



APLICAȚIE DE GESTIONARE A EVENIMENTELOR DE TIP WORKSHOP/PREZENTARE ÎN DOMENIUL IT

PROIECT DE DIPLOMĂ

Autor: **Tămaș Bogdan David**

Conducător științific: **Asis. dr. ing. Dan Radu**

2020

MINISTERUL EDUCAȚIEI NAȚIONALE



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Vizat,

DECAN
Prof.dr.ing. Liviu MICLEA

DIRECTOR DEPARTAMENT AUTOMATICĂ
Prof.dr.ing. Honoriu VĂLEAN

Autor : **Tămaș Bogdan David**

**Aplicație de gestionare a evenimentelor de tip
workshop/prezentare în domeniul IT**

1. **Enunțul temei:** *Aplicație de gestiune a unor evenimente în domeniul IT care să permită unui utilizator anumite operații de acces și de manipulare a evenimentelor.*
2. **Conținutul proiectului:** *Pagina de prezentare, Declarație privind autenticitatea proiectului, Sinteza proiectului, Cuprins, Introducere, Studiu Bibliografic, Analiză, proiectare, implementare, Rezultate și validare, Concluzii, Referințe Bibliografice*
3. **Locul documentației :** *Universitatea Tehnică din Cluj-Napoca*
4. **Data emiterii temei:** 01.04.2020
5. **Data predării:**

Semnătura autorului

Semnătura conducătorului științific _____

MINISTERUL EDUCAȚIEI NAȚIONALE



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

**Declarație 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:

Aplicație de gestionare a evenimentelor de tip workshop/prezentare în domeniul IT elaborată în vederea susținerii examenului de finalizare a studiilor de licență la **Facultatea de Automatică și Calculatoare**, specializarea **Automatică și Informatică Aplicată**, din cadrul Universității Tehnice din Cluj-Napoca, sesiunea Iulie 2020. a anului universitar 2019-2020, declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării, și în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

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

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență*.

Data

04.07.2020

Prenume NUME

Tămaș Bogdan David

MINISTERUL EDUCAȚIEI NAȚIONALE



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

**SINTEZA**

proiectului de diplomă cu titlul:

**Aplicație de gestionare a evenimentelor de tip
workshop/prezentare în domeniul IT**

Autor: **Tămaș Bogdan David**

Conducător științific: **Asis. dr. ing. Dan Radu**

1. Cerințele temei: Aplicația trebuie să permită înregistrarea și autentificarea unui utilizator într-o platformă care stochează evenimentele anterioare și viitoare. Se vor crea interfețe grafice care să permită utilizatorului următoarele acțiuni: propunerea unei teme, votarea oricărei teme existente și prioritizarea ei într-o listă, acordarea de feedback printr-un formular interactiv, accesarea unei pagini de instrucțiuni, accesarea unui profil personal, accesul la topul general și la o evidență a rezultatelor acumulate.

2. Soluții alese: Pentru a controla volumul mare de date acumulat după o eventuală expansiune a cantității de date din aplicație am folosit un server local (JSON Server) și o bază de date principală (MySQL). Comunicarea cu baza de date se face direct (NodeJS), iar designul (ReactJS) este implementat folosind preponderent obiecte din librăria « material-ui »

3. Rezultate obținute: Rezultatele obținute sunt adecvate, iar aplicația are o interfață interactivă cu utilizatorul respectând restricțiile și performanțele impuse.

4. Testări și verificări: Testare s-a realizat prin mai multe verificări manuale a performanțelor și a răspunsurilor primite de la baza de date precum și direct de la cod, dar și prin implementarea unor algoritmi de testare unitară.

5. Contribuții personale: Pre-analiza și documentarea necesară, implementarea propriu-zisă a funcționalității și a designului aplicației, configurarea și integrarea bazelor de date, modificări aferente necesare pentru a asigura un cod cât mai explicit.

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

Semnătura autorului

Semnătura conducătorului științific

MINISTERUL EDUCAȚIEI NAȚIONALE



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Table of Contents

1	INTRODUCERE.....	3
1.1	CONTEXT GENERAL	3
1.2	OBIECTIVE.....	4
1.3	SPECIFICAȚII	4
2	STUDIU BIBLIOGRAFIC.....	6
2.1	FRONT-END.....	6
2.1.1	HTML.....	6
2.1.2	CSS	7
2.1.3	Javascript.....	7
2.2	BACK-END.....	8
2.2.1	NodeJS	8
2.3	BAZA DE DATE.....	8
2.3.1	MySQL.....	8
2.4	SERVER LOCAL.....	9
2.4.1	JSON Server.....	9
2.5	PLATFORMA DE CONTROL.....	9
2.5.1	Github	9
3	ANALIZĂ, PROIECTARE, IMPLEMENTARE.....	11
3.1	SCOP, STIL ȘI INTEGRARE DATE.....	11
3.2	STRUCTURA ȘI INTEGRARE BAZA DE DATE	12
3.2.1	MySQL.....	12
3.2.2	JSON Server.....	13
3.3	STRUCTURĂ PROIECT	14
3.3.1	Front-end – « src »	14
3.3.2	Back-end – « api ».....	15
3.3.3	Server local– « jsonSever ».....	15
3.4	IMPLEMENTARE ȘI COD SURSĂ	17
3.4.1	Editor și formatare cod	17
3.4.2	Compoziție program	17
3.5	INTERFAȚA GRAFICĂ	20
3.5.1	Login/Register.....	20
3.5.2	Header/content/footer	23
3.5.3	Profile/ leaderboard /achievements	25
3.5.4	Future events/Past events/My events	27
3.5.5	Topics voting	29
3.5.6	About page	32
3.5.7	New event constrcutor.....	33
3.5.8	Private route	33
3.6	SECURITATE.....	34
3.6.1	Cookies acces	34
3.6.2	Criptare/decriptare	34
3.7	INTERFAȚA IN ANSAMBLU	35
4	REZULTATE ȘI VALIDARE	37
4.1.1	Apel MYSQL.....	37
4.1.2	Apel JSON Server.....	38
4.1.3	End-to-end testing	38

4.1.4	API testing.....	39
4.1.5	Unit testing	40
5	CONCLUZII.....	43
5.1	REZULTATE OBȚINUTE.....	43
5.2	DIRECȚII DE DEZVOLTARE ULTERIOARE.....	44
6	REFERINȚE BIBLIOGRAFICE	45
6.1	DOCUMENTAȚIE	45
6.2	IMAGINI.....	45

1 Introducere

1.1 Context general

În ultimii ani în domeniul IT s-a înregistrat o evoluție continuă cu o adaptabilitate pe schimbare axată 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 proactivă în care se dorește o dezvoltare permanentă a produsului și o evoluție a acestuia conform restricțiilor impuse de piață și de client.

Evoluția acestui domeniu a implicat și o creștere considerabilă a numărului de angajați deoarece există o mulțime de oportunități de dezvoltare personală și profesională.

Numărul evenimentelor și conferințelor din domeniul tehnic este într-o continuă creștere ceea ce sugerează o atmosferă progresivă a interesului către acest domeniu. Datorită acestui progres atât de alert în ultimii ani, s-a creat o peletizare de subdomenii ce oferă celor interesați oportunitatea abordării și aprofundării oricărei tehnici. Simultan cu creșterea numărului de evenimente se dorește și o organizare cât mai bună a acestora și o evidență cât mai divizată.

Aplicația trebuie să aibă un scop practic și efectiv în a ajuta la o planificare cât mai facilă a evenimentelor. Obiectivul principal este acela de a crea o interfață care să ilustreze un mediu familiar pentru utilizator fără a se face abstracție de preferințele acestuia.

Acordarea review-ului precum și sortarea evenimentelor și reținerea acestora într-un loc prestabilit au loc pe mai multe flancuri în aplicațiile generale ceea ce poate genera confuzie. Un mediu mai organizat ar elimina problema confuziei și ar ajuta la o eficiență mai mare a mediului funcțional.

Această lucrare de licență dorește să ajute utilizatorul să treacă prin aceste etape printr-o manieră mai agilă prin intermediul unei aplicații.

1.2 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 aduce cu ea mai multe oportunități:

- șansele de a găsi un job cât mai rapid cresc
- informarea despre beneficiile oferite de fiecare firmă
- aprofundarea unor subiecte de interes
- învățarea din alte experiențe de viață
- identificarea soluțiilor
- diversitatea temelor prezentate

De asemenea, votarea fiecărei teme care ar putea să fie prezentată într-un workshop/prezentare viitoare de către orice utilizator ajută organizatorii să se concentreze mai mult pe utilitatea și pe cererile generale ale participanților.

De ce această temă?

Consider că acest domeniu oferă multe oportunități de a învăța și a rămâne în temă cu tot ce este de actualitate, iar dezvoltarea unei platforme care să ajute la o mai mare exalație a informațiilor despre evenimentele organizate este fundamental. Varietatea temelor oferă o diversitate mult mai ridicată a direcțiilor pe care le poate urma o persoană.

1.3 Specificații

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

Specificatiile/funcționalitatea/permanențele/securitatea trebuie să respecte o serie de constrângeri:

- câmpurile cu datele confidențiale ale fiecărui utilizator nu vor fi vizibile
- datele confidențiale vor fi trimise codat către bază de date
- accesul la rutele private se va face pe o perioadă limitată de timp
- apelurile către baza de date trebuie să aibă o durată medie mai mică de 2 secunde, iar acest proces va fi mascat de fiecare dată de un buffer
- cheia accesului principal va fi implementată printr-un algoritm de generare cu un anumit model

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

- în cadrul procesului de înregistrare al utilizatorului va exista o serie de mesaje și de validări

- listele vor fi ilustrative și cât mai comprehensibile 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

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

În aplicația rezultată se dorește obținerea unei interfețe care să ajute utilizatorul să se acomodeze cât mai rapid cu o arhitectură software maleabilă.

2 Studiu bibliografic

Funcționalitatea aplicației este divizată după următoarea structură:

- **Front-end:**

- HTML
- CSS
- Javascript

Implementarea efectivă a designului 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 efektivă folosind sintaxa de Javascript și comenzi de SQL.

- **Baze de date:**

- MySQL

- **Server local**

- JSON Server

- **Platforma de control:**

- Github

2.1 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 în 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 designului creat în CSS se face, în general, în HTML, iar pentru a evita integrarea manuală se poate apela la diverse librării ca și jQuery, Vue.js, ReactJS...

2.1.2 CSS

Cascading Style Sheets este un standard de formatare a designului 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 reușit să creez un design integrat în fișierul de tip JS.

```
import { withStyles } from "@material-ui/core";
```

Figure 1.1 Importarea funcției « *withStyles* »

Stilurile tuturor componentelor s-au stocat într-un folder principal și s-au exportat printr-un index.

Asadar, datorită maleabilității oferite de aceasta librerie stilurile au fost definite ca și obiecte constante.

2.1.3 Javascript

Javascript este un limbaj de programare orientat pe obiect care se folosește mai ales la introducerea funcționalităților 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.

Designul 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 efektiv.

Interfața este dominată de elemente care fac parte din librăria **material-ui**. Am apelat la această librerie deoarece consider că are o structură elegantă și dinamică a componentelor, iar stilurile principale nu necesită implementare prin fișiere de tip CSS.

```
"material-ui": "^0.20.2",
```

Figure 1.3 Versiunea librăriei material-ui

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

- design & animații: material-ui-pickers, @material-ui/core, @material-ui/icons, @material-ui/lab, react-animated-modal

```
JS achievementsStyles.js
JS feedbackStyles.js
JS fullPageStyles.js
JS futureEventsStyles.js
JS globalTheme.js
JS index.js
JS infoAndNavigationStyles.js
JS leaderboardStyles.js
JS mainLayoutStyles.js
JS myEventsStyles.js
JS pastEventsStyles.js
JS profileStyles.js
JS selectorStyles.js
JS signingStyles.js
JS topicStyles.js
```

Figure 1.2 Structura folderului de stiluri

- 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

2.2 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 implementările de tip back-end și nu necesită 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.

```
const express = require('express');
const Router = express.Router();
```

Figure 1.4 Importare librărie « express »

2.3 Baza 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 tabelă
- INSERT INTO [nume_tabel] – inserarea în tabelă
- 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 câmp primar diferit de celelalte elemente.

Pentru lansarea bazei de date MySQL am folosit pachetul XAMPP deoarece oferă o permisivitate mare de modificare a datelor și este actualizat în permanență. De asemenea, oferă acces log-urilor comenzilor efectuate în cadrul bazei de date pentru a avea o statistică și o monitorizare continuă.



Figure 1.5 Lansarea bazei MySql din XAMPP

Baza de date are legătură directă cu NodeJS, iar rularea se face local pe portul specificat utilizând comanda:

```
node server.js
```

Figure 1.6 Comanda de pornire a bazei MySql

Daca baza de date este configurată corect terminalul va afișa mesajul din figura 1.7:

```
Server is listening on port 8000
Connected!
Database created
```

Figure 1.7 Rezultatul din terminal a comenzii de pornire

2.4 Server local

2.4.1 JSON Server

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

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

2.5 Platforma de control

2.5.1 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 în github sunt:

- « *git add.* » – adaugă modificările în zona de asamblare
- « *git commit* » – comandă urmată de un mesaj care va constitui eticheta modificărilor 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 fereastră de control a modificărilor existente.

3 Analiză, proiectare, implementare

3.1 Scop, stil și integrare date

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

Designul este din punct de vedere analitic bazat pe animații interactive care să împiedice cursul aplicației să devină unul monoton. Stilul general este unul elegant fără elemente poziționate sau modificate în mod excesiv. Fiecare stil al unei componente aferente este reținut în folderul principal al stilurilor, iar obiectul fiecărui stil este ilustrat în *figura 2.1*:

După ce obiectul principal de stil este importat în componentă, se va integra direct în elementele structurii prin metoda reprezentată în *figura 2.2*:

```
const { classes } = props;
```

Figure 2.2 Destructurarea stilurilor

```
const palette = {
  primary: {
    main: "#001e3d",
    light: "#334b63",
    dark: "#00152a",
    contrastText: "#ffffff",
  },
  secondary: {
    main: "#fbfff7",
    light: "#fbfff8",
    dark: "#afb2ac",
    contrastText: "#000000",
  },
};
```

Figure 2.1 Structura obiectelor de stil

Prin deconstructurarea obiectelor de stil importate în componenta actuală vor fi preluate în constanta « *classes* » și integrate în fiecare component prin argumentul « *className* ».

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 : valorile care nu respectă condiția menționată în funcție sunt eliminate din șirul principal
- sort : poziția fiecărui element din șir se va modifica în funcție de condiția principală rezultând un nou șir sortat iterativ.

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

3.2 Structura și 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 încorporat și etapa de controlare prin aplicația XAMPP.

Tabelul principal de administrare a securității aplicației are următoarea forma:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
id	VARCHAR(20)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
email	VARCHAR(30)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
firstname	VARCHAR(20)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
lastname	VARCHAR(20)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
password	VARCHAR(20)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
points	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 2.3. Structura câmpurilor tabelului MySQL

	id	email	firstname	lastname	password	points
▶	_7h67hedvv	bogdan.tamas@gmail.com	Tamas	Bogdan	Ym9nZGFuMjc=	58
	_9rnl7hdc7	alexandru.cosmin@gmail.com	Alexandru	Cosmin	Q29zbWluMjA=	0
	_j5eukonav	iulian.morozan@gmail.com	Iulian	Morozan	TW9yb2NhdTIx	0
	_mvalr76sy	adrian.costeniuc@gmail.com	Adrian	Costeniuc	Q29zdGVuaXVjMTIz	0
	_ylsvv7str	pop.david@gmail.com	Pop	David	Qm9nZGFuMTIz	0
*	NULL	NULL	NULL	NULL	NULL	NULL

Figure 2.4 Model de conținut in tabelul MySQL

Utilitatea fiecărui câmp este următoarea:

- **id**: cheia principală a tabelului care reține valoarea ce permite accesul utilizatorului în aplicație. Valoarea fiecărui câmp respectiv este generată în NodeJS prin funcția din figura 2.5:

```
const idGenerator = () => '_' + Math.random().toString(36).substr(2, 9);
```

Figure 2.5 Funcția de generare a id-ului

- **email**: valoarea câmpului email din cadrul procesului de înregistrare a fiecărui utilizator. Valoarea este unică fiecărui câmp, iar existența unei valori duplicate este împiedicată de funcționalitatea implementată.
- **firstname & lastname**: valoarea câmpurilor « *firstname* » și « *lastname* » stocate în procesul de înregistrare
- **password**: valoarea câmpului « *password* » & « *passwordConfirmation* » convertită în baza 64.
- **points**: valoarea ce determină poziția utilizatorului în top este configurată și actualizată la fiecare logare în aplicație și este sincronizată cu serverul local.

Comenzile CRUD ale bazei de date sunt manipulate în NodeJS prin componenta « Router »:

```
Router.get("/", (req, res)
```

Figure 2.6 Router.get

```
Router.post('/insert', (req, res)
```

Figure 2.7 Router.post

3.2.2 JSON Server

Creare serverului local este relativ simplă folosind comanda din *figura 2.8*:

```
json-server --watch appDatabase.json --port 3001
```

Figure 2.8 Comanda de pornire a serverului local

- « *json-server* » - comandă efectivă din librăria « npm »
- « *--watch* » [nume_fisier] – se specifică fișierul în care sunt stocate informațiile necesare serverului local
- « *--port* » [numar_port] – numărul portului folosit.

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

3.3 Structură proiect

Structura codului este organizată și definită prin mai multe procese de refactorizare realizate aferent implementării.

Aplicația este divizată în 3 părți.

3.3.1 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 lui sunt incluse mai multe subfoldere în care se respectă o structură de tipul list-item. Subfolderele care definesc componentele aplicație sunt: *achievements*, *feedback*, *fullPage*, *futureEvents*, *infoAndNavigation*, *leaderboard*, *myEvents*, *pastEvents*, *profile*, *signing*, *voteTopic*. Accesul direct către componente se face folosind un index ce va asigura dispoziția lor către accesul extern.

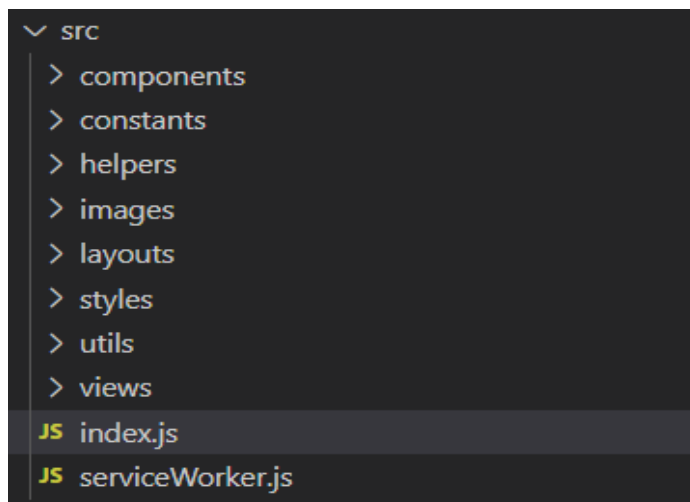


Figure 2.12. Organizarea folderului « src »

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 poate fi observată în figura 2.13.

```
const EVENT_TYPE = {
  FUTURE_EVENTS: "future events",
  MY_EVENTS: "my events",
  PAST_EVENTS: "past events",
  VOTE_TOPICS: "vote topics",
};
const MAX_ATTENDANTS_ON_EVENT = 10;

export { MAX_ATTENDANTS_ON_EVENT, EVENT_TYPE };
```

Figure 2.13 Structura unui fișier de constante

Helpers – reține funcțiile folosite repetitiv în cadrul aplicației.

Images – imaginile folosite în cadrul aplicației.

Layouts – aici se rețin schema de înregistrare și schema generală.

Styles – aici sunt reținute stilurile.

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, Selectorii, Rute Private sau Tranziții.

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

```
const AchievementsView = () => (
  <MuiThemeProvider theme={theme}>
    <Achievements />
  </MuiThemeProvider>
);
```

Figure 2.14. Structura unui obiect de tip « views »

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} />
```

Figure 2.15. Definirea unei rute publice și a unei rute private

3.3.2 Back-end – « api »

În acest folder avem următoarele componente :

- o funcție de generare a id-ului fiecărui utilizator la apelarea metodei 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

3.3.3 Server local- « jsonSever »

Serverul local 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 serverului local este exemplificat în *figura 2.8*.

```
{ } appDatabase.json X
jsonServer > { } appDatabase.json > [ ] proposedTopics
1 {
2 > "events": [ ...
99 ],
100 > "achievements": [ ...
143 ],
144 > "pointsReceived": [ ...
229 ],
230 > "proposedTopics": [ ...
395 ],
396 }
```

Figure 2.8 Câmpurile principale ale serverului local

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

```
{
  "name": "Python basics",
  "type": "Presentation",
  "date": "July 27, 2020 10:00 ",
  "lang": "PY",
  "duration": "1h",
  "difficulty": "Expert",
  "timestamp": 1595844000,
  "attendanceIds": [],
  "waitingListIds": [
    "_7h67hedvv"
  ],
  "feedback": [],
  "id": 4
},
```

Figure 2.9. Structura câmpului « events »

```
{
  "id": 3,
  "userId": "_7h67hedvv",
  "points": [
    {
      "id": 1,
      "value": 2,
      "type": "Feedback",
      "date": "11 Apr 2019"
    }
  ],
},
```

Figure 2.11. Structura câmpului « pointsReceived »

```
{
  "id": 1,
  "picturePath": "../../images/mug.png",
  "title": "Community/Trainer Mug",
  "description": "Circumfluus aeris obsistitur",
  "points": 10
},
```

Figure 2.10 Structura câmpului « achievements »

```
{
  "userId": "_7h67hedvv",
  "isUserPresenter": false,
  "topicType": "Presentation",
  "topicTitle": "Basic C",
  "topicDate": "July 17, 2020 19:30",
  "topicContent": "C for noobies",
  "topicDuration": "1h 30min",
  "difficultyType": "Beginner",
  "programmingLanguage": "C",
  "timestamp": 1595014200,
  "sumOfVotes": -1,
  "userVotes": [
    {
      "userId": "_7h67hedvv",
      "vote": -1
    }
  ],
  "id": 1
},
```

Figure 2.12. Structura câmpului « proposedTopic »

După comanda de lansare a serverului pe portul 3001, în terminal se va afișa rezultatul din figura 2.13.

```
Done

Resources
http://localhost:3001/events
http://localhost:3001/achievements
http://localhost:3001/pointsReceived
http://localhost:3001/proposedTopics

Home
http://localhost:3001
```

Figure 2.13 Rutele serverului local

Cele 4 rute de mai sus vor fi definite de array-urile reținute în *JSON*, iar modificarea datelor se va face direct pe ruta dorită.

Acest folder are în componența sa un fișier de tip json care va stoca datele de tip șir de obiecte și un fișier de documentație care să explice modul de rulare și folosire general.

3.4 Implementare și cod sursă

3.4.1 Editor și formatare cod

Editorul folosit pentru implementarea codului sursă este *Visual Studio Code*. Am folosit acest editor deoarece structura codului este foarte expresivă aici și permite diverse tehnici de refactorizare. 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 ca rezultat un cod organizat în pagină.

3.4.2 Compoziție program

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 bazată pe componente s-a renunțat și la axarea pe ciclul vieții componentelor și la manipulări de date folosind state sau constructori care să stocheze proprietățile.

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

Props – sunt argumentele trimise prin attribute HTML componentelor din React

Proprietățile sunt preluate în funcții constante ca parametri.

```
const NewEvent = (props) => {
```

Figure 2.14 Preluarea proprietăților.

Pentru o mai bună practică a funcționalității se aplică și o tehnică de destructurare a datelor după preluare lor ca în exemplul din *figura 2.15*.

```
const {
  topicTitle,
  topicType,
  topicDate,
  programmingLanguage,
  topicDuration,
  difficultyType,
  timeStamp,
  userId,
  isUserPresenter,
} = props;
```

**Figure 2.15
Destructurarea proprietăților**

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

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*.

```
import { useState, useEffect } from "react";
```

Figure 2.16 Importarea metodelor « *useState* » și « *useEffect* »

Modul de conversie a state-ului unei componente prin metoda « *useState* » se poate observa în figura 2.17.

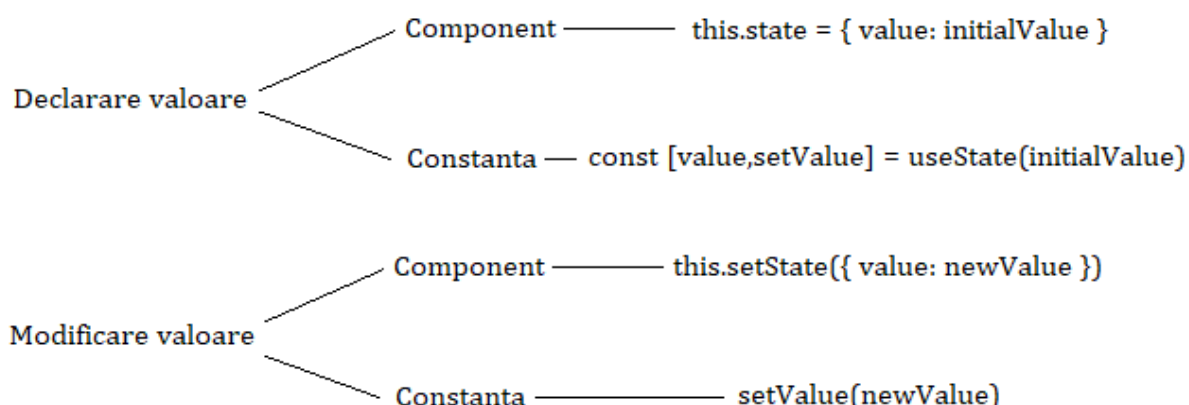


Figure 2.17 Schema de conversie a state-ului

Pentru a substitui metodele « *componentDidMount* » și « *componentDidUpdate* » existente în ciclul vieții unei componente am folosit metoda « *useEffect* » bazată pe funcții asincrone. Astfel la fiecare modificare funcțională a unui element funcția « *useEffect* » va fi executată. Implementarea asincronă este folosită în acest caz deoarece împiedică apariția unei bucle infinite care să apară în cazul unor modificări persistente asupra elementului principal sau a unui apel recursiv.

Integrarea unei funcții « *useEffect* » în cadrul unui element este reprezentată în figura 2.18.

```
useEffect(() => {
  async function fetchData() {
    const result = await axios(Achievements_URL);
    setAchievements(result.data);
    getCurrentPoints();
  }
  fetchData();
}, []);
```

Figure 2.18 Implementarea funcției « *useEffect* » cu comportament asincron

După cum se poate observa mai sus, funcția `fetchData` în care se face apelul către « `ACHIEVEMENTS_URL` » este făcut cu expresia « `await` ». Expresia « `await` » realizează o pauză în execuția funcției asincrone până « Promise-ul » apelului este rezolvat.

****Promise** – procesul de execuție și completare al unui apel asincron.

CORPUL FUNCȚIILOR CONSTANTE

Funcțiile constante au o structură bine definită restricționată după o anumită arhitectură impusa în pasul pre-analizei.

Implementarea obiectelor ce constituiesc componentele principale ale aplicației respectă în totalitate structura observabilă în *figura 2.19*:

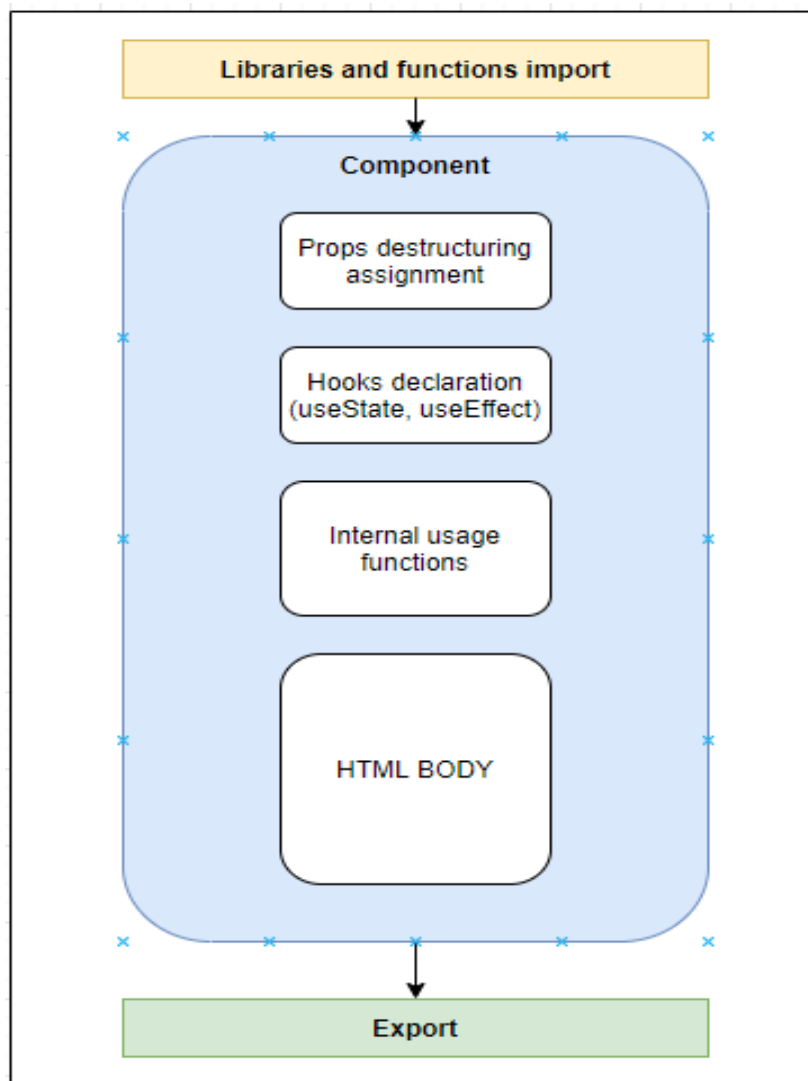


Figure 2.19 Structura generală a componentelor

Pentru a exemplifica teoria de mai sus într-o manieră practică o voi explica ilustrând-o printr-un pseudocod JavaScript :

- 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 funcțion = () => { operatii ; return rezultat ; }`
- HTML BODY
`const Component = () => { return HTML_ELEMENTS ; }`

- Export

```
export default withStyles(componentStyle)(Component) ;
```

3.5 Interfața grafică

Aplicația are în componența sa 2 interfețe grafice în etapa de înregistrare a utilizatorului și 8 interfețe grafice în cadrul etapei post-înregistrare.

După etapa de înregistrare în aplicație se va genera o cheie de accesare va verifica și va valida accesul la fiecare rută privată.

3.5.1 Login/Register

3.5.1.1 Login

Pagina de logare în aplicație este pagina introductivă regăsită pe ruta simplă « / » și pe ruta « /login » și are aspectul ilustrat în *figura 2.20*.

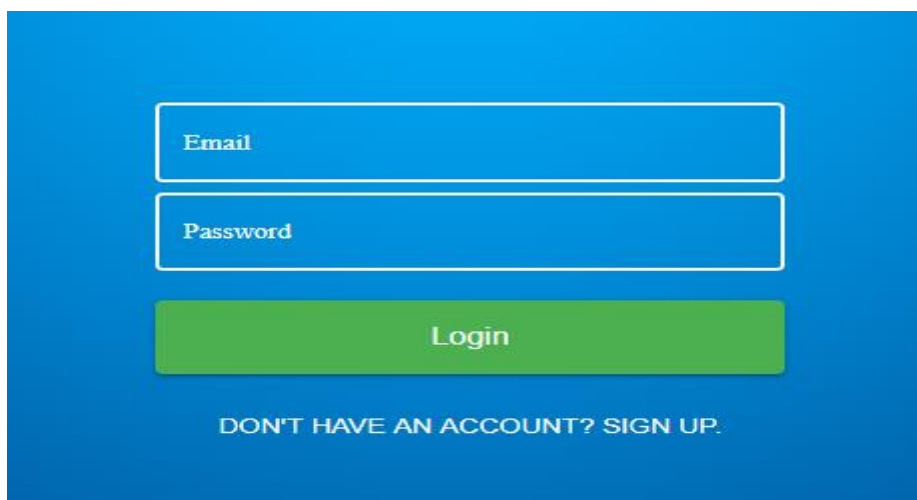


Figure 2.20 Design-ul paginii de login

Structura este relativ simplă, alcătuită din două câmpuri cu o validare implementată individual prin următoarele metode :

- **Email:** Validarea se face printr-un regex complex ce validează structura textului introdus în mod succesiv. Regexul de validare este :
`/^([<>@\\[\]\\\.\:\s@"]+(\.[^<>@\\[\]\\\.\:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/.`
- **Password:** Validarea se face mai simplu doar prin verificarea lungimii textului din câmp. Se verifică dacă utilizatorul a completat câmpul, iar în caz afirmativ se impune o lungime a textului mai mare de 8 caractere.

În cazul în care datele din cele doua câmpuri nu sunt valide se vor afișa următoarele erori sugestive tratate după cazurile:

- email greșit – « Email is not registered !»
- parolă greșită – « Wrong password ! »
- datele sunt introduse, dar nu trec de validare - « Invalid data »
- câmpurile nu au fost completate – « All fields are required ! »

Erorile sunt afișate în snackbox-uri temporare.

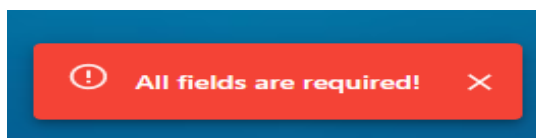


Figure 2.21 Modelul unui « snackbox » de tip eroare

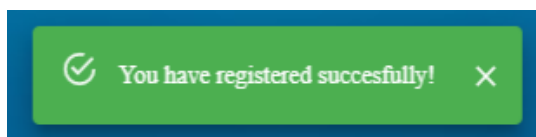


Figure 2.22 Modelul unui « snackbox » de tip success

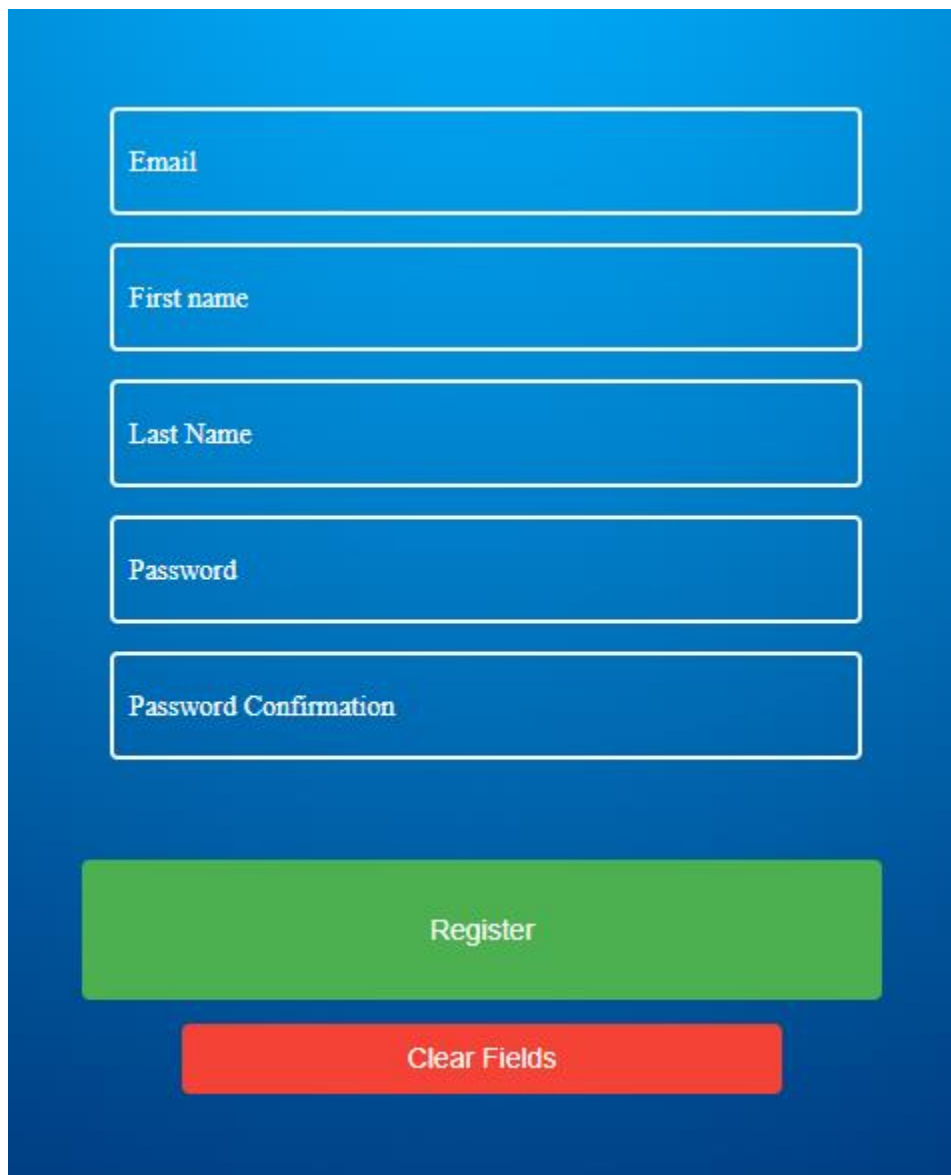
În cazul în care datele introduse sunt corecte, iar apelul către baza de se realizează fără impedimente mesajul afișat va fi « Login succesfully ! » într-un snackbox de tip confirmare.

Butonul aflat succesul câmpurilor de date va realiza validarea și respectiv intrarea în aplicația propriu-zisă la apăsarea lui. Textul aflat sub buton are implementat un efect de tip hover, iar la apăsarea pe el va redirecționa utilizatorul către pagina de înregistrare.

La expirarea cheii de acces sau la pierderea conexiunii de acces către rutele private redirecționarea se va face către această pagină introductivă.

3.5.1.2 *Register*

Pagina de înregistrare a utilizatorului în cadrul aplicației este aflată pe ruta publică « /register » și aspectul ilustrat în *figura 2.23*.



The image shows a registration form on a dark blue background. It consists of five white input fields stacked vertically, each with a label inside: 'Email', 'First name', 'Last Name', 'Password', and 'Password Confirmation'. Below these fields is a large green button labeled 'Register'. At the bottom is a smaller red button labeled 'Clear Fields'.

Figure 2.23 Designul paginii de înregistrare

Interfața paginii de înregistrare are un comportament similar celei de login. Câmpurile sunt validate la apăsarea butonului « register », iar snackbar-urile aruncă erori specifice la fel că și în pagina precedentă.

Ulterior introducerii unor date valide în cele 5 câmpuri și apăsării, snackbox-ul din *figura 2.22* va fi observabil în partea inferioară a paginii, iar noul utilizator va fi adăugat în baza de date.

Butonul « Clear Fields » adițional are rolul de a oferi utilizatorului oportunitatea de a elimina valorile și erorile din câmpuri.

În partea de sus a paginii avem și o săgeată cu efect hover ce are scopul de a redirecționa utilizatorul către pagina de login.

3.5.2 Header/content/footer

3.5.2.1 Header

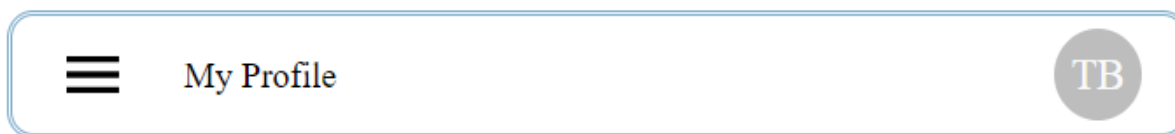


Figure 2.24 Designul unui header

Header-ul este compus din 3 componente :

- drawer-ul : ajută la o mai bună navigare în cadrul aplicației deschizând temporar un meniu ce îi permite utilizatorului să parcurgă către orice rută privată din aplicație. Rutele puse la dispoziție în cadrul drawer-ului sunt:

- Future events
- Past Events
- My Events
- Voted Topics
- Leaderboard
- About
- Logout

Primele 5 elemente din cadrul drawer-ului sunt despărțite de ultimele două printr-un « Divider » :

- titlul paginii : variabil de la o pagină la alta afiseaza utilizatorului pagina curentă
- avatarul utilizatorului : oferă opțiunea de redirectionarea către profilul personal

În centrul acestui avatar vor fi afișate inițialele utilizatorului preluate după procesul de înregistrare. Cele două nume vor fi concatenate la crearea profilului creând un string ce reprezintă numele întreg al utilizatorului pe care se va executa următoare operație de preluare a inițialelor folosind un regex :

« `toUpperCase()` » – transformă stringul inițial într-un string de majuscule.

« `match()` » – extrage caracterele dintr-un string ce respectă expresia impusă de regex.

« `join()` » – grupează un șir de string-uri într-un string final format din concatenarea tuturor elementelor din șir

```
return fullName
    .toUpperCase()
    .match(/(\b[A-Z](?!\s))/g)
    .join("");
```

Figure 2.25 Funcția de extragere a inițialelor

Așa cum se poate observa în figura precedentă apelarea în lanț a unor funcții definite sau importate este echivalentă cu apelarea pe rând în mod iterativ și nu impactează funcționalitatea sau rezultatul final al operațiilor efectuate.

3.5.2.2 *Content*

Forma fiecărei paginii este împărțită în 10% header, 80% content, 10% footer. Așadar, partea care se afla în proporție cea mai ridicată în cadrul unei pagini este conținutul propriu-zis.

Schema principală a paginii are conținutul integrat, dar componenta lui este deependentă de ruta curentă deoarece conținutul este în permanență variabil.

```
<Content>{children}</Content>
```

Figure 2.26 Implementarea componentei de continut

« {children} » - reprezintă aici elementul integrat în conținut

În timp ce Header-ul are în componență doar titlul dinamic, iar celelalte elemente constante, Content-ul are componentele nu doar dinamice, dar și temporare.

Content-ul se malează în funcție de componentul principal existent și de locația curentă în aplicație.

La apariția unor formulare « externe » tranziția nu este făcută în afara conținutului, ci în interiorul acestuia, conținutul aferent devenind opac, iar accentul punandu-se pe formularul lansat de acțiunea curentă.

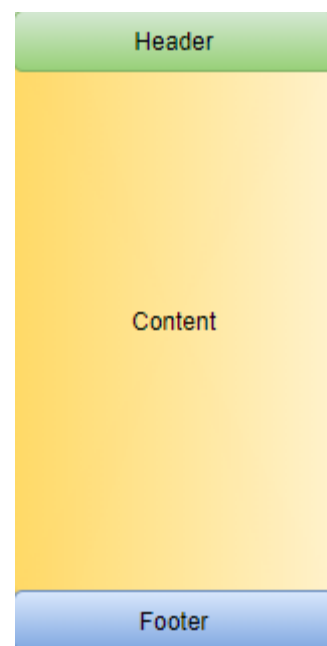
Așadar, funcționalitatea principală este creată din componente ce au implementate anumite acțiuni și funcții precum și un design propriu, iar funcționalitatea este integrată în conținut.

Structura fixă impusă este extrem de utilă în determinarea locației unei nefuncționalități, dar și pentru o telemetrie mai bună deoarece copilul aferent conținutului este monitorizat în mod constant.

3.5.2.3 *Footer*

Cea mai simplă componentă din cardul paginii. Nu se modifică indiferent de ruta curentă având un comportament static.

Are o culoare aflată în contrast cu header-ul și content-ul, iar scopul principal este de a delimita pagina și a crea un design uniform.



**Figure 2.27
Schema principală a
paginiilor**

3.5.3 Profile/ leaderboard /achievements

3.5.3.1 Profile

Antecedent procesului de logare în aplicație utilizatorul va fi întâmpinat de pagina profilului personal care este constituită din mai multe componente principale divizate după structura prezentată mai sus.

Conținutul paginii este format din mai multe elemente caracteristice.

- « summary » : este alcătuită din 3 elemente secundare :
 - primul element : numărul de puncte acumulate de utilizatorul curent
 - al doilea element: punctele necesare pentru dobândirea următorului premiu
 - al treilea element : premiul precedent câștigat
- « leaderboard » & « achievements » : direcționează utilizatorul către pagina de leaderboard și respectiv achievements.

58 Points

Next Achievement: 75 points

National Tech Conference

Leaderboard

Achievements

- « feedback points »: așa cum se poate observa în imaginea alăturată istoricul de feedback este reprezentat de o lista de elemente care au structura de tipul:
« +[puncte][tip] [data] ».
Aferent istoricului de feedback avem și un text « Load more » cu efect de tip hover care încarcă lista în mod iterativ adăugând un număr prestabilit de elemente.

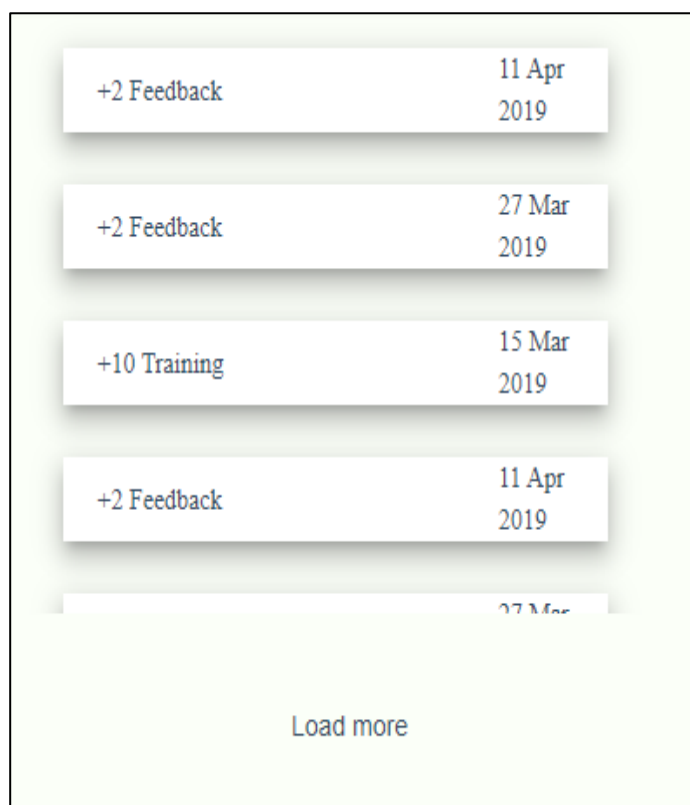


Figure 2.28 Lista feedback

3.5.3.2 *Leaderboard*

Pagina de « Leaderboard » constituie topul general al aplicației. Fiecare utilizator poate să găsească aici situația lui raportată la ceilalți utilizatori.

Structura paginii este relativ simplă, compusă dintr-o listă cu elemente de tipul:



Figure 2.29 *Structura elementelor în pagina « Leaderboard »*

[nume_intreg]

[puncte_acumulate]

Deasupra topului se află o imagine reprezentativa conținutului paginii.

3.5.3.3 *Achievements*

În pagina « Achievements » avem o listă cu premiile acordate sau care se pot acorda în cazul acumulării unui număr necesar de puncte.

Elementele din lista au culoarea de fundal verde dacă au fost dobândite

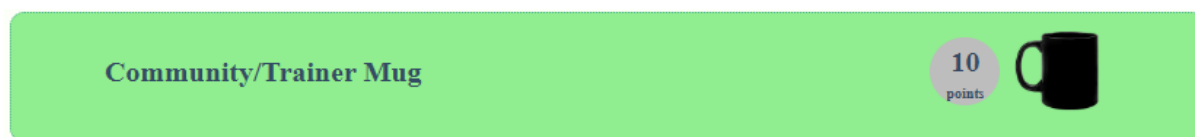


Figure 2.30 *Scheletul elementului unui premiu castigat*

și respectiv culoarea de fundal albă dacă nu au fost câștigate de utilizatorul curent



Figure 2.31 *Scheletul elementului unui premiu necastigat*

Structură : [premiul]

[puncte_necesare]

Acest comportament este implementat pentru a asigura utilizatorului o privire de ansamblu printr-un mod mai simplu.

3.5.4 Future events/Past events/My events

3.5.4.1 Future events

Scopul paginii « Future events » este de a afișa totalitatea evenimentelor ce vor avea loc în viitor indiferent dacă utilizatorul este înscris sau nu la ele.

Arhitectura paginii are în componenta un și un mesaj sugestiv aflat sub header-ul principal. :

This is a list of all future events. You can choose to unsubscribe. Please do this as soon as you know that you won't be able to participate, so we can notify the people in the waiting list.

Figure 2.32 Mesaj introductiv pagina « Future events »

Lista evenimentelor viitoare este compusă din item-uri ce au forma ilustrata în figura 2.32:



Figure 2.33 Structura elementelor din « Future events »

Avatarul (« PY ») conține limbajul de programare ales ca temă principală de prezentare.

« Python basics » – Titlul evenimentului

« Presentation(Expert) – 1h » – Tipul evenimentului, dificultatea acestuia și respectiv durata totală.

« You are in the waiting list » - Unul din stadiile în care se află utilizatorul raportat la eveniment. Există 3 stadii :

- « Subscribe » - când utilizatorul nu este înscris la eveniment, iar apăsarea pe text va realiza înscrierea acestuia la eveniment o
- « You are in the waiting list » - utilizatorul dorește înscrierea la eveniment, dar va fi adăugat într-o lista de așteptare deoarece locurile sunt limitate o
- « Unsubscribe » - utilizatorul este înscris la eveniment, iar prin apăsare el va fi eliminat din lista de înscrieri

« 27 Jul '20 10 :00 » - Data și ora la care va avea loc evenimentul.

Datele din listă « Future events » vor fi filtrate și verificate folosind librăria « moment », iar dacă data unui eveniment este surclasata în timp live evenimentul va fi transferat automat în lista « Past events ».

Validarea datei evenimentului prin comparare cu data curentă se va face folosind conversia din tipul de date în tipul number reprezentat prin timestamp și rezultat apelând funcția « getTime » care va face posibilă compararea cu data curentă conversionata prin moment.

3.5.4.2 *Past events*

Așa cum am menționat mai sus, pagina « Past events » are o arhitectură echivalentă cu cea a paginii « Future events » având urmatorul mesaj caracteristic :

This is a list of all the past events where you participated. Please don't forget to leave a quick Feedback and you will be rewarded! There are 5 more events that require your feedback.

Figure 2.34 Mesaj introductiv pagina « Past events »

Cifra cu fontul bold din textul de mai sus reprezintă numărul de evenimente la care utilizatorul are posibilitatea să ofere un feedback ulterior.

Forma iteme-lor din listă conținută în « Past events » este diferențiată de cea din « Future events » prin substituirea celor 3 stadii cu 2 stadii noi:

Feedback - primul stadiu al evenimentelor care se află în lista « Past events ».

Prin apăsarea butonului « Feedback » se declanșează acțiunea de deschidere a unui form caracterizat de designul din figura:






					
Clarity of fullPage/Content	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Originality of the presentation	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Complexity of the subject	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Engagement with the audience	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Cursive flow, good pace	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure 2.35 Formularul de feedback

Având un stil interactiv, formularul de feedback permite utilizatorului o evaluare convenabilă a evenimentului precedent bazându-se pe 5 criterii evocatoare.

Se poate observa în figura 2.35 ca valoarea initiala a câmpurilor din formularul de feedback este prestabilita pentru fiecare categorie.

Thank you! - cel de-al doilea stadiu al evenimentului este stadiul post acordării de feedback. Starea evenimentului nu mai poate fi schimbată după acordarea de feedback.

3.5.4.3 *My events*

Mesajul aflat în partea superioară a paginii :

This is a list of all future events to which you've subscribed. You can choose to unsubscribe. Please do this as soon as you know that you won't be able to participate, so we can notify the people in the waiting list!

Figure 2.34 Mesaj introductiv pagina « My Events »

Evenimentele filtrate în pagina « My events » sunt preluate de pe același URL de evenimente. Datele apărute în lista de « My events » sunt condiționate de utilizator, iar lista are în componență evenimente la care utilizatorul s-a înscris sau se află pe lista de așteptare.

3.5.5 Topics voting

Pagina de votare a temelor este una dintre cele mai importante pagini ale aplicației deoarece fiecare eveniment pornește inițial de la stadiul de tema propus. Interfața grafică este constituită dintr-un buton care efectuează acțiunea de deschidere a formularului de propunere a unui tema și de o listă a temelor ordonată în funcție de voturile acumulate în fiecare tema.



Figure 2.36 Structura elementelor din « Topics voting »

Efectul de apariție al fiecărui item este implementat folosind sublibrariile « useSpring » și « animated » din librăria principală « react spring ».

```
const animation = useSpring({
  width: "90%",
  margin: "auto",
  height: "100%",
  from: { height: "20%", width: "50%" },
  config: {
    mass: 3,
    tension: 350,
    friction: 30,
  },
  border: "1px dotted rgba(28,110,164,0.45)",
  borderRadius: "10px",
});
```

Figure 2.37 Animatie de extindere prin « useSpring »

Numărul alcătuitor primului element este numărul de voturi total acumulat de tema respectiv.

Numărul definește importanța și prioritatea temei în listă.

Al doilea element este compus din titlul și respectiv descrierea temi.

Cel de-al treilea element este reprezentat de votul actual al utilizatorului curent raportat la tema din listă. Există 3 stări ale acestui element :

- « vote up » : incrementarea numărului de voturi
- « vote down » : decrementarea numărului de voturi
- « neutral » : utilizatorul nu a votat tema



Starea temei raportate la fiecare utilizator rămâne salvată, dând astfel posibilitatea utilizatorului să controleze și să modifice permanent decizia luată.

În partea superioară a listei de teme este butonul de propunere al unei teme.

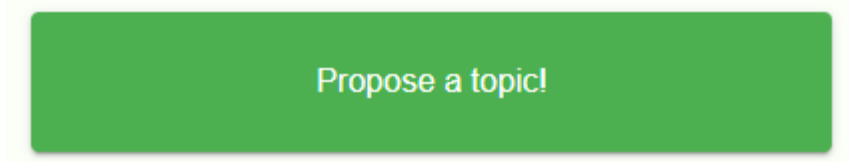


Figure 2.38 Designul butonului de propunere a unei teme

Formularul de propunere al unei teme este compus din mai multe elemente care să ajute la o cât mai bună organizare a subiectului care aferent poate să devină eveniment.

Modalul formularului este importat din librăria « react-animated-modal ». Apariția modalului este realizată printr-un efect de tip « rotateIn », iar vizibilitatea este gestionată printr-o variabilă din state.

```
<Modal
  visible={this.state.open}
  closemodal={this.handleClick}
  type="rotateIn"
>
```

Figure 2.39 Definirea unui modal cu animație

Elementele definitorii ale formularului de propunere a temelor sunt :

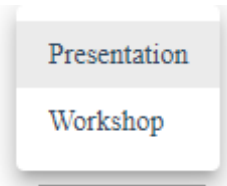
- titlu & subtitlu : **Propose a topic**

Propose a topic that you would like to present or if you would like to be presented by some else.

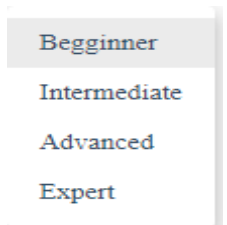
- campuri de completare :

- camp I : un checkbox care determină dacă utilizatorul care propune tema va prezenta/nu va prezenta evenimentul

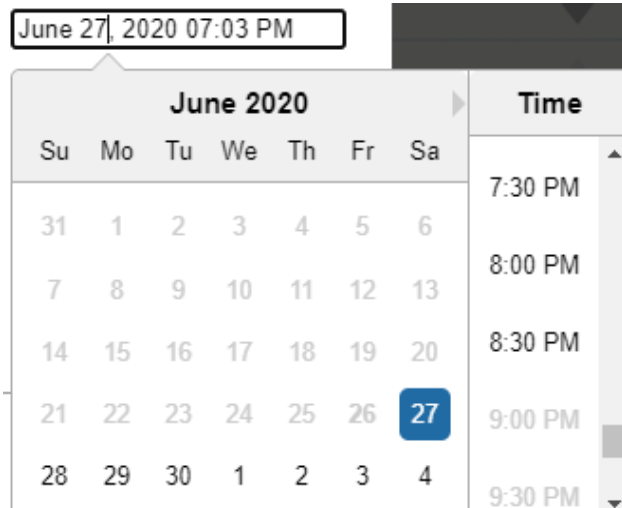
- camp II : dropdown din care se va alege tipul temei



- camp III : dropdown care va determina dificultatea/nivelul prezentării :



- camp IV : în cadrul acestuia se află un datepicker prin care utilizatorul va alege data și ora prezentării temei. Elementul « Datepicker » este configurat cu anumite restricții ce împiedică o alegere incorectă a datei de prezentare.



- camp V : dropdown care va afișa opțiunile de durată a evenimentului:
 - 30min
 - 1h
 - 1h30min
 - 2h
- camp VI : dropdown cu opțiuni de alegere a unui limbaj de programare:
 - JS
 - Java
 - PHP

- .Net
 - C
 - C++
- titlul temei: un câmp cu restricții impuse și cu o limită de 100 de caractere ce va constitui titlul evenimentului
- detalii tema : similar titlului are ca diferență numărul de caractere limită 300.

Topic title

100 characters left

Topic details

Ex. If it's a programming language, how new is it, what type is it (static/dynamic, interpreted/compiled). Is it good because it's performant or is it good because it's flexible?

300 characters left

După completarea câmpurilor din formular utilizatorul va putea să-și înregistreze tema în lista prin apăsarea butonului « send », iar ulterior lista temelor se va popula.

3.5.6 About page

Pagina de informații se află pe o rută publică, prin urmare poate fi accesată de orice utilizator indiferent dacă el deține sau nu un cont în aplicație.

Structura paginii de informații are următorul conținut :

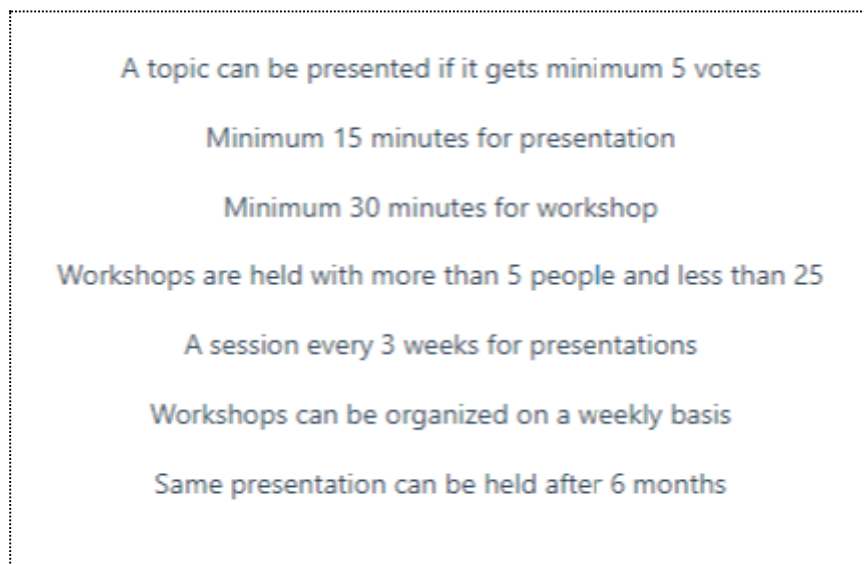


Figure 2.40 Reguli în corpul paginii « About page »,

În drawer-ul temporar avem o opțiune de logout care redirectionează utilizatorul către pagina de login eliminând din memoria aplicației cheia de acces.

3.5.7 New event constrcutur

Constructorul de creare al unui eveniment se gaseste în folderul « utils ».

O tema care a acumulat mai mult de 10 voturi se converzioneaza automat intr-un eveniment viitor prin intermediul unui pseudo-constructor prin urmasorii pasi :

- destructurarea proprietatilor transmise: `const { topicTitle, topicType, topicDate, programmingLanguage, topicDuration, difficultyType, timestamp, userId, isUserPresenter } = props;`
- stocare proprietatilor intr-un obiect nou prin câmpuri diferite facandu-se conversia prin atribuire:
 - o `name -> topicTitle`
 - o `type -> topicType`
 - o `date -> topicDate`
 - o `lang -> programmingLanguage`
 - o `duration -> topicDuration`
 - o `difficulty -> difficultyType`
 - o `timestamp -> timestamp`
 - o `attendanceIds, waitingListIds, feedback -> []`
- transmiterea unui câmp aditional în cazul în care utilizatorul este prezentatorul viitorului eveniment prin verificare conditiei : `if(isUserPresenter)`
 - o `userId -> userId`
- la final se returneaza obiectul nou creat care va fi trimis către server ca un nou eveniment avand toate specificatiile necesare

3.5.8 Private route

Crearea unei rute private se bazeaza pe mai multe structuri conditionale care se bazeaza pe verificare accesului în memorie.

Ruta privata presupune un acces limitat și specific și poate fi accesata doar prin trecerea tuturor conditiilor impuse.

```
export const PrivateRoute = ({ component: Component, ...rest }) =>
  Cookie.get("token") ?
    <Route {...rest} render={ (props) => {
      return(
        Cookie.get("token")
          ? <Component {...props} />
          : <Redirect to="/" />
        )
      }
    } />
    :
    <Redirect to="/" />
```

Figure 2.41 Algoritm implementare unei rute private

Componentele Route și Redirect sunt importate din biblioteca « react-router-dom » și au scop de directionare către o ruta specifică care să se actualizeze condiționat de anumite proprietăți și respectiv de redirectionare.

3.6 Securitate

3.6.1 Cookies acces

Cookies sunt niște date care permit stocarea în aplicație a unor informații de acces. Sunt importante din biblioteca « universal-cookie ».

Setarea cheii de acces în memorie se face folosind funcția « set » prin intermediul căreia se poate stabili și o perioadă de disponibilitate.

```
Cookies.set("token", currentToken, { expires: IN_ONE_HOUR });
```

Figure 2.42 Setarea unei chei de acces în memorie

Preluarea/verificarea existenței cheii de acces în memorie se realizează prin funcția « get »

```
new Cookies().get("token");
```

Figure 2.43 Verificarea existenței în memorie a unei chei de acces

Cheia de acces este generată în timpul primului proces de logare și se reține în baza principală de date pentru a oferi accesul ulterior. Fiecare cheie de acces este unică pentru fiecare utilizator și este reprezentată de câmpul id.

Este păstrat în memorie într-un tabel definitiv alcătuit din câmpurile ilustrate în figura 2.44 :

Name	Value	Domain	Path	Expires / Max-Age
token	_7h67hedv	localhost	/	2020-06-28T10:30:31.000Z

Figure 2.44 Pastrarea în memorie a token-ului

După login, accesul este permisiv utilizatorului pentru o perioadă de o oră, aferent accesul la rutele private fiind oprit, iar token-ul dispăre din memorie.

3.6.2 Criptare/decriptare

Criptarea sau ascunderea codului de software este folosit în protecția copierii de software împotriva ingineriei inverse, analiza aplicațiilor neautorizată, crack-uri și pirateria software.

Pentru restaurarea datelor originale de la cele codate se aplică procesul de decriptare.

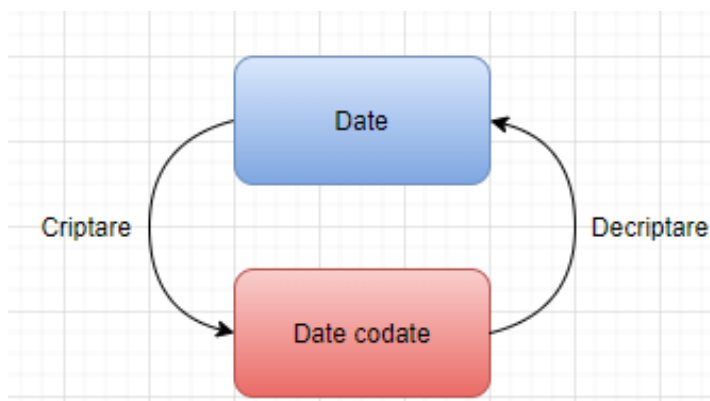


Figure 2.45 Schema de definire a criptării și decriptării

Pentru convertirea datelor într-o bază specificată se poate folosi clasa « Buffer » inclusă în NodeJS.

Criptarea :

```
Buffer.from(password).toString("base64")
```

Decriptarea :

```
Buffer.from(password, "base64").toString()
```

3.7 Interfața în ansamblu

În acest subcapitol se vor ilustra paginile complete cu design distinct pentru a ilustra o privire de ansamblu a aplicației.

Făcând abstracție de paginile de login și register și de formularele și mesajele adiționale afișate aplicația este compusă din următoarele pagini aflate într-o ordine succesivă:

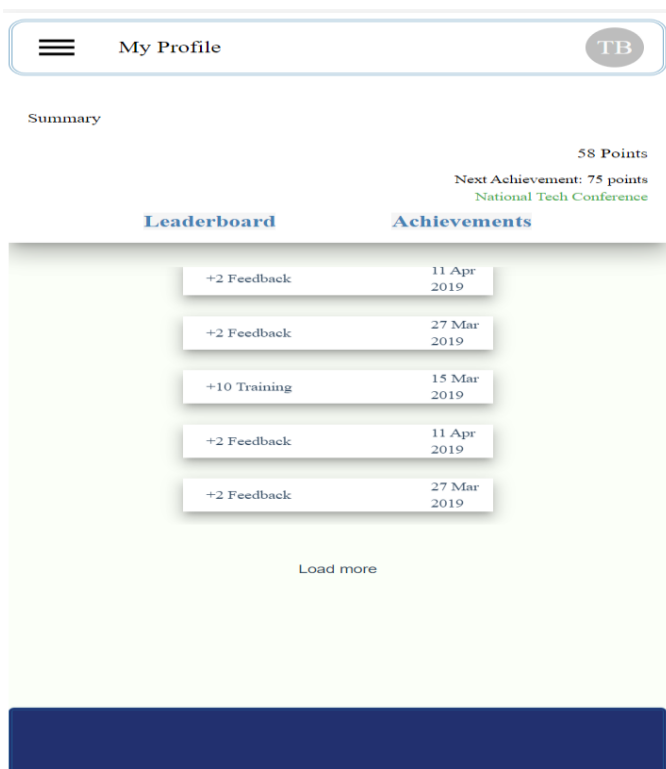


Figure 2.46 Pagina « My Profile »



Figure 2.47 Pagina « Achievements »

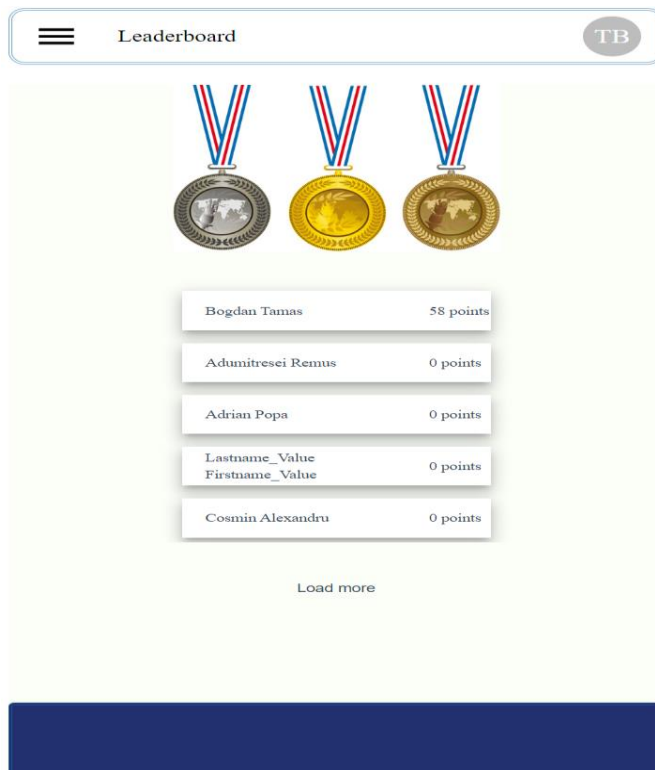


Figure 2.48 Pagina « Leaderboard »

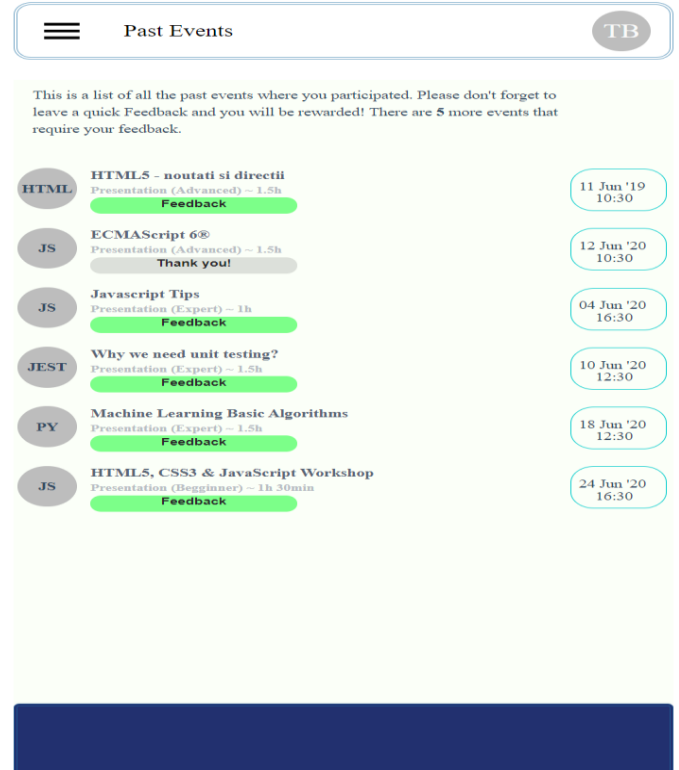


Figure 2.49 Pagina « Past events »

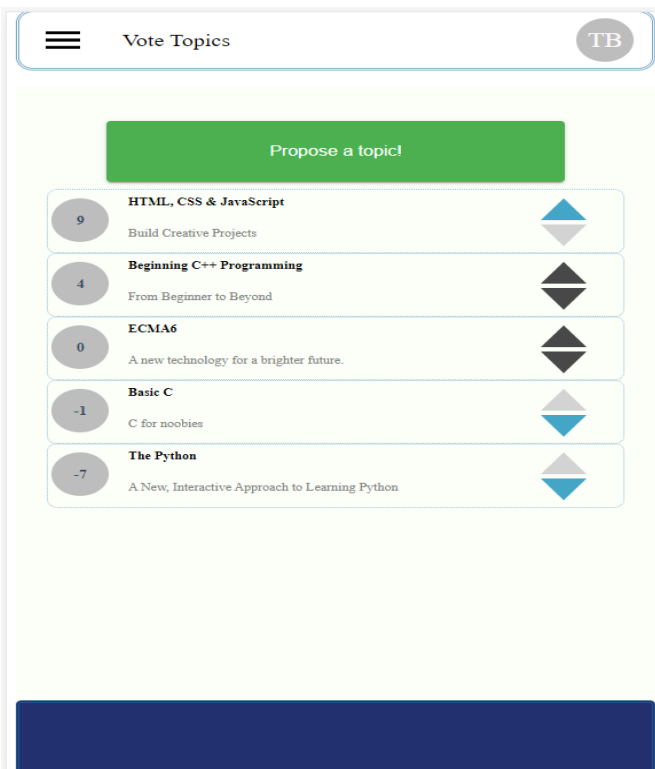


Figure 2.50 Pagina « Vote topics »

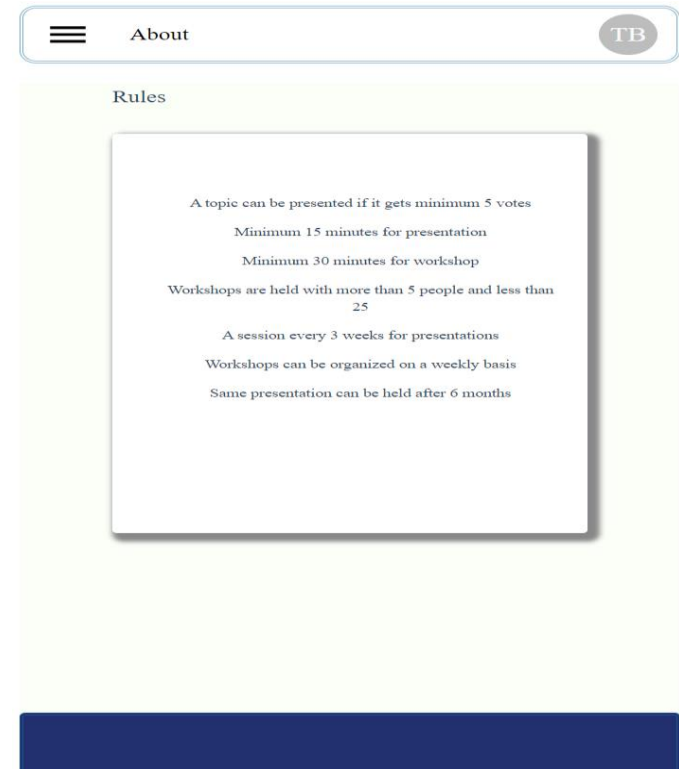


Figure 2.51 Pagina « About »

4 Rezultate și validare

Rezultatele obținute post-implementării sunt satisfăcătoare, iar performanțele sunt îndeplinite, respectandu-se restricțiile și limitele impuse în etapa de proiectare.

Apelurile către cele două baze de date sunt făcute în mod eficient având o durată medie redusă.

4.1.1 Apel MYSQL

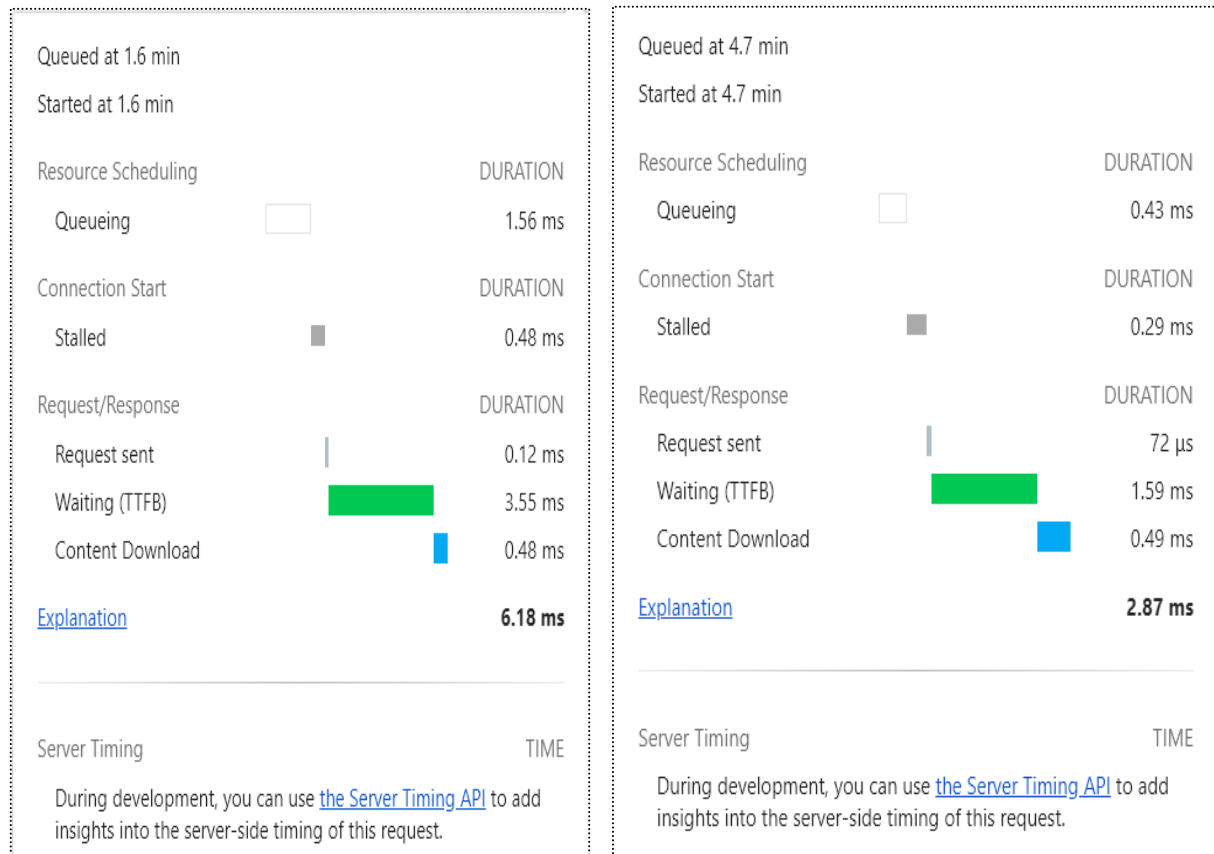


Figure 3.1 Rezultate apel tip « get » baza MySql

Figure 3.2 Rezultate apel tip « post » baza MySql.

După cum se poate observa în figurile de mai sus specificațiile impuse privind durata medie a apelurilor sunt îndeplinite.

Raportând durata medie la cele două figuri de mai sus rezultă o durată medie de **4.525 ms**.

4.1.2 Apel JSON Server

Durata apelurilor de tip get/patch

```
GET /pointsReceived 304 4.713 ms - -
GET /achievements 304 4.394 ms - -
GET /events?_expand=users 304 4.931 ms - -
GET /events?_expand=users 304 4.335 ms - -
GET /events/2 200 5.034 ms - 258
PATCH /events/2 200 147.184 ms - 422
GET /events?_expand=users 200 4.396 ms - -
GET /proposedTopics 304 3.139 ms - -
```

Figure 3.3 Durata apelurilor de tip get/patch către serverul

```
POST /proposedTopics 201 5.763 ms - 379
```

Figure 3.4 Durata apelurilor de tip post către serverul local

4.1.3 End-to-end testing

Deoarece codul sursă este format în preponderență din componente, constante și containere cu un comportament vizual a fost necesară o verificare manuală amănunțită a flow-ului aplicației.

Verificare și validarea de designului și funcționalității s-a realizat printr-o parcurgere a aplicației end-to-end. Bug-urile apărute pe parcursul verificărilor au fost reparate și retestate aferent. Procesul de testare end-to-end s-a realizat urmând pașii :

- Înregistrarea utilizatorului în aplicație testând validarea pe fiecare câmp din procesul de înregistrare precum și înregistrarea succesivă.
- Intrarea în aplicație testând câmpurile principale și comunicarea cu ruta './register'.
- După intrarea accesului în aplicație se verifică prezența token-ului de acces.
- Se validează structura paginii de profil precum și prezența în mod individual a tuturor datelor incluse în componentele paginii.
- Durata apelului de populare a listei de feedback este observabilă prin loader-ul circular.
- Aferent încărcării listei de feedback se apasă pe butonul « Load more » și se confirmă popularea listei în mod asincron.
- Se apasă ulterior pe butonul « Achievements » și se confirmă corectitudinea culorilor de fundal ale elementelor raportate la punctele acumulate.
- Ulterior se accesează succesiv primele 5 pagini conținute în drawer-ul temporar pentru a valida gestionarea rutelor, iar apoi se revine la prima pagină.
- Se verifică reacția fiecărei acțiuni prin apăsarea butonului « subscribe » și respectiv « unsubscribe » pentru a se verifica dacă mesajul afișat este corespunzător.
- Similar pentru paginile « My Events » și « Past Events » se validează corectitudinea mesajului din dialog și respectiv formul de feedback.

- În pagina « Vote Topics » se va testa acordarea individuală a voturilor pentru fiecare temă, iar apoi se va apăsa pe butonul « Propose a topic ! ».
- După apariția formularului de propunere a unei teme se vor testa câmpurile de tip checkbox, dropdown și date modificând succesiv valorile. Se vor testa câmpurile input și respectiv actualizarea caracterelor rămase disponibile în fiecare câmp. Se încearcă și o închidere « accidentală » a formularului pentru a putea observa dacă datele sunt reținute în memorie la o deschidere a formularului ulterioară.
- Se accesează pagina de « Leaderboard » și se verifica dacă lista se populează asincron prin apăsarea butonului « Load more ». Popularea listei poate fi observată prin expansiunea slidebar-ului aflat în dreapta listei.
- După verificare cu atenție a pașilor de mai sus se va accesa pagina « About » din drawer cu mențiunea că accesul pe ruta privată este acum disponibil.
- Se apasă pe butonul de logout.
- Se accesează din nou ruta «. /about » fără ca accesul pe ruta privata să fie disponibil și se confirmă tipul rutei

4.1.4 API testing

Testarea API este un tip de testare software ce implică testarea interfețelor de programare a aplicațiilor direct și ca parte a testării integrării pentru a determina dacă îndeplinesc așteptările privind funcționalitatea, fiabilitatea, performanța și securitatea

Ulterior procesului de testare end-to-end flow s-a testat și corectitudinea populării câmpurilor din severul JSON Server prin realizarea unor apeluri directe folosind « Postman ».

Postman este o platforma de colaborare pentru dezvoltarea interfetelor aplicatiilor. Model inserare date în baza de date prin postman.

KEY	VALUE
<input checked="" type="checkbox"/> firstname	Firstname_Value
<input checked="" type="checkbox"/> lastname	Lastname_Value
<input checked="" type="checkbox"/> email	Email@Value.com
<input checked="" type="checkbox"/> password	Password_Value

Figure 3.5 Model de inserare date prin postman

Fiecare parametru conține un câmp cheie și o valoare specifică. Prin completarea unui câmp acesta va fi automat integrat în link-ul de mai sus. Link-ul reprezintă ruta pe care se va face apelul. Acțiunea dorită (get/post/patch/...) se va alege din lista aflată în dreapta link-ului.

După completarea parametrilor necesari apelului se va efectua apelul apăsând pe butonul send.

Rezultatul va fi afișat în postman în funcție de reușita sau eșecul apelului.

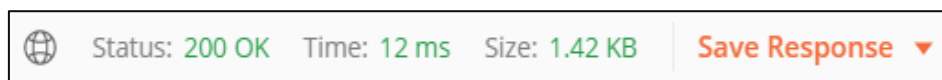


Figure 3.6 Apel reușit în postman



Figure 3.7 Apel eșuat în postman

Statusurile afișate pentru apelurile eșuate pot fi :

- 400 = HTTP_BAD_REQUEST
- 401 = HTTP_UNAUTHORIZED
- 403 = HTTP_FORBIDDEN
- 405 = HTTP_METHOD_NOT_ALLOWED
- 408 = HTTP_REQUEST_TIMEOUT

4.1.5 Unit testing

Testarea unitară se referă la scrierea unor bucăți de cod, denumite cod de testare, care validează codul de producție. Testarea aplicației devine așadar automată.

Funcțiile principale de manipulare (helpers) a datelor au fost verificate prin implementare unor algoritmi de testare unitara folosind JEST.

Jest – este o librărie JavaScript de creare, rulare și structurarea testelor unitare.

Configurarea librăriei într-o aplicație oarecare se face prin adăugarea următoarei opțiuni în secțiunea « scripts » inclusă în « package.json ».

```
"test": "jest",
```

Fișierele de tip teste au extensia « spec.jsx ».

```
getDateFormat.spec.jsx
getFonts.spec.jsx
getInitials.spec.jsx
getTimeFromStamp.spec.jsx
```

Figure 3.8 Fișierele de teste


```
it("should return the correct date", () => {
  const date = new Date("2020-07-20T10:00:00.614Z");
  expect(getDateFormat(date)).toEqual("July 20, 2020 13:00");
});

it("should return an error if the date is invalid", () => {
  expect(getDateFormat(INVALID_DATE)).toEqual(new Error(INVALID_DATE));
});
```

Figure 3.9 Teste din fișierul « getDateFormat »

```
it("should return the corect initials", () => {
  const width_1 = 1500;
  const width_2 = 500;
  const width_3 = 4000;

  expect(fontTitle(width_1)).toEqual(17);
  expect(Math.floor(fontTitle(width_2))).toEqual(13);
  expect(fontTitle(width_3)).toEqual(17);
  expect(fontSubtitle(width_1)).toEqual(15);
  expect(Math.floor(fontSubtitle(width_2))).toEqual(11);
  expect(fontSubtitle(width_3)).toEqual(15);
  expect(fontBottom(width_1)).toEqual(13);
  expect(Math.floor(fontBottom(width_2))).toEqual(9);
  expect(fontBottom(width_3)).toEqual(13);
});
```

Figure 3.10 Teste din fișierul « getFonts »

```
it("should return the corect initials", () => {
  const fullName_1 = "Bogdan David";
  const fullName_2 = "bogdan david";
  const fullName_3 = "Bogdan david";
  const fullName_4 = "bogdan David";

  expect(getInitials(fullName_1)).toEqual("BD");
  expect(getInitials(fullName_2)).toEqual("BD");
  expect(getInitials(fullName_3)).toEqual("BD");
  expect(getInitials(fullName_4)).toEqual("BD");
});
```

Figure 3.11 Teste din fișierul « getInitials »

```
it("should return the correct date from timestamp", () => {
  const timestamp_1 = 1591288200;
  const timestamp_2 = 1596123000;
  const timestamp_3 = 1595014200;
  expect(getTimeFromStamp(timestamp_1)).toEqual("16:30");
  expect(getTimeFromStamp(timestamp_2)).toEqual("15:30");
  expect(getTimeFromStamp(timestamp_3)).toEqual("19:30");
});

it("should return an error if the date is invalid", () => {
  const timestamp_error = 1414634759012000000;
  expect(getTimeFromStamp(timestamp_error)).toEqual(new Error(INVALID_TIMESTAMP));
});
```

Figure 3.12 Teste din fișierul « *getTimeFromStamp* »

După rularea testelor unitare implementate s-a obținut următorul rezultat.

```
PASS src/helpers/_tests/getInitials.spec.jsx
PASS src/helpers/_tests/getFonts.spec.jsx
PASS src/helpers/_tests/getTimeFromStamp.spec.jsx
PASS src/helpers/_tests/getDateFormat.spec.jsx

Test Suites: 4 passed, 4 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        6.507s
Ran all test suites.
Done in 8.35s.
```

Figure 3.13 Teste din fișierul « *getTimeFromStamp* »

Din rezultatul afișat mai sus putem observa că precizia funcțiilor implementate de mai sus este corectă.

Testele au o structură analitică prin care se validează comportamentul fiecărei funcții dacă parametrii sunt valizi și pentru unele funcții ce nu tratează validitatea precedent se va verifica direct în corpul structural.

5 Concluzii

Tema abordată a avut scopul de-a demonstra în linii generale că orice aplicație de gestionare are o structură maleabilă în ceea ce privește designul propriu, iar interactivitatea și interfața cu utilizatorul pot fi oricând dezvoltate din punct de vedere operativ pentru a ajuta la crearea unui mediu cât mai familiar pentru utilizator.

Evoluția dinamică a modului de implementare a aplicațiilor web a creat și creează în continuare o creștere preponderentă a opțiunilor de design și a varietății de librării existente.

5.1 Rezultate obținute

Rezultatele obținute sunt satisfăcătoare din punct de vedere al aspectului și al funcționării aplicației.

Aplicația rezultată are o operativitate corespunzătoare din punct de vedere al acțiunilor desfășurate și a scopurilor realizate minimizând efortul și timpul necesar pentru utilizator să monitorizeze sau chiar să intervină în starea temelor actuale și să posede o statistică în timp real a evenimentelor actuale.

Predictibilitatea rezultatelor oferite de aplicație este mai ridicată deoarece timpul de răspuns a acțiunilor este unul redus, iar snackbox-urile ce oferă informații despre validitatea acțiunii au o viteză ridicată.

Animațiile incluse în apariția formularelor nu afectează decât într-un procentaj minor viteza de execuție a acțiunii având ca scop principal comportamentul dat și nu viteza de apariție.

Existența comunicării dintre un server și o interfață grafică determină aplicația să fie de tip client-server ceea ce poate să presupună și o dezvoltare ulterioară în care să se diversifice opțiunile de filtrare, sortare, adăugare și modificare a datelor actual implementate actualizând totodată și noua relație de comunicare și eventual o migrare pe alte tipuri de dispozitive.

Din punct de vedere algoritmic cerințele pre-stabilite au fost respectate. Algoritmii implementați se bazează pe un comportament integrat în tip object în proporție ridicată și manifestarea asincronă a fost integrată pentru funcția substituente etapei de inițializare din ciclul vieții componentelor.

Din punct de vedere al organizării s-a respectat o structură impusă care conservă componententele, constantele, stilurile, containerele, funcțiile externe și elementele de utilitate în foldere separate definite prin titluri expresive și regăsite într-un fișier index.

Așadar, din punct de vedere al performanței aplicația are structura și comportamentul sustenabile.

5.2 Direcții de dezvoltare ulterioare

Pot să confirm cu certitudine că aplicația rezultată poate fi dezvoltată într-un ansamblu mult mai amplu și mai complex cu adăugări diversificate cu scopul creării unei noi funcționalități :

- Adăugarea a mai multor câmpuri pentru temele propuse și respectiv eveniment.
- Sortarea evenimentelor după mai multe tipuri.
- Recuperare parolă prin notificare email.
- Atașament poza pentru utilizator.
- Câmp de selecție a firmelor pentru a ști ce firmă reprezintă prezentatorul evenimentului.
- Adăugare a unui comportament de filtrare preferențial care să țină cont de toate deciziile luate de utilizator.
- Customizare design prin punerea la dispoziție a unor teme generale a aplicației.
Ex : tema întunecată ce realizează transformarea caracteristicilor fontului și creează un mediu dorit de utilizator
- Câmp locații și respectiv orașe – aici este necesară existența unei baze de date foarte mari în care să se țină cont de toate locațiile care permit organizarea unui eveniment.
- Filtrare disponibilitate în funcție de ore.

În concluzie, consider că tema abordată este benefică și eficace pentru subcategoria din domeniul specificat și reușește să permită soluționarea problemelor specificate în primul capitol.

6 Referințe bibliografice

6.1 Documentație

- <https://www.w3schools.com/html/>
- <https://developer.mozilla.org/en-US/docs/Web/CSS>
- <https://www.codegrip.tech/productivity/everything-you-need-to-know-about-code-smells/>
- https://www.w3schools.com/css/css_intro.asp
- <https://www.todaysoftmag.ro/article/377/din-unelte-artizanului-software-unit-testing>
- <https://ro.wikipedia.org/wiki/Criptare>
- <https://ro.wikipedia.org/wiki/MySQL>
- <https://www.npmjs.com/package/json-server>
- <https://medium.com/@elinahovakimyan/code-refactoring-tips-for-react-projects-f7240156e403>
- <https://material-ui.com/>

6.2 Imagini

- <https://freefrontend.com/assets/img/html-css-404-page-templates/404-Animated-Page.jpg>
- https://images.assetsdelivery.com/compings_v2/dvarg/dvarg1406/dvarg140601049.jpg
- <https://img.tineye.com/result/df8c42bda3651e030c0afe6db9481b5d2e52f683858b056b0f8739fab199b8c8?size=160>
- <https://cdn.vectorstock.com/i/thumb-large/04/37/sweatshirt-icon-flat-style-vector-13580437.jpg>
- <https://www.welovesolo.com/champion-cup-and-medals-design-vector-set-01/>
- <https://sep.yimg.com/ay/yhst-83673118800219/jeep-pick-your-surprise-apparel-35.jpg>
- http://img.freeflagicons.com/thumb/round icon/romania/romania_640.png
- <https://i.pinimg.com/474x/42/e8/82/42e882f0bacd598811a7557b3868456c.jpg>
- <https://s3.amazonaws.com/img.iluria.com/product/5E2C25/E69F60/450xN.jpg>
- <https://avatanplus.com/files/resources/original/575179fc86d2615516448288.jpg>
- <https://cdn4.vectorstock.com/i/thumb-large/78/73/house-locate-place-logo-design-template-vector-23237873.jpg>
- <https://www.piliapp.com/skype-emoticons/>