,
40        redirect: "follow",
41      }
42    );
43    const data = await response.json();
44    console.log(data);
45    return data;
46  } catch (error) {
47    console.error("Error fetching cities:", error);
48    throw error;
49  }
```

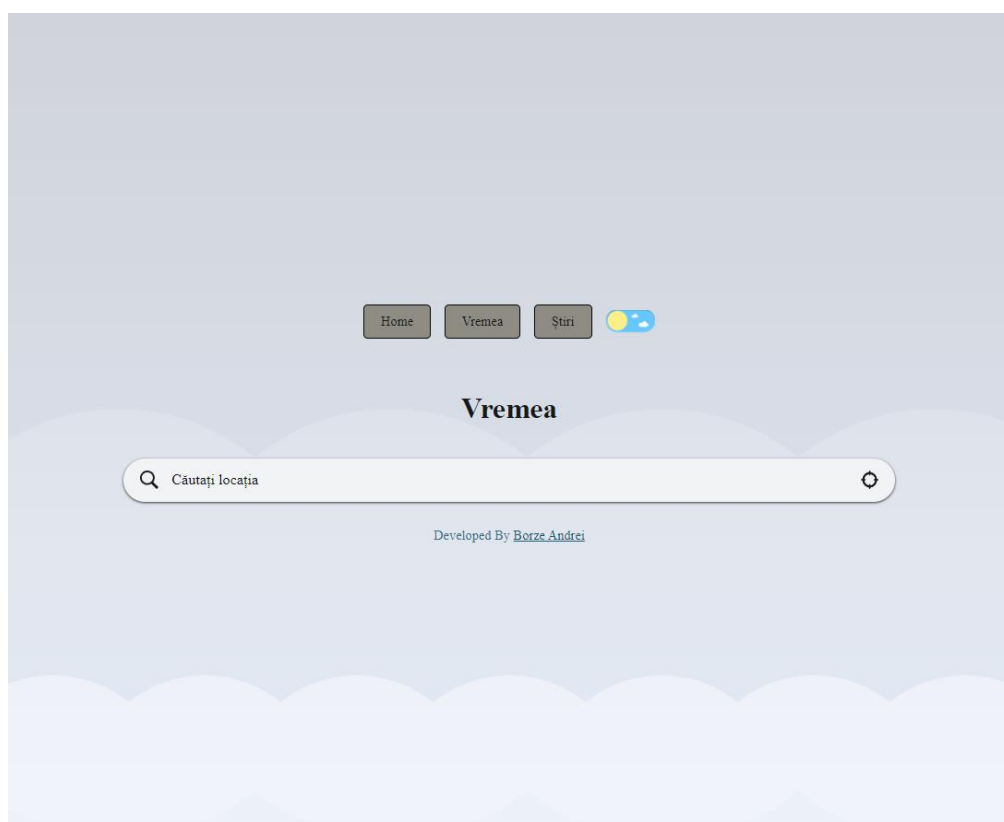
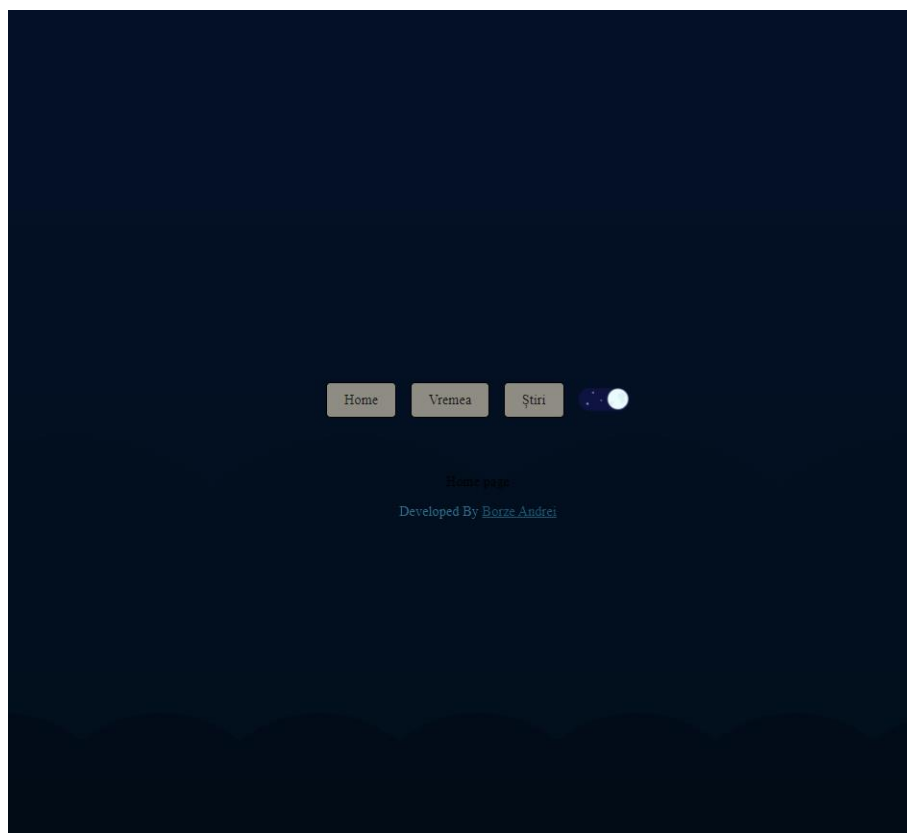
Figura nr. 2.19 getNews.tsx

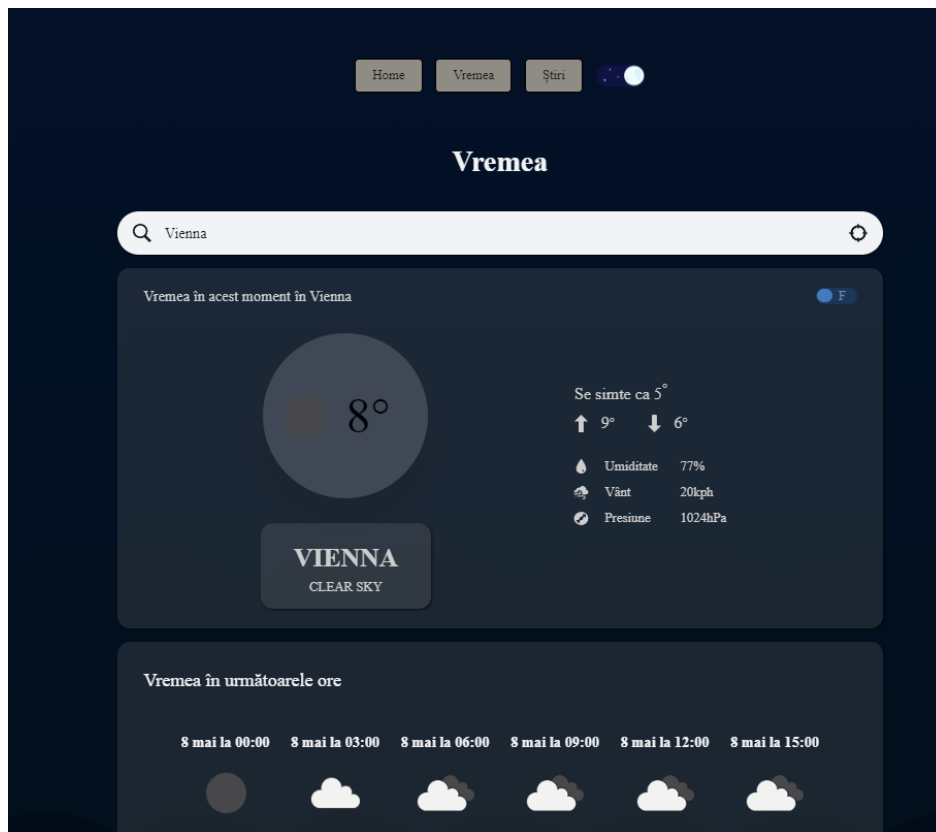
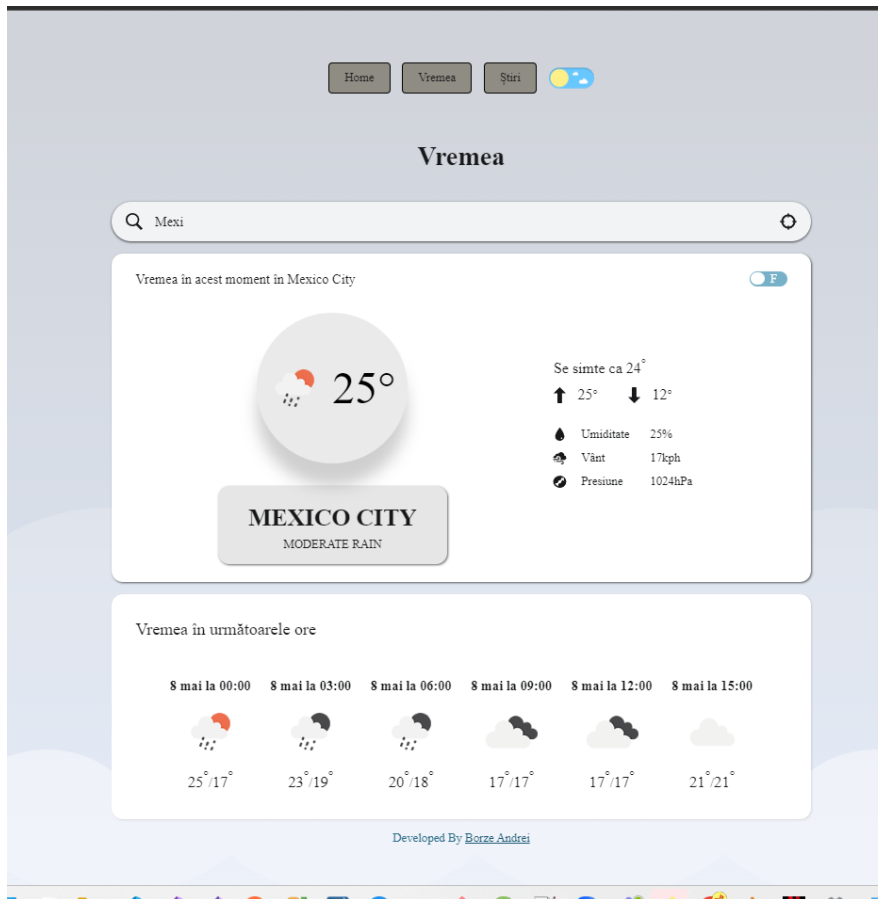
CONCLUZII

În concluzie, utilizarea Redux împreună cu React poate aduce multiple beneficii dezvoltatorilor care lucrează la aplicații complexe, care au nevoie de o gestionare eficientă a stării aplicației. Redux poate ajuta la organizarea mai bună a codului și a logicii aplicației, menținerea unei stări globale consistente și predictibile, ușurința în testare, dar și permite dezvoltatorilor să scrie cod mai modular și să reutilizeze mai multe componente.

În plus, Redux poate fi integrat cu ușurință în aplicații React și poate fi folosit alături de alte biblioteci sau tehnologii. Însă, utilizarea Redux poate aduce și unele dezavantaje, cum ar fi o curba de învățare mai abruptă pentru dezvoltatorii noi sau o posibilă creștere a complexității aplicației. Prin urmare, decizia de a utiliza sau nu Redux depinde de specificul aplicației și de nevoile dezvoltatorilor implicați.

IMAGINI CU APLICAȚIA





2023-5-2

Apple and Google have announced a partnership to tackle the issue of unwanted tracking through the likes of AirTags and Tile devices. The companies have proposed industry standards "to help combat the misuse of Bluetooth location-tracking devices for unwanted...



The Morning After: Apple and Google team up to combat Bluetooth tracker stalking

2023-5-3

Apple and Google have announced a partnership to tackle unwanted tracking through the likes of AirTags and Tile devices. The companies have proposed industry standards "to help combat the misuse of Bluetooth location-tracking devices for unwanted tracking."Ap...



Previous 1 2 3 4 5 ... 1217 1218 1219 Next

Developed By [Borze Andrei](#)

Home Vremea Știri

Știri

Global News

Top News

Q SAMSUNG

Popularitate

10

Cautăți

Total results: 12189

Samsung's News app brings daily headlines and podcasts to Galaxy devices

2023-4-18

Samsung is replacing its Free app on Galaxy devices with a client that's more focused, if also a little familiar. The company is launching a beta News app that, somewhat like its Apple equivalent, concentrates on top headlines (here organized into morning and...



Samsung's Galaxy Watch 5 Pro is 30 percent off right now

2023-5-3

The Samsung Galaxy Watch 5 Pro woke up and chose (relative) affordability today. It's available on Amazon with a 30 percent discount dropping its cost from \$450 to \$315.49. This offer is the lowest price we've seen for the Galaxy Watch 5 Pro so far, dipping





The Morning After: Apple and Google team up to combat Bluetooth tracker stalking

Mat Smith

Publicat: 05/03/2023 14:15

Apple and Google have announced a partnership to tackle unwanted tracking through the likes of AirTags and Tile devices. The companies have proposed industry standards "to help combat the misuse of Bluetooth location-tracking devices for unwanted tracking." Ap...



Sursa: Engadget

[Citește mai mult...](#)

Developed By: Bogdan Andrei