VIZUALIZAREA INTERACTIVĂ A DATELOR MEDICALE CU PACHETUL MERN ȘI PLOTLY.JS

Stud. Andrei Eduard KUKUCSKA
Coordonator: Conf. univ. dr. Florentina Anica PINTEA

Universitatea „Tibiscus" din Timişoara, Facultatea de Calculatoare şi Informatică Aplicată
Centrul de Cercetare în Informatică Multidisciplinară, UTT

Autorul de corespondenţă: Andrei Eduard Kukucska, andreikukucska@gmail.com

DOI: https://doi.org/10.52744/CAPPS.2025.01

REZUMAT: *Există o nevoie acută de soluții tehnice pentru eficientizarea volumelor de date din ce în ce mai mari din domeniul cercetării medicale. Soluțiile existente prezintă o serie de limitări și dezavantaje, limitând procesul de analizare a acestora. Soluția prezentată propune crearea unei aplicații web, realizate cu pachetul de tehnologii MERN și biblioteca Plotly.js, care să permită crearea unor aplicații intuitive, optimizate și personalizate nevoilor exacte ale cercetătorilor din domeniul medical. Aplicația folosește React.js, React Router, React Material UI și Emotion.js pentru partea de frontend, Node.js, Express.js, JWT pentru partea de backend, MongoDB și Mongoose pentru baza de date și Plotly.js pentru crearea graficelor. Prin vizualizarea datelor cu ajutorul unor grafice intuitive și interactive, aplicația oferă o alternativă optimizată pentru analiza seturilor de date foarte mari, cum sunt cele din domeniul medical.*

CUVINTE CHEIE: *date medicale; vizualizare; MERN; Plotly.js, grafice; analiză; aplicație web;*

1. INTRODUCERE

1.1 Descrierea problemei

Asistăm la o creștere exponențială a volumului de date provenite din diferite domenii. Pe lângă valoarea economică pe care o reprezintă, ele au o importanță din ce în ce mai mare și din alte puncte de vedere, ca de exemplu cel al eficientizării cercetării medicale. Dar aceste volume ridicate de date aduc cu sine și nevoia unor soluții optimizate pentru analiza datelor. Instrumentele, care permit vizualizarea acestor date, se pot dovedi foarte utile în activitatea cercetătorilor [1].

Această lucrare propune o soluție personalizată pentru vizualizarea unor astfel de date, mai exact a unor date colectate de la senzori medicali utilizați în studii pe pacienți de diabet.

1.2 Abordări existente

Printre instrumentele existente, care sunt folosite, găsim atât tehnologii create pentru analiza datelor în general, precum D3.js, Tableau, Power Bi ori Google Charts, cât și instrumente create special pentru monitorizarea glicemiei, cum ar fi Tidepool sau Dexcom Clarity.

Instrumentele generice, din prima categorie, deși oferă diferite funcționalități pentru munca cu date, nu sunt create special pentru date medicale, prezentând o serie de dezavantaje, precum posibilități limitate de personalizare și adaptare pentru nevoile concrete ale studiilor medicale, nu sunt optimizate pentru a lucra cu seturi de date foarte mari [2], [3].

Instrumentele din cea de-a doua categorie, cu toate că sunt create special pentru monitorizarea nivelurilor glucozei, sunt destinate în principal medicilor și pacienților și nu cercetătorilor și prezintă limitări și mai serioase în ceea ce privește personalizarea lor sau adaptarea pentru lucrul cu alte date sau pentru vizualizarea unor corelații cu anumite evenimente [4].

1.3 Alternativa propusă

Propunerea prezentată constă în realizarea unei aplicații web, care poate fi creată și personalizată astfel încât să permită vizualizarea și analiza datelor într-un mod dinamic, intuitiv și interactiv. De asemenea aplicația este construită cu tehnologii des întâlnite în dezvoltarea aplicațiilor web, bazate pe JavaScript, permițând astfel realizarea acesteia de către dezvoltatori web fără să aibă nevoie de pregătire specializată suplimentară.

2. TEHNOLOGIILE FOLOSITE

Aplicația urmează modelul arhitectural Model-Vizualizare-Controlor sau *MVC* (*Model-View-Controller*) și este realizată cu pachetul de tehnologii MERN (MongoDB, Express.js, React.js, Node.js).

Pentru crearea diferitelor grafice este folosită biblioteca Plotly.js [5].

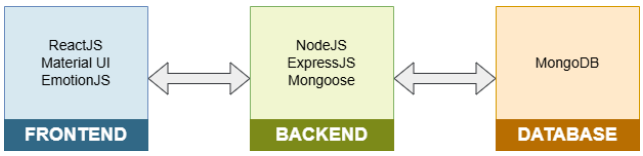

Fig. 1. Arhitectura aplicației
Sursă: prelucrare proprie

Toate bibliotecile și tehnologiile folosite în realizarea aplicației se bazează pe limbajul JavaScript. Iar baza de date folosită, MongoDB, este o bază de date de tip non-relațională, care stochează datele într-un format flexibil, asemănător formatului JSON (*JavaScript Object Notation*), ceea ce îi conferă un grad de compatibilitate ridicat cu tehnologiile bazate pe JavaScript [6].

2.1 Tehnologii frontend

React.js este o bibliotecă JavaScript de frontend, care permite crearea unor interfețe de utilizator moderne, dinamice și intuitive.
React are o arhitectură modulară, care permite crearea unor componente reutilizabile. De asemenea, React folosește un DOM Virtual, care îi permite să re-randeze doar acele elemente, care au fost modificate, optimizând astfel performanța.
Aplicația utilizează JSX (*JavaScript XML*), care este o extensie pentru JavaScript, care permite scrierea unui cod similar codului HTML în interiorul componentelor React.
De asemenea folosește *react hooks*, precum *useState*, care permite componentelor să aibe o stare locală sau *useEffect*, folosit pentru a gestiona anumite efecte secundare (de ex. în cazul apelurilor de tip REST API).
Pentru gestionarea stării globale, aplicația se bazează pe soluția integrată Context API, prin intermediul metodei *createContext()* și a hook-ului *useContext()*.
Pentru gestionarea erorilor este folosită componenta de tip clasă *React Error Boundary*, care practic prinde erorile apărute oriunde în componentele cuprinse în ea, înainte ca ele să ducă la prăbușirea întregii aplicații [7], [8].
React Router este cea mai populară bibliotecă inclusă ca dependență în aplicațiile construite cu React.js. Este folosită pentru navigare, în aplicațiile de tip *SPA* (*Single Page Application*) și la fel ca React, are un model, care se bazează pe componente.
În aplicația prezentată, React Router a permis crearea rutelor de navigare, atât a celor publice, cât și a celor private, prin folosirea componentelor precum *BrowserRouter*, *Routes* și *Route*.
De asemenea hook-ul *useNavigation* a permis utilizarea navigării programatice [9].
React Material UI este o bibliotecă de componente de design populară, care implementează stilul de design Material Design de la Google [10].
Biblioteca este cunoscută pentru crearea unor interfețe moderne și foarte intuitive, datorită combinației dintre stilul minimalist, elementelor inspirate din materiale (ex. hârtie sau cerneală) și prin acordarea unor semnificații practice animațiilor și interacțiunilor.
Printre elementele folosite din React Material Design, amintim diferitele componente grafice, proprietatea *sx*, care permite personalizarea stilului elementelor individuale sau componentele de layout.
Emotion.js este o bibliotecă de stil, de tip CSS-in-JS, care este și soluția de stilizare implicită a bibliotecii React Material UI. Abordarea CSS-in-JS înseamnă practic folosirea codului de tip CSS direct în componente, în fișierele de tip React/JavaScript, acest lucru permițând folosirea *props*, stării aplicației sau a unor instrucțiuni condiționale pentru definirea stilului [11].

2.2 Tehnologii backend

Node.js este un mediu de execuție, care permite rularea limbajului JavaScript în afara unui browser.
În cazul aplicației de față, Node.js a permis folosirea aceluiași limbaj (JavaScript) atât pe partea de frontend, cât și pe backend, reducând astfel dificultatea și timpul de dezvoltare [12].
Express.js este un framework de JavaScript, folosit pentru crearea de aplicații și API-uri (*Application Programming Interface*).
Express a permis crearea diferitelor tipuri de rute de tip RESTful (*Representational State Transfer*) ale aplicației și definirea metodelor acceptate de către acestea (ex. GET, POST, DELETE). Express a permis și crearea unor rute agregate, care combină date din mai multe surse [13].
De asemenea Express a fost aplicat și pentru crearea așa numitelor funcții de tip *middleware*, care gestionează memoria tip cache sau erorile apărute.
JWT (*JSON Web Tokens*) este un format modern pentru tokenuri, care permite securizarea datelor trimise între părți. Aplicația prezentată folosește JWT pentru autentificarea utilizatorilor [14].

2.3 Tehnologii folosite pentru baza de date

MongoDB este o bază de date de tip non-relațională, bazată pe documente, care a permis o integrare facilă cu celelalte tehnologii din aplicație [15].
Baza de date a aplicației constă în șase colecții:

- *users* - date de cont și de autentificare ale utilizatorilor;
- *anomalies* - date despre tipare neobișnuite;
- *events* - evenimente din jurnalul pacienților;
- *patches* - date de la senzorii medicali;
- *studies* - date despre studii;
- *subjects*.- date despre participanți.

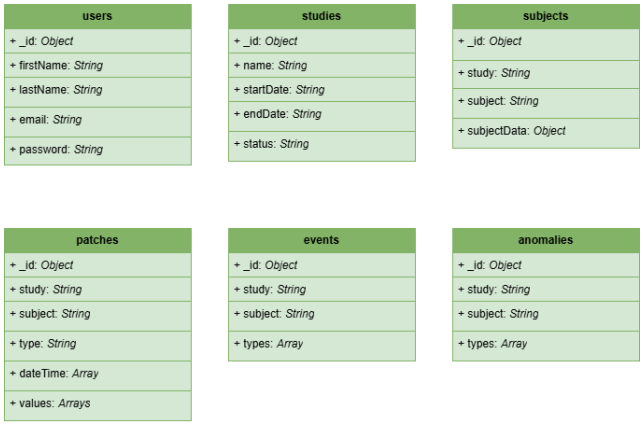

Fig. 2. Structura colecțiilor bazei de date
Sursă: prelucrare proprie

Mongoose este un instrument de tip ODM (*Object Data Modeling*), care a fost folosită pentru a implifica interacțiunile cu baza de date MongoDB [16].

2.4 Tehnologii folosite pentru vizualizare

Plotly.js este o bibliotecă JavaScript, folosită pentru vizualizarea datelor, prin grafice și diagrame. Cu ajutorul Plotly.js au fost create mai multe grafice cu bare și grafice de dispersie cu linii și marcatori [17].

3. STRUCTURA

Aplicația este formată din două tipuri de pagini: pagini publice (paginile de *Sign in* și *Sign up*) și pagini private (paginile *Studies*, *Patterns*, *Events* și *Events Details*). Pagina de *Sign in* conține formularul de autentificare iar cea de *Sign up* formularul de înregistrare pentru utilizatori noi.

Fig. 3. Pagina *Sign up*
Sursă: prelucrare proprie

Pagina *Studies* afișează metadate despre studiile analizate de către aplicație, cu ajutorul unui grafic cu bare și a unor componente de tip card. Pagina *Patterns* prezintă coeficientul de corelație a lui Pearson dintre nivelul glucozei și diferitele tipuri de evenimente, folosind în partea superioară un filtru, iar în cea inferioară afișând un grafic cu bare cu rezultatele.

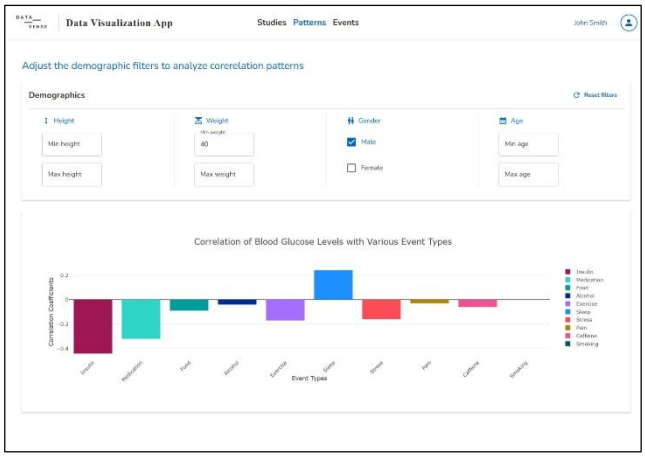

Fig. 4. Pagina *Patterns*
Sursă: prelucrare proprie

Pagina *Events* afișează o listă cu toate studiile incluse, prezentând anumite date statistice despre ele. Iar pe pagina *Events Details* se pot viziona grafice cu diferitele tipuri de evenimente, corelate cu valorile glucozei participanților la studii.

4. FUNCȚIONALITĂȚI

Pe paginile de *Sign in*, respectiv *Sign up*, utilizatorii se pot autentifica, respectiv înregistra, folosindu-se de formularele existente. În cazul în care datele introduse nu sunt corecte, vor fi afișate mesaje de avertizare.
Din paginile private, utilizatorul poate accesa anumite ferestre de dialog, prin care poate gestiona sesiunea sau contul de utilizator.
Pagina *Studies* conține un grafic cu bare, care prezintă în mod dinamic, numărul participanților din fiecare studiu în parte.
În partea de jos a paginii *Patterns* se găsește un grafic care vizualizează coeficientul de corelație a lui Pearson, calculat pentru fiecare tip de eveniment în parte, în raport cu nivelul glucozei de sânge.
Pagina *Events Details* conține grafice, de tip bară combinate cu un grafic de tip dispersie cu linii și markere, care vizualizează modificările glucozei serice survenite în timpul unor evenimente.

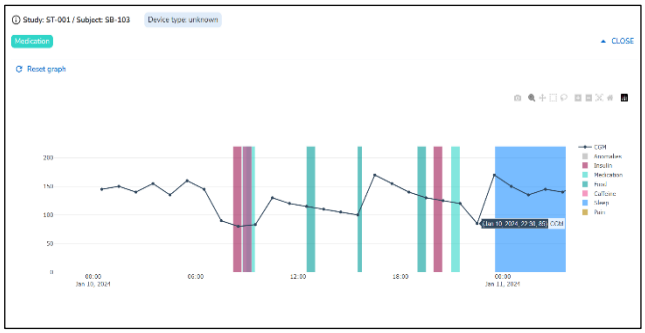

Fig. 5. Grafic cu glucoza serică și diferite evenimente
Sursă: prelucrare proprie

Pe pagina *Patterns* se găsește un filtru, care permite selectarea participanților la studii în funcție de anumite date demografice; opțiunile din acest filtru decid datele căror participanți vor fi afișate.

5. DESIGN

Designul aplicației respectă recomandările stilului Material Design dezvoltat de Google. Acesta conferă aplicației un stil minimalist, intuitiv și centrat pe utilizator [18].

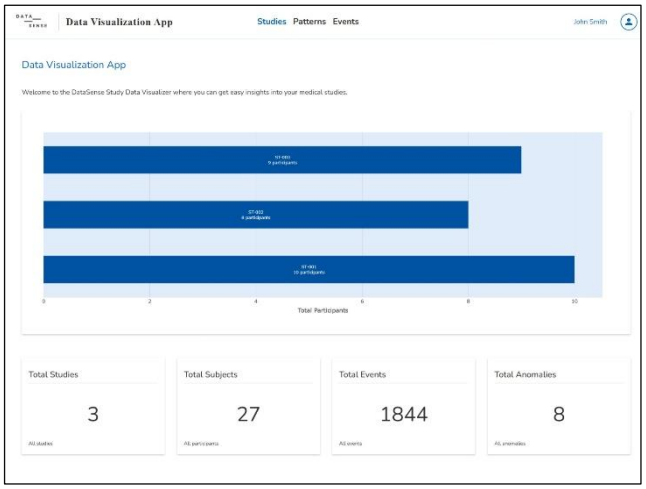

Fig. 6. Pagina *Studies*
Sursă: prelucrare proprie

Interfața de utilizator a aplicației este realizată cu ajutorul bibliotecii React Material UI, care este construită pe baza principiilor Material Design, acest lucru permițând folosirea consistentă a acestui stil pe tot cuprinsul aplicației.

CONCLUZII

Aplicația web realizată reprezintă o soluție viabilă pentru optimizarea analizei și vizualizării seturilor de date provenite din studii medicale realizate pe pacienți cu diabet și poate fi folosită cu succes de către cercetătorii și specialiștii implicați în astfel de studii.
Combinația de tehnologii alese, în primul rând tehnologiile din pachetul MERN și biblioteca Plotly.js, a facilitat dezvoltarea unei aplicații ușor de utilizat și personalizat, cu o performanță superioară.
Funcționalitățile și integrările prezentate reprezintă doar câteva exemple din potențialul vast posibil. Ca și idei pentru dezvoltare putem aminti integrarea unei funcționalități care permite memorarea și chiar transmiterea anumitor selecții sau exportarea unor fișiere cu date, care pot fi folosite în alte aplicații sau instrumente (de ex. MATLAB). De asemenea inteligența artificială poate oferi o serie de soluții pentru eficientizarea sau chiar automatizarea unor procese, de exemplu depistarea automată a anomaliilor, sugerarea unor tipare găsite de către modelele de învățare automată sau implementarea unui robot de tip chat, care să poată răspunde la interogări legate de studiile analizate.

BIBLIOGRAFIE

[1]. *** Real-Time Data Visualization: Turning Raw Data into Insights, https://hopara.io/blog/real-time-data-visualization
[2]. *** Tableau Manual, https://tableaumanual.com/
[3]. *** D3js, https://d3js.org/
[4]. *** Tidepool API Documentation, Tidepool, https://github.com/tidepool-org/TidepoolApi
[5]. *** Understanding MVC Architecture in the MERN Stack, https://medium.com/@ansari028amaan/understanding-mvc-architecturein-the-mern-stack-5cc083828298
[6]. *** Why Single Page Application (SPA) architecture is so popular?, https://medium.com/nerd-for-tech/why-single-pageapplication-spa-architecture-is-so-popular-141b85400204
[7]. *** React Documentation, https://react.dev/learn
[8]. *** Getting started with React, Getting started with React - Learn web development | MDN
[9]. *** React Router Official Documentation, https://reactrouter.com/home
[10].*** Material UI, https://mui.com/material-ui/gettingstarted/
[11].*** Emotion Introduction, https://emotion.sh/docs/introduction
[12].*** About Node.js, https://nodejs.org/en/about
[13].*** Express - Node.js web application framework, https://expressjs.com/
[14].*** JSON Web Tokens, https://jwt.io/
[15].*** MongoDB: The Developer Data Platform, https://www.mongodb.com/
[16].*** Mongoose v8.12.1: Getting Started, https://mongoosejs.com/docs/
[17].*** Plotly JavaScript Open Source Graphing Library, https://plotly.com/javascript/
[18].*** Material Design, https://m3.material.io/