***Rezumat funcționalitate aplicație***

1. Aplicația WEB

* **App.js**
* Componenta părinte a aplicației.
* Este compusă din componenta TopBar (creată de noi) și dintr-o componentă Switch (preluată din React-Router).
* TopBar-ul este randat tot timpul (în partea superioară a paginii).
* Switch-ul conține mai multe rute și face redirecționarea către acestea în funcție de opțiunea aleasă de administrator (în cazul în care acesta este conectat). În cazul în care nu este conectat nici un admin, vom fi redirecționați automat către ruta de autentificare și va fi randată componenta Login.
* **ForgotPassword.js**
* Componentă folosită pentru resetarea parolei administratorului WEB

1. Interfața-utilizator

* Este formată dintr-un container principal în care se află un formular, ambele fiind componente preluate din React-Bootstrap. Formularul conține componente definite de către noi.
* Formularul este format din mai multe componente create de noi: CustomAlert, CustomInput, CustomButton și CustomText.
* CustomAlert-ul este randat condiționat (este vizibil doar dacă variabila „alertIsVisible” este adevărată) și are mai multe proprietăți:
* message (conține mesajul de eroare sau de succes)
* onClose (conține o metodă anonimă care setează variabila „alertIsVisible” pe fals)
* variant (conține tipul alertei: de eroare sau de succes)
* CustomInput-ul are următoarele proprietăți:
* id (pentru identificarea componentei)
* placeholder (este afișat în cazul în care câmpul este gol)
* reference (conține referința câmpului și este folosită pentru golirea textului, pentru preluarea textului introdus etc.)
* type (conține tipul câmpului)
* CustomButton-ul are următoarele proprietăți:
* onClick (conține referința metodei pentru verificarea validității adresei de email și resetarea parolei)
* text (conține textul care trebuie afișat pe ecran)
* CustomText-ul are următoarele proprietăți:
* onClick (conține referința metodei pentru redirecționarea către componenta de autentificare)
* text (conține textul care trebuie afișat pe ecran).

1. Metode & variabile

* Variabilele folosite sunt:
* auth (conține metoda getAuth() a Firebase)
* history (conține referința hook-ului useHistory – pentru accesarea istoricului browser-ului folosit)
* emailRef (conține referința hook-ului useRef – pentru a ține o referință a câmpului pentru email)
* alertIsVisible (este setată inițial pe fals)
* message (este inițial un mesaj gol)
* isSuccessful (este setată inițial pe fals)
* Metodele folosite sunt:
* redirectToLoginPageHandler() – pentru redirecționarea către pagina de autentificare inițială
* resetPasswordHandler(event)
* event.preventDefault() – previne trimiterea formularului după apăsarea butonului pentru resetarea parolei
* variabila enteredEmail conține textul introdus în câmpul pentru adresa de email
* metoda emailIsValid(enteredEmail) este o metodă definită de noi pentru asigurarea că adresa de email introdusă este validă
* în cazul în care adresa de email introdusă este validă, se apelează metoda Firebase sendPasswordResetEmail(auth, enteredEmail) - trimite un email pentru confirmarea resetării parolei
* în cazul în care adresa de email introdusă nu este validă, se va afișa pe ecran un mesaj corespunzător
* **Login.js**
* Componentă folosită pentru autentificarea administratorului WEB

1. Interfața-utilizator

* Este formată dintr-un container principal în care se află un formular, ambele fiind componente preluate din React-Bootstrap. Formularul conține componente definite de către noi.
* Formularul este format din mai multe componente create de noi: CustomAlert, CustomInput (x3), CustomButton și CustomText (x2).
* CustomAlert-ul este randat condiționat (este vizibil doar dacă variabila „errorIsVisible” este adevărată) și are mai multe proprietăți:
* message (conține mesajul de eroare)
* onClose (conține o metodă anonimă care setează variabila „errorIsVisible” pe fals)
* variant (conține tipul alertei: de eroare)
* CustomInput-urile au următoarele proprietăți:
* id (pentru identificarea componentei)
* placeholder (este afișat în cazul în care câmpul este gol)
* reference (conține referința câmpului și este folosită pentru golirea textului, pentru preluarea textului introdus etc.)
* type (conține tipul câmpului)
* CustomButton-ul are următoarele proprietăți:
* onClick (conține referința metodei pentru verificarea validității formularului și autentificarea administratorului)
* text (conține textul care trebuie afișat pe ecran)
* CustomText-ul are următoarele proprietăți:
* onClick (conține referința metodei pentru redirecționarea către componenta de autentificare)
* text (conține textul care trebuie afișat pe ecran).

1. Metode & variabile

* Variabilele folosite sunt:
* auth (conține metoda getAuth() a Firebase)
* dispatch (conține referința hook-ului useDispatch – pentru expedierea unei acțiuni către variabilele din Redux)
* history (conține referința hook-ului useHistory – pentru accesarea istoricului browser-ului folosit)
* emailRef (conține referința hook-ului useRef – pentru a ține o referință a câmpului pentru email)
* passwordRef (conține referința hook-ului useRef – pentru a ține o referință a câmpului pentru parolă)
* rememberMeIsChecked (este setată inițial pe fals – folosită pentru a ține valoarea câmpului rememberMe)
* errorIsVisible (este setată inițial pe fals – folosită pentru a cunoaște dacă eroarea este sau nu vizibilă pe ecran)
* errorMessage (este inițial un mesaj de eroare gol)
* Metodele folosite sunt:
* loginHandler(event)
* event.preventDefault() – previne trimiterea formularului după apăsarea butonului pentru autentificare
* variabilele enteredEmail & enteredPassword conțin textul introdus în câmpurile pentru adresa de email și pentru parolă
* metoda loginIsValid(enteredEmail, enteredPassword) este o metodă definită de noi pentru asigurarea că adresa de email și parola introduse sunt valide
* în cazul în care adresa de email și parola introduse sunt valide, se apelează metoda Firebase signInWithEmailAndPassword(auth, enteredEmail, enteredPassword) – se ocupă de autentificarea administratorului și îi permite acestuia să își acceseze contul în cazul în care și-a confirmat contul prin email; se expediază acțiunea de modificare a administratorului curent în Redux
* în cazul în care adresa de email și parola introduse nu sunt valide, se va afișa pe ecran un mesaj corespunzător
* redirectHandler() – pentru redirecționarea către pagina dorită
* rememberMeHandler(event) – pentru setarea variabilei rememberMeIsChecked și este apelată de fiecare dată când administratorul bifează/debifează căsuța pentru Remember Me
* showError(message) – pentru afișarea unei erori ce conține un anumit mesaj (în funcție de eroarea apărută)
* **Register.js**
* Componentă folosită pentru autentificarea administratorului WEB

1. Interfața-utilizator

* Este formată dintr-un container principal în care se află un formular, ambele fiind componente preluate din React-Bootstrap. Formularul conține componente definite de către noi.
* Formularul este format din mai multe componente create de noi: CustomAlert, CustomInput (x4), CustomButton și CustomText.
* CustomAlert-ul este randat condiționat (este vizibil doar dacă variabila „errorIsVisible” este adevărată) și are mai multe proprietăți:
* message (conține mesajul de eroare)
* onClose (conține o metodă anonimă care setează variabila „errorIsVisible” pe fals)
* variant (conține tipul alertei: de eroare)
* CustomInput-urile au următoarele proprietăți:
* id (pentru identificarea componentei)
* placeholder (este afișat în cazul în care câmpul este gol)
* reference (conține referința câmpului și este folosită pentru golirea textului, pentru preluarea textului introdus etc.)
* type (conține tipul câmpului)
* CustomButton-ul are următoarele proprietăți:
* onClick (conține referința metodei pentru verificarea validității formularului și înregistrarea administratorului)
* text (conține textul care trebuie afișat pe ecran)
* CustomText-ul are următoarele proprietăți:
* onClick (conține referința metodei pentru redirecționarea către componenta de autentificare)
* text (conține textul care trebuie afișat pe ecran).

1. Metode & variabile

* Variabilele folosite sunt:
* history (conține referința hook-ului useHistory – pentru accesarea istoricului browser-ului folosit)
* emailRef (conține referința hook-ului useRef – pentru a ține o referință a câmpului pentru email)
* passwordRef (conține referința hook-ului useRef – pentru a ține o referință a câmpului pentru parolă)
* firstNameRef (conține referința hook-ului useRef – pentru a ține o referință a câmpului pentru prenumele administratorului)
* lastNameRef (conține referința hook-ului useRef – pentru a ține o referință a câmpului pentru numele de familie al administratorului)
* errorIsVisible (este setată inițial pe fals – folosită pentru a cunoaște dacă eroarea este sau nu vizibilă pe ecran)
* errorMessage (este inițial un mesaj de eroare gol)
* Metodele folosite sunt:
* createPersonalInformationPath(personalInformation) – pentru crearea unui nod separat dedicat informațiilor personale ale administratorului
* redirectToLoginPageHandler() – pentru redirecționarea către pagina de autentificare a administratorului
* registerHandler(event)
* event.preventDefault() – previne trimiterea formularului după apăsarea butonului pentru autentificare
* variabilele enteredEmail, enteredPassword, enteredFirstName, enteredLastName conțin textul introdus în câmpurile pentru adresa de email, pentru parolă, pentru numele și prenumele administratorului
* metoda registrationIsValid(enteredEmail, enteredPassword, enteredFirstName, enteredLastName) este o metodă definită de noi pentru asigurarea că formularul este valid
* în cazul în care formularul este valid, se apelează metoda Firebase createUserWithEmailAndPassword(auth, enteredEmail, enteredPassword) – se ocupă de înregistrarea administratorului și îi trimite acestuia confirmarea contului prin email
* în cazul în care formularul nu este valid, se va afișa pe ecran un mesaj corespunzător
* **AllReports.js**
* Componentă folosită pentru randarea listei de rapoarte din baza de date Firebase

1. Interfața-utilizator

* Este formată dintr-un container principal în care se află lista de rapoarte primită ca parametru, precum și un Spinner preluat din React-Bootstrap. Acest Spinner va fi afișat pe ecran și se va roti până când lista de rapoarte va fi randată cu succes. În cazul în care această listă este goală, se va afișa pe ecran un mesaj corespunzător.
* În momentul în care s-a preluat cu succes lista de rapoarte din baza de date Firebase se va mapa lista și va fi iterată pas cu pas, afișând pe ecran câte o componentă definită de noi (ReportItem).
* Componenta ReportItem are următoarele proprietăți:
* key (pentru identificatorul unic al obiectului din listă)
* report (obiectul curent)

1. Metode & variabile

* Variabilele folosite sunt:
* isLoading (este setată inițial pe adevărat – folosită pentru a afișa Spinner-ul pe ecran cât timp este adevărată)
* reportsList (conține lista de rapoarte primită ca parametru)
* Hook-urile folosite sunt:
* useEffect (setează valoarea variabilei isLoading o singură dată: atunci când se randează prima dată componenta)
* **AllUsers.js**
* Componentă folosită pentru randarea listei de utilizatori din baza de date Firebase

1. Interfața-utilizator

* Este formată dintr-un container principal în care se află lista de utilizatori primită ca parametru, precum și un Spinner preluat din React-Bootstrap. Acest Spinner va fi afișat pe ecran și se va roti până când lista de utilizatori va fi randată cu succes. În cazul în care această listă este goală, se va afișa pe ecran un mesaj corespunzător.
* În momentul în care s-a preluat cu succes lista de utilizatori din baza de date Firebase se va mapa lista și va fi iterată pas cu pas, afișând pe ecran câte o componentă definită de noi (UserItem).
* Componenta UserItem are următoarele proprietăți:
* key (pentru identificatorul unic al obiectului din listă)
* user (obiectul curent)

1. Metode & variabile

* Variabilele folosite sunt:
* isLoading (este setată inițial pe adevărat – folosită pentru a afișa Spinner-ul pe ecran cât timp este adevărată)
* usersList (conține lista de utilizatori primită ca parametru)
* Hook-urile folosite sunt:
* useEffect (setează valoarea variabilei isLoading o singură dată: atunci când se randează prima dată componenta)
* **CustomAlert.js**
* Componentă folosită pentru afișarea pe ecran a unui Alert preluat din React-Bootstrap, dar modificat vizual de către noi.
* Este formată dintr-un container principal în care se află Alert-ul și are diferite dimensiuni în funcție de ecranul pe care este randată.
* **CustomButton.js**
* Componentă folosită pentru afișarea unui buton preluat din React-Bootstrap, dar modificat vizual de către noi.
* Este formată dintr-un container principal în care se află butonul și are diferite dimensiuni în funcție de ecranul pe care este randată.
* **CustomInput.js**
* Componentă folosită pentru afișarea unui câmp preluat din React-Bootstrap, dar modificat vizual de către noi.
* Este compusă dintr-un container principal în care se află câmpul. Acest câmp este afișat condiționat în funcție de tipul primit ca parametru (poate fi un simplu Input sau un CheckBox).
* **CustomModal.js**
* Componentă folosită pentru randarea unui Modal preluat din React-Bootstrap, dar modificat vizual de către noi.
* Este formată dintr-un Modal care conține detaliile raportului selectat și le afișează pe ecran. Nota raportului este randată în antetul Modal-ului, restul detaliilor în corpul acestuia, iar butonul de închidere al Modal-ului în subsolul acestuia.
* **CustomNavLink.js**
* Componentă folosită pentru afișarea unui element al meniului. Aceasta are următoarele proprietăți:
* active (variabilă ce ține cont dacă elementul meniului este sau nu activ în momentul de față)
* className (pentru setarea unei anumite clase în funcție de navbarIsExpanded și de active)
* onClick (pentru executarea unei metode onClick primită ca parametru)
* **CustomText.js**
* Este formată dintr-un paragraf HTML ce afișează un text primit ca parametru.
* **Logo.js**
* Componentă folosită pentru randarea siglei aplicației WEB
* Este formată dintr-un container principal în care se află sigla (logo) și numele aplicației.
* Sigla aplicației este preluată din React-Icons.
* **Map.js**
* Componentă folosită pentru afișarea pe hartă a tuturor rapoartelor.

1. Interfața-utilizator

* Este formată dintr-o componentă GoogleMap (importată din API-ul GoogleMaps) ce conține unul sau mai multe Markere pentru locație și un Modal preluat din React-Bootstrap.
* În cazul în care lista ce conține locațiile tuturor rapoartelor nu este goală, aceasta este mapată și iterată pas cu pas, afișându-se câte un Marker pentru fiecare raport ce conține locație.
* Dacă dăm click pe un Marker, vom lua din baza de date detaliile raportului selectat (în funcție de id) și informațiile despre utilizatorul care a realizat raportul și le vom afișa în Modal.

1. Metode & variabile

* Variabilele folosite sunt:
* modalIsVisible (este setată inițial pe fals)
* selectedReport (este inițializată cu un obiect gol)
* selectedReportDateTime (conține data și timpul la care a fost adăugat raportul și este setată inițial pe undefined)
* selectedUserDetails (conține detaliile utilizatorului al cărui raport îl vizualizăm)
* centerCoordinates (conține coordonatele locației administratorului și este primită ca parametru)
* selectedReportLocation (conține locația raportului selectat și este primită ca parametru)
* reportsLocation (este o listă ce conține coordonatele tuturor rapoartelor sub forma unui obiect)
* Metodele folosite sunt:
* closeModalHandler() – pentru setarea variabilei modalIsVisible pe fals și ascunderea Modal-ului ce conține detaliile raportului selectat
* **PageNotFound.js**
* Componentă folosită pentru afișarea pe ecran a unei pagini care nu există

1. Interfața-utilizator

* Este formată dintr-un container principal în care se află un mai multe sub-containere, fiecare afișând un mesaj.

1. Metode & variabile

* Variabilele folosite sunt:
* history (conține referința hook-ului useHistory – pentru accesarea istoricului browser-ului folosit)
* Metodele folosite sunt:
* goBackHandler(event) – pentru reîntoarcerea la pagina anterioară
* **Profile.js**
* Componentă folosită pentru afișarea adresei de email, a prenumelui și a numelui administratorului, cu posibilitatea modificării acestora. Totodată, mai există și opțiunile de: modificare a adresei de email, modificare a parolei, ștergere a contului și de modificare a prenumelui și a numelui.

1. Interfața-utilizator

* Este formată dintr-un container principal și două Modal-uri preluate din React-Bootstrap.
* Container-ul este compus dintr-un Card preluat din React-Bootstrap, Card ce conține trei câmpuri (primul este pentru vizualizarea adresei curente de email și celelalte două sunt destinate modificării prenumelui și a numelui administratorului) și patru butoane:
* Schimbarea adresei de email
* Schimbarea parolei
* Ștergerea contului
* Modificarea prenumelui și a numelui
* Primul Modal este vizibil după ce administratorul apasă unul dintre primele trei butoane cu opțiuni disponibile. Acest Modal face tranziția către cel de-al doilea Modal (cel pentru introducerea datelor corespunzătoare și confirmarea acțiunii dorite).
* Cel de-al doilea Modal este destinat confirmării acțiunii dorite de către administrator:
* de introducere a noii adrese de email și a parolei în cazul schimbării adresei de email
* de introducere a vechii parole și a noii parole (precum și a confirmării acesteia) în cazul schimbării parolei
* de introducere a parolei curente în cazul ștergerii contului

1. Metode & variabile

* Variabilele folosite sunt:
* auth (conține metoda getAuth() a Firebase)
* db (conține metoda getDatabase() a Firebase)
* dispatch (conține referința hook-ului useDispatch – pentru expedierea unei acțiuni către Redux)
* history (conține referința hook-ului useHistory – pentru accesarea istoricului browser-ului folosit)
* emailRef, newEmailAddressRef, passwordRef, oldPasswordRef, passwordConfirmationRef, firstNameRef, lastNameRef (conține referința hook-ului useRef – pentru a ține o referință a câmpului dorit: email, noua parolă, vechea parolă, prenume, nume etc.)
* currentAdmin (obiect ce conține administratorul curent din Firebase)
* adminPersonalInformation (obiect ce conține informațiile personale ale administratorului curent preluate din Redux)
* adminPersonalInformationRef (conține o referință la nodul „personalInformation” al administratorului curent din baza de date Firebase)
* **ProfileIcon.js**
* Componentă folosită pentru afișarea inițialei prenumelui administratorului curent în interiorul unui container circular.
* Este formată dintr-un container principal în care se află inițiala prenumelui adminstratorului.
* **ReportDetails.js**
* Componentă folosită pentru afișarea detaliilor unui raport selectat de către administrator.

1. Interfața-utilizator

* Este formată dintr-un container principal în care se află mai multe linii și coloane. Pe prima linie vor fi randate detaliile raportului. Pe prima coloană a primei linii va fi afișată imaginea raportului selectat, iar pe cea de-a doua vor fi afișate detaliile acestuia. Pe cea de-a doua linie va fi afișată condiționat harta cu locația raportului (în cazul în care administratorul dorește să o vizualizeze).

1. Metode & variabile

* Variabilele folosite sunt:
* selectedReport (preia din Redux detaliile raportului selectat de către administrator)
* db (conține metoda getDatabase() a Firebase)
* GOOGLE\_KEY (conține cheia de la API-ul Google Maps și o preia din fișierul privat .env)
* customStyle (conține stilizarea unui element HTML)
* googleMapURL (conține adresa URL la care vom face o cerere de tip GET)
* userPersonalInformationRef (conține referința la nodul în care se află informațiile despre utilizator)
* retrievedPersonalInformation (conține un obiect în care vom adăuga informațiile utilizatorului din baza de date)
* selectedReportDateTime (conține data și timpul la care a fost adăugat raportul)
* selectedReportLocation (conține locația raportului)
* mapsVisible (este setată inițial pe true și se schimbă atunci când administratorul dă click pe afișare/ascundere hartă)
* Metodele folosite sunt:
* toggleMapHandler() – pentru setarea variabilei mapIsVisible
* **ReportItem.js**
* Componentă folosită pentru randarea unui raport din listă.

1. Interfața-utilizator

* Este formată dintr-un element din listă (list item) ce conține un Card și un Modal preluate din React-Bootstrap.
* Card-ul este împărțit pe orizontală în trei containere:
* Container ce afișează categoria raportului
* Container ce afișează data și timpul la care a fost înregistrat raportul
* Container ce randează iconițele preluate din React-Bootstrap. Prima iconiță arată starea raportului (rezolvat/nerezolvat), iar cea de-a doua redirecționează administratorul către pagina în care sunt afișate detaliile raportului (dacă a dat click pe aceasta).
* În cazul în care ecranul are dimensiunea XL, LG sau MD, Card-ul conține o singură linie cu trei coloane.
* În cazul în care ecranul are dimensiunea SM sau XS, Card-ul conține o singură linie cu două coloane. Prima coloană conține două linii (pe prima este afișată categoria raportului și pe cea de-a doua data și timpul adăugării acestuia), pe când cea de-a doua coloană conține o lingură linie (pe aceasta este randat container-ul cu iconițe).
* Dacă raportul a fost deja rezolvat, iconița va avea culoarea aurie (gold). În caz contrar, aceasta va avea culoarea vișinie (crimson).
* Modal-ul este afișat pe ecran în momentul în care administratorul dă click pe iconița pentru modificarea stării raportului. În cazul în care administratorul dă click pe butonul de salvare („save”), starea raportului se va modifica în baza de date și va fi vizibilă imediat în lista de rapoarte.

1. Metode & variabile

* Variabilele folosite sunt:
* adminPersonalInformation (conține informațiile personale ale administratorului curent, acestea fiind luate din Redux)
* db (conține referința metodei getDatabase() a Firebase)
* dispatch (conține referința hook-ului useDispatch – pentru expedierea unei acțiuni către variabilele din Redux)
* history (conține referința hook-ului useHistory – pentru accesarea istoricului browser-ului folosit)
* report (obiect ce conține detaliile raportului și primit ca parametru)
* reportDetailsRef (referință la nodul în care sunt stocate detaliile raportului în baza de date Firebase)
* modalIsVisible (este setată inițial pe fals)
* modalMessage (este setată inițial ca mesaj gol)
* isChecked (este setată inițial pe valoarea stării raportului)
* reportDateTime (conține data și timpul la care a fost adăugat raportul)
* reportParsedDateTime (conține data și timpul formatate– DD/MM/YY hh:mm:ss)
* Metodele folosite sunt:
* checkReportHandler() – apelează metoda showModalHandler()
* checkReportDetailsHandler() – expediază acțiunea de setare a raportului selectat în Redux și redirecționează administratorul către ruta cu detaliile raportului (/reports/:reportId)
* closeModalHandler() – ascunde Modal-ul
* modifyReportStatusHandler()
* modifică în baza de date starea raportului (și adaugă id-ul administratorului care l-a rezolvat dacă este cazul)
* modifică vizual starea raportului (în lista de rapoarte se modifică iconița)
* expediază acțiunea de actualizare a raportului selectat din Redux
* actualizează în baza de date nivelul utilizatorului (în cazul în care a fost modificat)
* actualizează procentajul de reducere a impozitului
* ascunde Modal-ul
* showModalHandler() – afișează Modal-ul în funcție de starea raportului selectat
* **Topbar.js**
* Componentă folosită pentru randarea barei de navigare.

1. Interfața-utilizator

* Este formată dintr-un Navbar preluat din React-Bootstrap ce conține mai multe componente create de către noi (CustomNavLink).
* Aplicația noastră WEB conține un număr de cinci componente de tip CustomNavLink: Reports, Users, Map, Profile, Log out.

1. Metode & variabile

* Variabilele folosite sunt:
* dispatch (conține referința hook-ului useDispatch – pentru expedierea unei acțiuni către variabilele din Redux)
* reportsLinkIsActive, usersLinkIsActive, mapLinkIsActive, profileLinkIsActive (conțin starea unui link: activ/inactiv)
* navbarIsExpanded (conține starea barei de navigare: extins/nu)
* auth (conține referinta metodei getAuth() a Firebase)
* history (conține referința hook-ului useHistory – pentru accesarea istoricului browser-ului folosit)
* location (conține referința hook-ului useLocation – pentru identificarea rutei pe care se află administratorului)
* Metodele folosite sunt:
* currentPathHandler() – identifică ruta curentă și setează link-urile drept active/inactive în funcție de aceasta
* logoutHandler() – apelează metoda signOut a Firebase, expediază acțiunea de deconectare a administratorului pentru Redux și îl redirecționează către ruta de deconectare („/logout”)
* redirectHandler() – pentru redirecționarea către pagina dorită
* toggleNavbarHandler() – pentru modificarea valorii variabilei navbarIsExpanded
* Hook-urile folosite sunt:
* useEffect (apelează metoda currentPathHandler de fiecare dată când administratorul navighează către o altă pagină)
* **UserItem.js**
* Componentă folosită pentru randarea unui utilizator din listă.

1. Interfața-utilizator

* Este formată dintr-un element din listă (list item) ce conține un Card preluat din React-Bootstrap.
* Card-ul este împărțit în trei containere:
* Container ce afișează numele și prenumele utilizatorului
* Container ce afișează nivelul curent al utilizatorului
* Container ce randează o iconiță preluată din React-Bootstrap. Această iconiță redirecționează administratorul către pagina în care sunt afișate detaliile utilizatorului (dacă a dat click pe aceasta).
* În cazul în care ecranul are dimensiunea XL, LG sau MD, Card-ul conține o singură linie cu trei coloane.
* În cazul în care ecranul are dimensiunea SM sau XS, Card-ul conține o singură linie cu două coloane. Prima coloană conține două linii (pe prima este afișat numele utilizatorului și pe cea de-a doua nivelul acestuia), pe când cea de-a doua coloană conține o lingură linie (pe aceasta este randată iconița).

1. Metode & variabile

* Variabilele folosite sunt:
* dispatch (conține referința hook-ului useDispatch – pentru expedierea unei acțiuni către variabilele din Redux)
* history (conține referința hook-ului useHistory – pentru accesarea istoricului browser-ului folosit)
* user (obiect ce conține detaliile utilizatorului și primit ca parametru)
* userPersonalInformation (obiect ce stochează informațiile personale ale utilizatorului)
* Metodele folosite sunt:
* checkUserDetailsHandler() – expediază acțiunea de setare a utilizatorului selectat în Redux și redirecționează administratorul către ruta cu detaliile acestuia (/users/:userId)
* **UserProfile.js**
* Componentă folosită pentru afișarea detaliilor unui utilizator selectat de către administrator.

1. Interfața-utilizator

* Este formată dintr-un container principal în care se află mai multe coloane. Pe prima coloană va fi afișată imaginea profilului utilizatorului selectat, iar pe cea de-a doua vor fi afișate informațiile acestuia.

1. Metode & variabile

* Variabilele folosite sunt:
* selectedUser (preia din Redux detaliile utilizatorului selectat de către administrator)
* userPersonalInformation (conține un obiect ce stochează informațiile personale ale utilizatorului)
* userPersonalReports (conține un obiect ce stochează lista de rapoarte ale utilizatorului)
* api.js
* auth-slice.js
* index.js (store)
* location-slice.js
* report-slice.js
* reports-list-slice.js
* user-slice.js
* users-list-slice.js
* custom-methods.js
* firebase.js

1. Aplicația mobilă

* feature
* addreport
* authentication
* editprofile
* editreport
* home
* myreports
* nearbydangersmap
* selectphoto
* splashscreen
* model
* AdminPersonalInformation
* MyCustomDate
* MyCustomDateTime
* Report
* UserLocation
* UserPersonalInformation
* utility
* CategoriesSpinnerAdapter
* DatePickerFragment
* DeleteReportCustomDialog
* JsonPlaceHolderAPI
* MyCustomMethods
* MyCustomVariables
* TimePickerFragment