Aplicatie de gestionare a evenimentelor de tip workshop/prezentare in domeniul IT

PROIECT DE DIPLOMĂ

Autor: **Tămaș Bogdan David**

Conducător științific: **Titlu.ing. Dan Radu**

|  |  |
| --- | --- |
| DECAN  **Prof.dr.ing. Liviu MICLEA** | Vizat,  DIRECTOR DEPARTAMENT AUTOMATICĂ  **Prof.dr.ing. Honoriu VĂLEAN** |

Autor : **Tămaș Bogdan David**

Aplicatie de gestionare a evenimentelor de tip workshop/prezentare in domeniul IT

1. **Enunţul temei:** *Aplicatie de gestiune a unor evenimente in domeniul IT care sa permita unui utilizator anumite operatii de acces si de manipularea a unor evenimente.*
2. **Conţinutul proiectului:** *Pagina de prezentare, Declarație privind autenticitatea proiectului, Sinteza proiectului, Cuprins, Titlul capitolului 1, Titlul capitolului 2,… Titlul capitolului n, Bibliografie.*
3. **Locul documentaţiei :** *Universitatea Tehnică din Cluj-Napoca*
4. **Consultanţi:** *ing. Dan Radu (dacă este cazul)*
5. **Data emiterii temei:** 01.04.2020
6. **Data predării:**

Semnătura autorului

Semnătura conducătorului științific

**Declaraţie pe proprie răspundere privind**

**autenticitatea proiectului de diplomă**

Subsemnatul(a)  **Tămaș Bogdan David**, legitimat(ă) cu CI/BI seria MM

Nr. 801902, CNP 1970727244219, autorul lucrării:

Aplicatie de gestionare a evenimentelor de tip workshop/prezentare in domeniul IT elaborată în vederea susţinerii examenului de finalizare a studiilor de licență la **Facultatea de Automatică și Calculatoare**, specializarea **Automatică și Informatică Aplicată,** din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea Choose an item. a anului universitar 2019-2020, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

In cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

Data Prenume NUME

(semnătura)

**SINTEZA**

proiectului de diplomă cu titlul:

Aplicatie de gestionare a evenimentelor de tip workshop/prezentare in domeniul IT

Autor: **Tămaș Bogdan David**

Conducător științific: **Titlu.ing. Dan Radu**

1. Cerinţele temei: Aplicația trebuie să permită înregistrarea și autentificarea unui utilizator într-o platforma care stochează evenimentele anterioare și ulterioare în doua interfețe diferite. Evenimentele la care utilizatorul s-a înscris vor apărea într-o a treia interfață. Se vor crea și alte interfețe care să permită utilizatorului următoarele acțiuni: propunerea unui topic, votarea oricărui topic existent și prioritizarea acestuia, acordarea de feedback evenimentelor anterioare într-un mod interactiv, accesarea unei pagini de instrucțiuni, accesarea unui profil personal, accesul la topul general precum și la o evidenta a rezultatelor acumulate.

2. Soluţii alese: Pentru a controla volumul mare de date acumulat după popularea aplicației am folosit o baza de date principala (MySql) și una secundara(Json-Server). Comunicarea cu baza de date se face direct (NodeJS), iar design-ul (ReactJS) este implementat folosind preponderent librăria material-ui.

3. Rezultate obţinute: Rezultatele obținute sunt mulțumitoare, iar aplicația are o interfață interactiva cu utilizatorul respectând restricțiile și performantele impuse.

4. Testări şi verificări: Testare s-a realizat prin mai multe verificări manuale a performantelor și a răspunsurilor primite de la baza de date precum și direct de la cod. Expresivitatea codului s-a produs prin mai multe procese de refactorizare și s-a verificat prin folosirea bibliotecii eslint.

5. Contribuţii personale: Pre-analiza și documentarea necesara, implementarea propriu-zisa a funcționalității și a design-ului aplicației, configurarea și integrarea bazelor de date, modificare aferente necesare pentru a asigura un cod cât mai explicit. 6.

Surse de documentare: Algoritmi de implementare eficienta, algoritmi de comunicarea rapida cu baza de date, site-uri web pentru librarii de design, tehnici de criptare, tutoriale de integrare a stilurilor, tehnici de refactorizare.

Semnătura autorului

Semnătura conducătorului științific

Cuprins

[1 Introducere 2](#_Toc477457095)

[1.1 Context general 2](#_Toc477457096)

[1.2 Obiective 2](#_Toc477457097)

[1.3 Specificații 2](#_Toc477457098)

[2 Studiu bibliografic 3](#_Toc477457099)

[3 Analiză, proiectare, implementare 4](#_Toc477457100)

[4 Concluzii 5](#_Toc477457101)

[4.1 Rezultate obținute 5](#_Toc477457102)

[4.2 Direcții de dezvoltare 5](#_Toc477457103)

[5 Reguli de formatare 6](#_Toc477457104)

[5.1 Formatarea paginii 6](#_Toc477457105)

[5.2 Titluri și stiluri 6](#_Toc477457106)

[5.3 Figuri, tabele și ecuații 7](#_Toc477457107)

[5.3.1 Figuri 7](#_Toc477457108)

[5.4 Tabele 7](#_Toc477457109)

[5.5 Ecuații 7](#_Toc477457110)

[5.6 Referințe bibliografice 8](#_Toc477457111)

[6 Bibliografie 9](#_Toc477457112)

# Introducere

## Context general

În ultimii ani în domeniul IT s-a înregistrat o evoluție continua cu o adaptabilitate pe schimbare axata pe informații și tehnologii cât mai recente ce a determinat o explozie din punct de vedere atât a calității, cât și a cantității de produse. Acest domeniu presupune o viață proactiva în care se dorește o dezvoltare permanenta a produsului și o evoluție a acestuia conform restricțiilor impuse de piață și de client.

Evoluția acestui domeniu a ajutat și la o creștere considerabila a numărului de angajați deoarece aceștia au o mulțime de oportunități de dezvoltare personala și profesionala.

Numărul evenimentelor și conferințelor din domeniul tehnic este într-o continua creștere ceea ce sugerează o atmosfera progresiva a interesului către acest domeniu. Deoarece acest progres a avut un ritm atât de alert în ultimii ani a dus la o peletizare de subdomenii care-i oferă participantului oportunitarea de a se îndreptă înspre orice tehnica dorește s-o abordeze. Odată cu creșterea numărului de evenimente s-a dorit și o organizare cât mai buna a acestora și o evidenta cât mai divizata.

Aplicația creata are un scop extrem de practic și efectiv în a ușură păstrarea evidentei și a ajuta la o planificare cât mai facila a evenimentelor. Obiectivul principal este acela de-a crea o interfață care să ilustreze un mediu familiar pentru utilizator ținând totodată cont de preferințele acestuia.

Acordarea de review precum și sortarea evenimentelor și reținerea acestora într-un loc prestabilit se întâmplă de obicei pe mai multe flancuri în aplicațiile generale ceea ce poate genera confuzie. Acest proiect dorește să ajute utilizatorul să treacă prin aceste etape într-un mod mult mai plăcut și rapid.

**// Prezentare pe capitole**

## 

## Obiective

Obiectivul principal al acestei aplicații este de a crește rata productivității și a vitezei de organizare în ceea ce privește răspândirea informațiilor în domeniul IT și de a determina oamenii din acest domeniu să participe la cât mai multe evenimente.

Participarea la evenimentele/workshop-urile angajatorilor din domeniu aduce cu ea mai multe oportunități:

- șansele de a găsi un job cât mai rapid cresc

- afli beneficiile oferite de fiecare firmă

- identificarea soluțiilor

- gratuitate

- diversitatea temelor prezentate

De asemenea, votarea fiecărui topic care ar putea să fie prezentat într-un workshop/prezentare viitoare de către orice utilizator ajută organizatorii să aleagă o temă cât mai utilă și care să stârnească interesul cât mai rapid.

De ce această temă?

Consider că acest domeniu oferă multe oportunități de a învăță și a rămâne în temă cu tot ce este de actualitate, iar dezvoltarea unei platforme care să ajute la o mai mare popularitate a evenimentelor este un lucru vital. Varietate temelor oferă de asemenea o mai mare maleabilitate în ceea ce privește orientarea fiecărui individ.

## Specificații

În aplicația rezultată se dorește obținerea unei interfețe friendly care să ajute utilizatorul să se acomodeze cât mai rapid cu o arhitectură software maleabilă care să manipuleze date într-un volum mediu cât mai agil.

Specificațiile/funcționalitatea/performanțele/securitatea trebuie să respecte o serie de constrângeri:

- câmpurile cu datele confidențiale ale fiecărui utilizator nu vor fi vizible

- datele confidențiale vor fi trimise codat către bază de date

- accesul la rutele private se va face pe o perioadă limitată de timp

- call-urile către bază de date (principală/ secundară) trebuie să aibă o duarată medie mai mică de 4 secunde, iar acest proces va fi mascat de fiecare dată de un buffer

- token-ul accesului principal va fi implementat printr-un algoritm de generare cu un anumit pattern

- existența unei pagini de informare a unui utilizator este impusă, iar aceasta va putea fi accesată de pe o rută publică

- la procesul de înregistrare al unui user va exista o serie de mesaje și de validări care să poată să-l ghidez

- top-urile și listele vor fi ilustrative și cât mai comprehensibile pentru user implicând un design cât mai transparent

- accesul la profilul principal se va putea efectua de pe orice rută privată

- elementele din liste vor fi afișate printr-un algoritm care să limiteze o recurență fără a le afișa exhaustiv de la început

- user-ul va avea control deplin asupra evenimentelor personale, dar nu va putea să influențeze deciziile celorlalți

În aplicația rezultată se dorește obținerea unei interfețe friendly care să ajute utilizatorul să se acomodeze cât mai rapid cu o arhitectură software maleabilă care să manipuleze date într-unvolum mediu cât mai agil.

# Studiu bibliografic

Funcționalitatea aplicației este divizată după următoarea structura:

- **Front-end**:

- HTML

- CSS

- Javascript

Implementarea efectivă a design-ului și funcționalității din față s-a realizat ajutorul librăriei **ReactJS** care constituie o comprimare a tehnologiilor de mai sus într-o manieră mai organizată pentru construirea interfețelor cu utilizatorul.

- **Back-end**:

- NodeJS

Comunicarea cu baza de date principale s-a realizat prin **NodeJS** printr-o metodă simplă și efectivă folosind sintaxa de JS și comenzi de SQL.

- **Baze de date**:

- MySQL

- JSON Server

- **Platforma de control:**

**-** Github

## Front-end

**2.1.1. HTML**

HyperText Markup Language reprezintă limbajul de marcare a componentelor în paginile WEB. Prin HTML se asigură organizarea prezentării informațiilor, ierarhia lor și poziția in pagină.

Extensia “.html” sau “.htm” definesc fișierele de tip html, iar citirea și modificarea acestora nu necesită un editor sau mediu de programare specific.

Componentele dintr-o pagină html sunt definite de anumite atribute, iar tipul lor este ilustrat printr-un tag specific.

Integrarea algoritmilor implementați în JS și a design-ului creat în CSS se face în general în HTML, iar pentru a evita integrarea manuală se poate apela la diverse librării ca si jQuery, Vue.js, ReactJS…

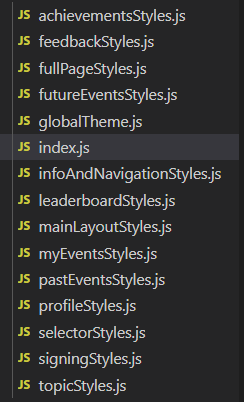
**2.1.2. CSS**

Cascading Style Sheets este un standard de formatare a design-ului unor componente care ajută la crearea unor interfețe personalizate.

Modul în care am implementat structura de design de tip CSS este unul direct, folosind librăria material-ui și apelând la metoda “withStyles” am putut să creez un design integrat în fișierul de tip JS.



Style-urile tuturor componentelor s-au stocat intr-un folder principal si s-au exportat printr-un index.



Asadar, datorita maleabilitatii oferite de aceasta librarie stylurile au fost definite ca si obiecte constante.

**2.1.3. Javascript**

Javascript este un limbaj de programare orientat pe obiect care se folosește mai ales la introducerea functionalitatilor într-o pagină web/ aplicație, dar și pentru accesul la obiecte încapsulate în diferite aplicații. Deoarece acest limbaj este bazat pe mai multe concepte de prototipuri acesta are o comunicare mai practică în ceea ce privește structura paginilor web înglobând paginile de tip HTML cu ușurință și având o structura mult mai ductilă.

Design-ul principal și cea mai mare parte a funcționalității aplicației a fost implementată în framework-ul React deoarece consider că are o sintaxa foarte ușor de înțeles și multe tehnici de organizare a codului într-un mod efectiv.

Interfața este dominată de elemente care fac parte din librăria **material-ui**. Am apelat la această librărie deoarece consider că are o structura eleganta și dinamica a componentelor, iar stilurile principale nu necesita existența unor fișiere de tip CSS.



Dependințele/librăriile folosite exclusiv în React pentru implementarea aplicației sunt următoarele:

- design & animații:

- material-ui-pickers

- @materia-ui/core

- @material-ui/icons

- @material-ui/lab

- react-animated-modal

- funcționalitate:

- moment

- react-datepicker

- react-dom,

- react-duration-picker

- react-form-validator-core

- react-hot-loader

- react-material-ui-form-validator

- react-router-dom

- react-spring

- react-transition-group

- react-scripts

- validator

- securitate:

- react-cookie

- react-password-strength

- js-cookie

- universal-cookie

-comunicare cu baza de date:

- axios

- json-server

## Back-end

**2.2.1 NodeJS**

NodeJS este un mediu rulare a codului Javascript în afara unui browser Web. Acesta se folosește în preponderență pentru implementariile de tip back-end și nu necesita o interfață grafică pentru a-și îndeplini scopul.

În aplicația creată am folosit NodeJS pentru a realiza comunicarea cu baza de date manipulând acțiunile de tip CRUD și comenzile de SQL printr-un Router pe care l-am extras din librăria express.



## Baze de date

**2.3.1. MySQL**

MySQL este un mediu folosit pentru gestiunea bazelor de date de tip relațional folosind tabele.

Datele sunt stocate și manipulate prin comenzi specifice că și:

- SELECT \* FROM [nume\_tabel] – acțiunea de selectare dintr-o tabela

- INSERT INTO [nume\_tabel] – inserarea în tabela

- UPDATE [nume\_tabel] SET [câmpuri] – modificare a câmpurilor dintr-o tabelă

- DELETE FROM [numa\_tabel] WHERE [condiție] –ștergerea unui element dintr-o tabelă.

Se impune datelor reținute în tabele să aibă anumite câmpuri definitorii printre care și un camp primar diferit de celelalte elemente.

Pentru lansarea bazei de date MySQL am folosit pachetul XAMPP deoarece acesta oferă o permisivitate mare asupra bazei de date și este actualizat în permanenta. Acesta ne oferă și acces la log-urile comenzilor efectuate în baza de date pentru a-o putea monitoriza continuu.



Baza de date are legătură directa cu NodeJS, iar rularea acesteia se face local pe portul 8080 utilizând comanda:



Daca baza de date este configurată terminalul va afișa:

C:\Users\bogdan.tamas\AppData\Local\Microsoft\Windows\INetCache\Content.Word\server_listening.png

**2.3.2. JSON-Server**

Json-Sever este un modul inclus în pachetul npm care permite crearea unui server local de tip REST similar unei baze de date.

Creare serverului local este una relativ simpla folosind comanda ‘json-server’ după cum urmează:



Comanda de mai sus va crea un server local în care operațiile de CRUD se vor executa asupra fișierului appDatabase.json folosind portul 3001.

Am ales utilizarea acestui tip de server deoarece controlul asupra datelor este accesibil, iar operarea datelor de tip array/object este mult mai practică.

## Platforma de control

**2.4.1. Github**

Github este o platformă de control care ajuta la păstrarea evidenței codului și la monitorizarea acestuia în permanență. Este des folosit în dezvoltarea proiectelor deoarece previne pierderea datelor și susține tehnici de combinare a codului într-un mod cât mai sigur.

Proiectul principal este împărțit în mai multe branch-uri care servesc un scop unic. De regula, exista mai multe strategii de branching, iar în funcție de aceasta se impune un anumit mod de munca și un anumit tipar al review-ului.

Utilizatorul își înregistrează modificările pe un anumit branch prin comanda “git commit –m [mesaj]”. Mesajul trebuie să fie unul explicit pentru ceilalți utilizatori și să respecte structura “type: scope”.

Principala calitate a platformelor de control este prezența preponderentă a review-ului pe cod ori de câte ori se realizează combinarea branch-urilor. Desigur că prezenta review-ului este uneori opțională, dar pentru o mai buna practică se recomandă utilizarea acesteia în orice împrejurime.

Cele mai comune comenzi pentru controlul software in github sunt:

- git add – adaugă modificările în zona de asamblare

- git commit – comandă urmată de un mesaj care va constitui eticheta modificarilor adaugate în zona de asamblare

- git push - local -> repository

- git pull – repository -> local

- git branch [nume\_branch] – crearea unui nou branch

Pentru o clarificare mai bună a ceea ce înseamnă un branch și cum funcționează comenzile de mai sus se poate folosi comanda “git gui” care va deschide o fereastra de control a modificărilor existente.

# Analiză, proiectare, implementare

**3.1. Scop, stil si integrarea datelor**

Așa cum am menționat mai sus, scopul principal al aplicației este de-a eficientiza lucrul cu evenimente și de-a le gestiona printr-o structura organizatorica implementata pe baza unui design predefinit.

Design-ul este din punct de vedere analitic unul bazat pe animații interactive care să împiedice cursul aplicației să devina unul monoton. Style-ul general este unul elegant fară elemente poziționate sau modificate în mod excesiv. Fiecare style al unei componente aferente este reținut în folderul principal al style-urilor, iar obiectul fiecărui style este de forma :



După ce obiectul principal de style este importat în componentă se va integra direct în elementele structurii prin metoda impusa de librăria « withStyles » :



Acum obiectele de style vor fi incluse în obiectul principal « classes » care este integrat în props prin metoda specificata mai sus.

Pentru implementarea propriu-zisă a funcționalității am folosit diverși algoritmi de implementare care să controleze datele manipulate atât din punct de vedere logic cât și analitic. Algoritmii principali sunt relativ simpli din punct de vedere matematic și se bazează dominant pe tehnici de filtrare a datelor apelând la diverse funcții predefinite de manipulare a șirurilor simple și șirurilor de obiecte. Metodele principale folosite sunt :

- filter : filtrează datele după o anumită condiție

- map : crearea unui șir nou rezultat din cel pe care se apelează metoda aplicând funcția impusă

- reduce : valoriile care nu respectă condiția menționată în funcție sunt eliminate din șirul principal

- sort : poziția fiecărui element din sir se vă modifica în funcție de condiția principala rezultând un nou sir sortat iterativ.

Deoarece datele reținute în Json-Server sunt suportate și într-un volum mai mare este necesară filtrarea acestora înainte de execuția efectivă a operațiilor pe acestea.

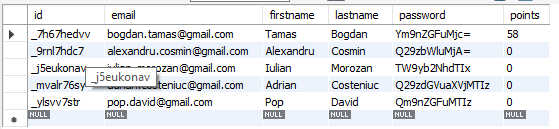
**3.2. Structura si integrare baza de date.**

**3.2.1. MySQL**

Baza principală de date presupune un singur tabel care are funcția de a stoca datele de securitate și de acces.

Configurarea bazei de date s-a realizat după instalarea aplicației MySql Workbench 8.0 și XAMPP prin etapa de inițializare a unei baze de date și a unui tabel incorporat și etapa de controlare a acesteia prin aplicația XAMPP.

Tabelul principal de administrare a securității aplicație are următoarea forma:



Utilitatea fiecărui camp este următoarea:

* **id**: cheia principala a tabelului care reține valoare ce permite accesul utilizatorului în aplicație. Valoare fiecărui camp respectiv este generata în NodeJS prin funcția următoare :



* **email**:valoarea câmpului email din cadrul procesului de înregistrate a fiecărui utilizator. Valoarea este unică fiecărui camp, iar existența unui câmp duplicat este împiedicată de funcționalitatea implementată.
* **firstname & lastname**: valoarea câmpurilor firstname și lastname din stocate în procesul de înregistrare
* **password**: valoarea câmpului password & passwordConfirmation convertita în baza 64.
* **points**: valoarea ce determină poziția user-ului în top este configurată și actualizată la fiecare logare în aplicație și este sincronizată cu baza secundară.

Comenzile CRUD ale bazei de date sunt manipulate în NodeJS prin componenta Router după urmatoareă formă generală :

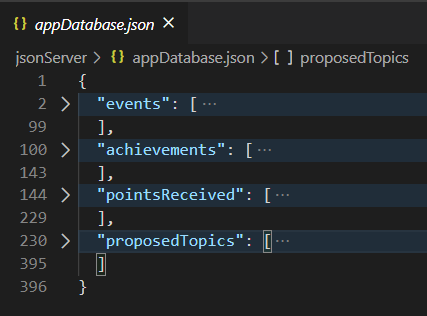




**3.2.2. Json-Server**

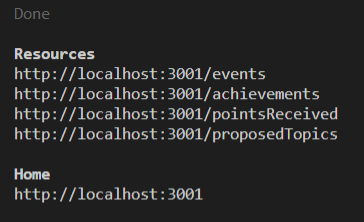
Baza secundară de date presupune un fișier de tip JSON ce are în componență sa mai multe array-uri de obiecte complexe unde se rețin date cu câmpuri unice.

Conținutul bazei de date este următorul:



Fiecare dintre aceste câmpuri principale are mai multe câmpuri secundare care se accesează prin parcurgere în adâncime sau prin destructurare succesiva.

După comanda de lansare a bazei de date pe portul 3001 în terminal se va afișa următorul rezultat:



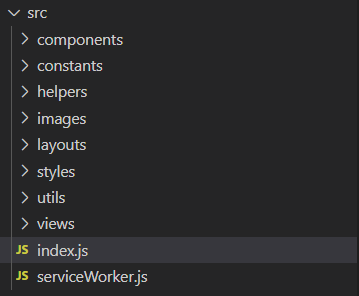
Cele 4 rute de mai sus vor fi definite de array-urile retinute in JSON, iar modficarea datelor se va face direct pe ruta dorită.

**3.3. Structură proiect**

Structura codului este una organizată care s-a definit prin mai multe procese de refactorizare realizate aferent implementării.

Aplicația este alcătuită din 3 foldere principale.

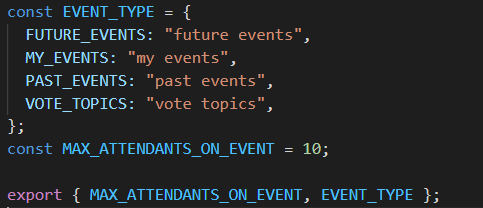
Front-end – **src**



Components – folderul ce are în compoziția sa elementele propriu-zise ale aplicației. Aici se integrează stiulul în componente și se administrează funcționarea fiecăreia. În compoziția acestuia avem mai multe subfoldere în care se respecta o structura de tipul list-item. Subfolderele care definesc componentele aplicație sunt: achievements, feedback, fullPage, futureEvents, infoAndNavigation, leaderboard, myEvents, pastEvents, profile, signing, voteTopics. Accesul extern spre importarea acestora se face printr-un index.

Constants – folderul în care se rețin datele folosite repetitiv în cadrul aplicației și care nu au un comportament dinamic sau sunt funcții. Constantele sunt exportate din index-ul folderului și sunt împărțite în: charLimitConstants, databaseConstants, eventsConstants, signingConstants, topicConstants.

Structura unui fișier de constante:



Helpers – reține funcțiile folosite repetitiv în cadrul aplicației.

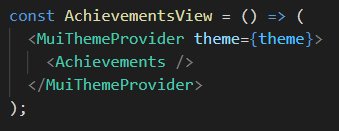
Images – imaginiile folosite în cadrul aplicației.

Layouts – aici se rețin schemele principale : schema de înregistrare și schema generala.

Styles – aici sunt reținute style-urile.

Utils – în acest folder sunt reținute componentele secundare și cele exclusiv funcționale care sunt folosite repetitiv de către componentele principale. Componentele secundare sunt de tip Alerte, Selectori, Rute Private sau Tranziții.

Views – aici sunt reținute containerele componentelor care au scopul de-a integra fiecare components în tema globală a aplicației.



index.js – aici se importă toate containerele aplicației și se atribuie fiecăreia o anumită rută specifică respectând structura următoare :

<Route path="/login" component={LoginView} />

<PrivateRoute path="/profile" component={ProfileView} />

Back-end – **api**

În acest folder avem următoarele componente :

* o funcție de generare a id-ului fiecărui ușer când se apelează metoda de insert
* fișierul de conexiune cu baza de date cu extensia « .sql »
* fișierul de rutare către baza de date
* fișierul serverului în care se stabilește portul aferent serverului principal al bazei de date
* fișierul de creare a conexiunii sql

Database – **jsonSever**

Acest folder are în componența sa un fișier de tip json care vă stoca datele și un fișier de documentație care să explice modul de rulare și folosire al acesteia

**3.4. Implementare tehnică si cod sursă**

**3.4.1. Editor si formatare cod**

Editorul folosit pentru implementarea codului sursa este Visual Studio Code. Am folosit acest editor deoarece structura codului este foarte expresivă aici și permite diverse tehnici de refactorizare inteligent. De asemenea este extrem de eficient în evidențierea sintaxei și a completării inteligente a codului.

Prettier este o aplicație de formatare a codului integrată în orice enviroment și l-am folosit pentru a avea că rezultat un cod cât mai elegant.

**3.4.2. Compoziție cod**

Majoritatea elementelor din compoziția aplicației sunt create și controlate prin funcții constante și nu prin componente pentru un mai bun consum de memorie și o mai mare operativitate.

Renunțând la o implementare bazata pe componente s-a renunțat și la axarea pe lifecycle a componentelor și la manipulări de date folosind state sau constructori care să stocheze props-urile.

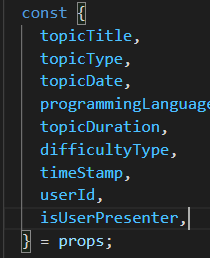
O structură bazată pe funcții constante duce la un randament mai bun al render-ului elementelor evitând apariția timpului mort sau call-urile de durată ridicată.

Props – sunt argumentele trimise prin atribute HTML componentelor din React

Props-urile sunt preluate în funcții constante prin parametri.



Pentru o mai bună practică a funcționalității se aplică și o destructurarea a acestora după preluare lor.

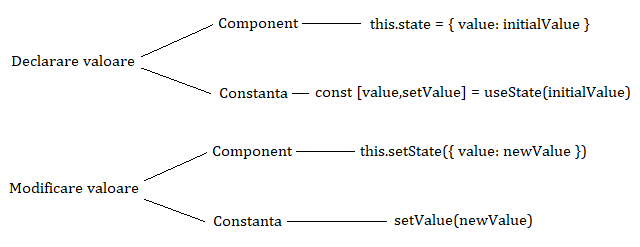


Hooks – metode adăugate în React 16.8 care permit utilizarea state-ului și a altor metode din cadrul claselor fară a fi nevoie de a crea o clasa.

Înlocuirea state-ului în cadrul obiectelor constante se face prin metoda *useState* din cadrul librăriei React, iar urmărirea și modificarea valorilor din simularea state-ului se face folosind *hooks*.

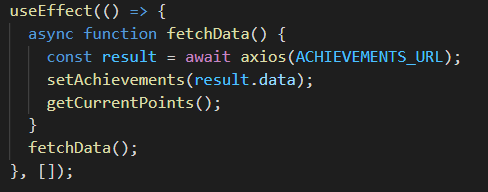
C:\Users\bogdan.tamas\AppData\Local\Microsoft\Windows\INetCache\Content.Word\hooks.png

Conversia state-ului unei componente prin metoda useState:



Pentru a substitui metodele componentDidMount și componentDidUpdate existente în cadrul lifecycle-ului componentelor am folosit metoda *useEffect* bazata pe funcții asincrone. Astfel că la fiecare modificare funcțională a unui element funcția useEffect vă fi executata. Implementare asincrona este des folosita în acest caz deoarece împiedică apariția unei bucle infinite care să apară în cazul unor modificări persistente asupra elementului principal sau a unui call recursiv.

Integrarea unei funcții useEffect în cadrul unui element se face după cum urmează :

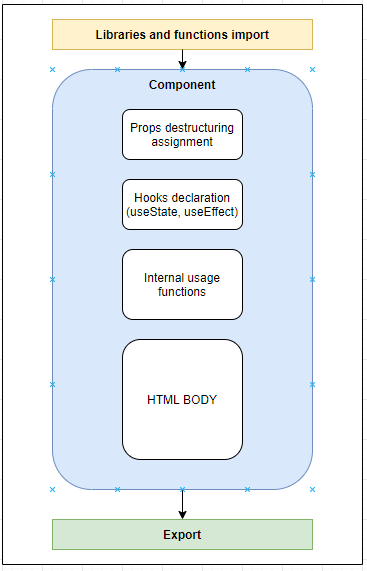
**

După cum se poate observa mai sus funcția fetchData în care se face call-ul către ACHIEVEMENTS\_URL este făcut cu expresia *await* care realizează o pauza în execuția funcției asincrone până Promise-ul call-ului este rezolvat.

CORP FUNCȚII CONSTANTE

Functiile constante constituie baza principala a elementelor din aplicatie. Au o structura bine definita restrictionata dupa o anumita arhitectura impusa de la inceput.

Elementele aflate in components respecta in totalitate structura urmatoare.



Pentru a exemplifica teoria de mai sus intr-o maniera practica o voi explica ilustrand-o printr-un pseudocod JS:

* Libraries and functions import

import { function1, function2 } from ‘./path’ ;

import { library1, library2 } from ‘./path1’ ;

* Component
  + Props destructuring assignment :

const { arg1, arg2 … } = props ;

* + Hooks declaration
  + Internal usage function

const function = () => { operatii ; return rezultat ; }

* + HTML BODY

const Component = () => { return HTML\_ELEMENTS ; }

* Export

export default withStyles(componentStyle)(Component) ;

**3.5. Interfata grafica**

Pagina introductiva a aplicatiei este pagina de login:

