**UNIVERSITATEA DIN CRAIOVA**

**FACULTATEA DE ȘTIINȚE**

**SPECIALIZAREA INFORMATICĂ**

**LUCRARE DE LICENȚĂ**

Îndrumător științific: Absolvent:

**Lect. univ. dr. Stancu Mihai** **Bondokić Nikola**

**CRAIOVA 2023**

**UNIVERSITATEA DIN CRAIOVA**

**FACULTATEA DE ȘTIINȚE**

**SPECIALIZAREA INFORMATICĂ**

Rapoarte grafice pentru aplicații Enterprise

Îndrumător științific: Absolvent:

**Lect. univ. dr. Stancu Mihai** **Bondokić Nikola**

**CRAIOVA 2023**

Cuprins

**Listă de figuri**

**Fig. 2-1 Prima diagrama de linii creată de Wiliam Playfair**........................................

**Fig. 2-2 Unul dintre modele de terminale grafice**.......................................................

**Fig. 2-3 Un exemplu de rapoarte grafice în mass-media**...........................................

**Fig. 2-4 Chart.js**.............................................................................................................

**Fig. 2-5 CanvasJS**.........................................................................................................

**Fig. 2-6 Google Charts**..................................................................................................

**Fig. 2-7 Rapoarte grafice de tip linie**...........................................................................

**Fig. 2-8 Rapoarte grafice de tip bară**...........................................................................

**Fig. 2-9 Rapoarte grafice de tip plăcintă**.....................................................................

**Fig. 2-10 Rapoarte grafice de tip polar**........................................................................

**Fig. 3-1 REST Api Client-Server**..................................................................................

**Fig. 3-2 REST API – Statelessness**..............................................................................

**Fig. 4-1 Operație CRUD**.................................................................................................

**Fig. 4-2 Diferența între CRUD și REST**........................................................................

1. **Introducere**

Proiectul își propune crearea și implementarea unei aplicații web ce permite

utilizatorului sa vede niște grafice și sa citește text despre compania “Bondi”.

Tot in acest proiect utilizatorul poate sa vede și tabela cu Angajatii și pagina cu produse care sunt imparțite pe pagini.

Acest capitol își propune sa realizeze o scurtă introducere în tematicele abordate de către această lucrare și să ofere contextual și motivația din spatele realizării acesteia.

* 1. **Contextul proiectului**

În lumea online, informațiile de pe internet sunt uneori complexe și greu de înțeles. Aceasta sunt informați care sunt prezentate folosind niște tabele în care trebuie să cauți ceea ce ai nevoie. Dar de aceea există informații care sunt afișate și prezentate într-un mod ordonat și interactiv, fără a fi nevoie să vă faceți griji cu privire la căautatea în tabele și analizarea datelor din aceleași tabele.

Graficele prezentate pe internet sunt prezentate cu ajutorul unor librări, iar în această lucrare, de exemplu, se folosește librărie ChartJS, care este folostiă pentru prezentarea de grafice de diverse forme. ChartJS este o librărie de tip open source, scrisă în limbajul de programare JavaScript, care este folosită pentru a reprezenta obiecte vizuale, care momentan conține 8 tipuri de grafice: bară, linie, radar, arie, polar, scatter (împrăștiat), bule (Bubble) și plăcintă (Pi). Librărie ChartJS printre multe alte librări, în momentul de față este cea mai populară librărie de rapoarte grafice de pe GitHub. În a doua parte a acestei lucrări se va prezenta o pagine cu mai multe pagini separate în care sunt prezentate mai multe tipuri de grafice, pentru a arăa ușorința prin care se pot obține rapoarte graice.

Utilizatorul va avea posibilitatea de a vedea toate acele grafice împărțite în mai multe pagini separate, iar administratorul site-ului va avea posibilitatea de a schimba aceleași grafice.

* 1. **Motivația proiectului**

Motivația pentru proiect provine din dorința de a permite accesul multor clienților la o aplicație web, capabilă să le ajuta să vadă toate pprodusele dintr-un magazin sau pentru a înțelege mai ușor unele date care sunt afișate cu unor grafice, cu o interfață prietenoasă, cu un meniu ușor de gestionat, și cu posibilitatea de a selecta limba.

Motivația a venit și din faptul că există multe astfel de aplicații pe internet,

deși aceste aplicații sunt foarte populare și destul de cautate.

* 1. **Obiectivele principale ale lucrării**

Principalele obiective al creării aplicației este de a crea un web site ușor

de utilizat și cu o interfață ușor de înțeles, care ar ajuta mulți utilizatori. Acest web site oferă fiecărui utilizator o reprezentare vizuală a graficelor de diverse tipuri, dar și o reprezentare vzuală a mărfurilor pe care le deține magazinul, precum și diverse alte informații despre această companie.

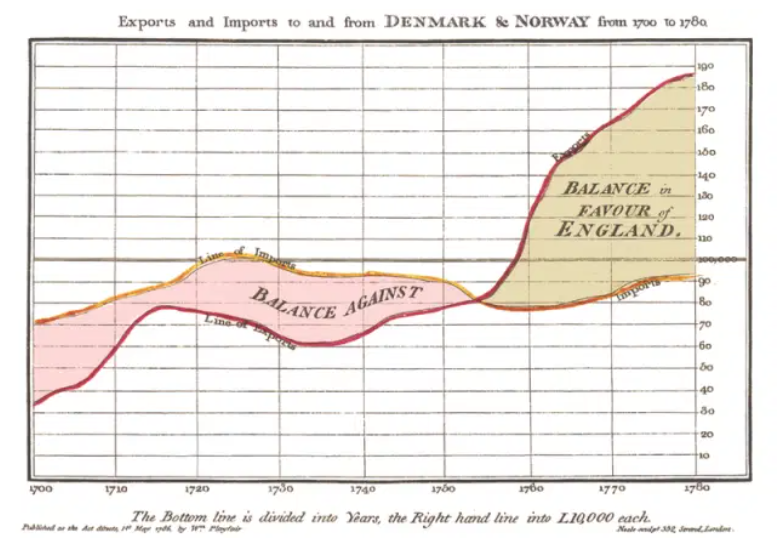
Obiectivele care vor fi atinse și îndeplinate în cadrul acestui proiect sunt:

* Învățarea limbajului de programare C#
* Învățarea limbajului de programare JavaScript
* Învățarea și înțelegerea anumitor librari care sunt folosite pentru a crea grafice cu ajutorul limbajului de programare JavaScript
  1. **Obiective secundare**

Obiectivele acestui proiect nu sunt de a câștiga bani, ci de a învăța noi tehnologii, noi limbaje de programare și de a învăța din greșeli.

1. **Rapoarte grafice**
   1. **Istoria și evoluția rapoartelor grafice**

În ceea ce este acum Parcul Național Serra De Capivara din Brazilia pereții

peșterilor poartă unele dintre cele mai vechi reprezentări grafice ale informațiilor iar acele grafice au o vechime de 25.000 de ani. În perioada Evului Mediu, utilizarea diagramelor și graficelor a câștigat popularitate ca mijloace de comunicare preferate în creștere, cu toate acestea, find limitată la anumite subiecte. După apariția presei tipografice în secolul al 15 - lea în anii 1400, oamenii au constatat că devine tot mai convenabil să se angajeze în scriere, desen și diseminare a informațiilor pe o scară largă. Edmund Hally este creditat pentru crearea diagramei bivariate inagurale, denumită și diagramă de dispersie sau grafic x-y, utilizănd date de observație referitoare la presiunea barometrică și altitudinea. La sfârșitul secului al 18 – lea, William Playfair a introdus conceptul de diagramă statică modernă. Pe parcursul vieții lui Wiliam Playfair el a conceput cele mai vechi exemple documentate ale diagramei cu bare, diagramei cu linii și diagramei circulare.

**Fig. 2-1 Prima diagrama de linii creată de Wiliam Playfair**

*Sursa: Don Kopf. When Did Charts Become Popular?, 2016*

De la începutul secolului 20, reprezentările grafice ale datelor au devenit comune în manuale și erau utilizate de guverne, finanțatori și oameni de știință.

 Este posibil ca data de 4 iulie 1933 să fie considerată momentul în care vizualizarea datelor a devenit o componentă importantă in media principală (mass media). În anii 1950 și 1960, inginerii au creat limbaje de programare pentru prelucrarea datelor statice (Fortran și S), precum și programe care au simplificat vizualizarea datelor (SAS și Lisp-Stat). Înțelegerea acestor instrumente inovatoare pentru analiza și vizualizarea datelor era limitată la un grup restrâns de programatori. În anii 1980, programe software precum Lotus 1-2-3 și Microsoft Excel au fost create și au permis mai multor oameni să creeze diagrame, și pentru jurnalisti muncă a devenit mai ușoară pentur că nu au avut nevoie sa deseneze diagramele de mana, și puteau face diagrame de pe calculatoare personale.

**Fig. 2-2 Unul dintre modele de terminale grafice**

*Sursa: Don Kopf. When Did Charts Become Popular?, 2016*

* 1. **Ce este un raport grafic ?**

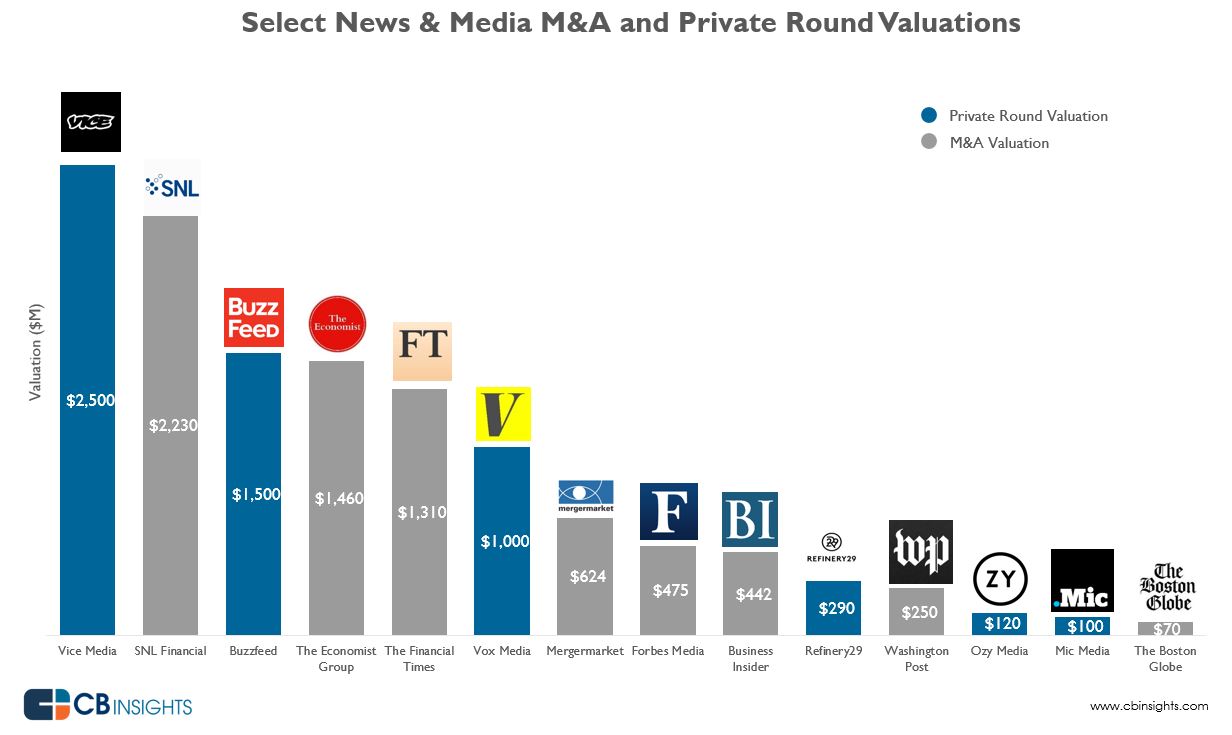
Digrama este o reprezentare grafică pentru vizualizarea unor date. O

diagrama poate reprezenta date numerice, funcții sau diferite alte informații care sunt prea complicate și lungi să fie transpuse într-o propoziție. Rapoarte grafice, cunoscute uneori și ca rapoarte vizuale, integrează elemente grafice pentru a îmbunătăți aspectul vizual al datelor prezentate și pentru a îmbunătăți ușorința în utilizare prin reprezentarea grafică a datelor sub fomră de grafice sau diagrame. Graficele pot ilustra rapid comortamentul general și pot evidenția fenomene, anomalii și relații între punctele de date care altfel ar putea fi trecute cu vederea, contribuind astfel la predicții și luarea de decizii mai bune, bazate pe date.

* 1. **Rolul rapoartelor grafice**

Rolul fundamental al rapoartelor grafice este acela de a compensa cantități

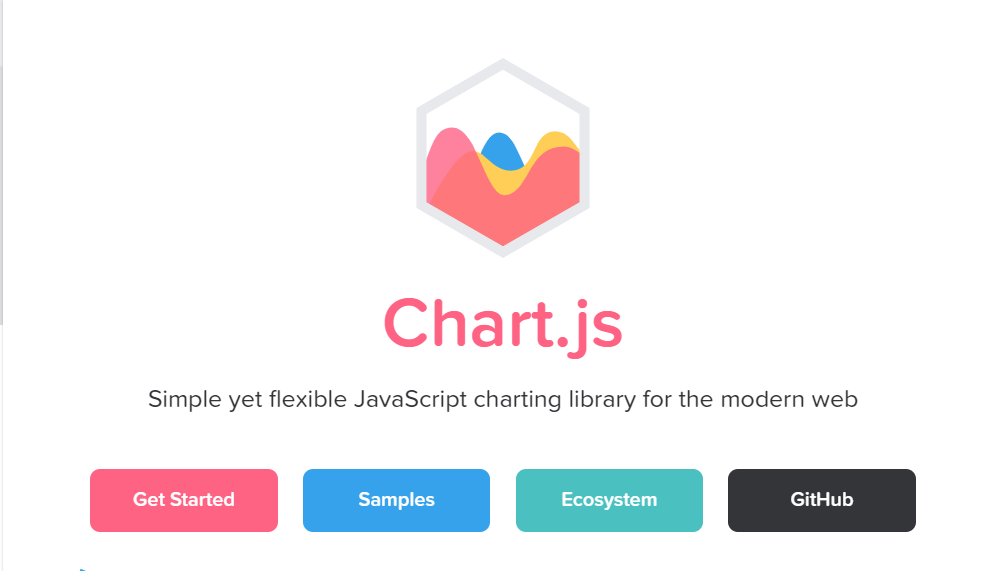
mari de informație in formate ușor de înțeles. Rapoarte grafice sunt folosite zilnic în diverse domenii, precum:

* Învățământ: Diagramele poate ajuta elevii sau studenții să înțeleagă unele date și probleme mai ușor și mai bine.
* Medicină: Diagramele poate fi folosite în medicină pentru o înțelegere mai ușoară a datelor de sănătete și pentru prezentarea lor ușoară.
* Știri: Rapoarte graficee pot fi folositee în jurnalism pentru a reprezenta diverse diagrame privind banii sau numărul de persoane sau creșterea prețurilor față de anii precedenți.

**Fig. 2-3 Un exemplu de rapoarte grafice în mass-media**

*Sursa:* [*https://www.cbinsights.com/research/news-media-valuations/*](https://www.cbinsights.com/research/news-media-valuations/)

* 1. **Librării pentru rapoarte gracie** 
     1. **ChartJs**

Chart.js este înzestat cu tipuri TypeScript încroporate și oferă

compatibilitate extinsă cu toate framework-urile JavaScript consacrate. În loc să utilizeze SVG, Chart.js preintă elementele diagramei peo pânză HTML5, spre deosebire de alte biblioteci de diagrame care se bazează, în principal pe D3.js.

**Fig. 2-4 Chart.js**

*Sursa:* [*https://www.chartjs.org/*](https://www.chartjs.org/)

În proiectul meu folosesc această bibliotecă pentru a crea grafuri, doarece a fost ușor de utilizat și totul a fost explicat frumos.

* + 1. **CanvasJS**

CanvasJS oferă suport extins pentru peste 30 de diagrame JavaScript,

inclusiv linie, coloană, bară, zonă, etc. Aceste diagrame sunt redări fără probleme pe diverse dispozitive, inclusiv dispozitive de la Apple precum sunt iPad, iPhone, iMac, Android și PC-uri. Cu ajutorul CanvasJS, puteți crea panouri de control bogate cu ușoriță.

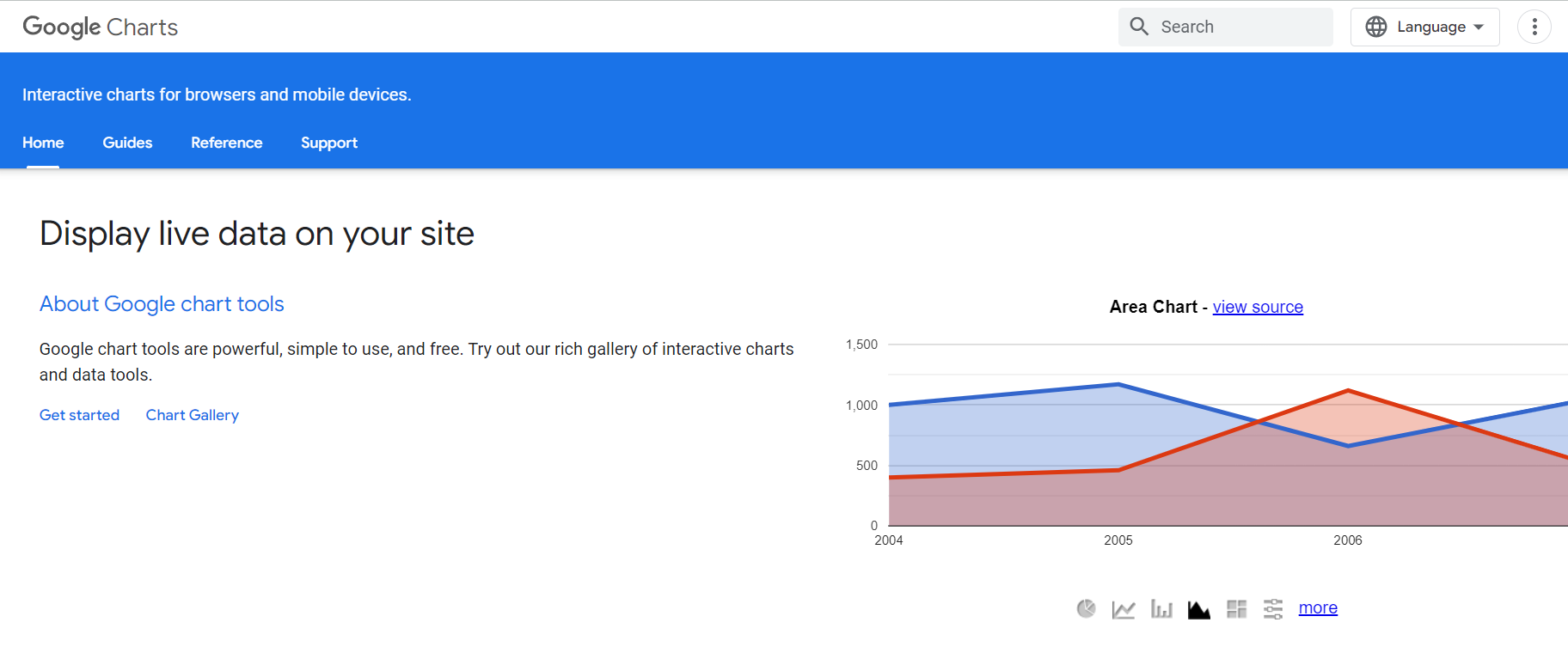
**Fig. 2-5 CanvasJS**

*Sursa:* [*https://canvasjs.com/*](https://canvasjs.com/)

* + 1. **Google Charts**

Google Charts este o suită de diagrame care există de mult. Vine cu o

bogată documentație este încărcată cu exemple pentru a vă ajuta să începeți.



**Fig. 2-6 Google Charts**

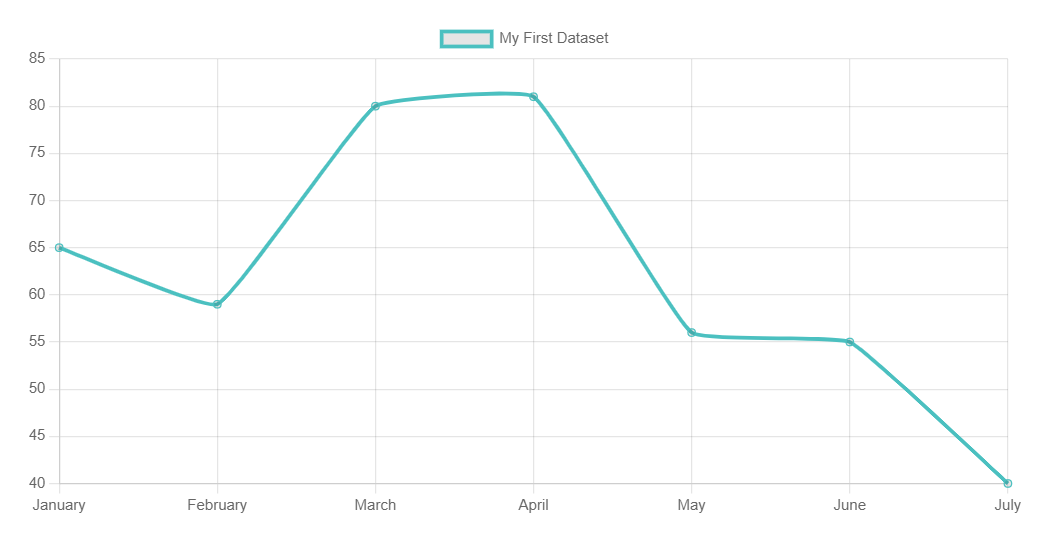
*Sursa:* [*https://developers.google.com/chart*](https://developers.google.com/chart)

* 1. **Tipuri de rapoarte grafice**

Existe mai multe tipuri de grafice printre care sunt:

* Grafice de tip linie (Line)
* Grafice de tip bară (Bar)
* Grafice de tip plăcintă (Pi)
* Grafice de tip polar (Polar Area)
* Grafice de tip arie (Area)
* Grafice de tip bule (Bubble)

Fiecare dintre aceste grafice are propriul mod specific de afișare a datelor.

* + 1. **Grafice de tip linie**

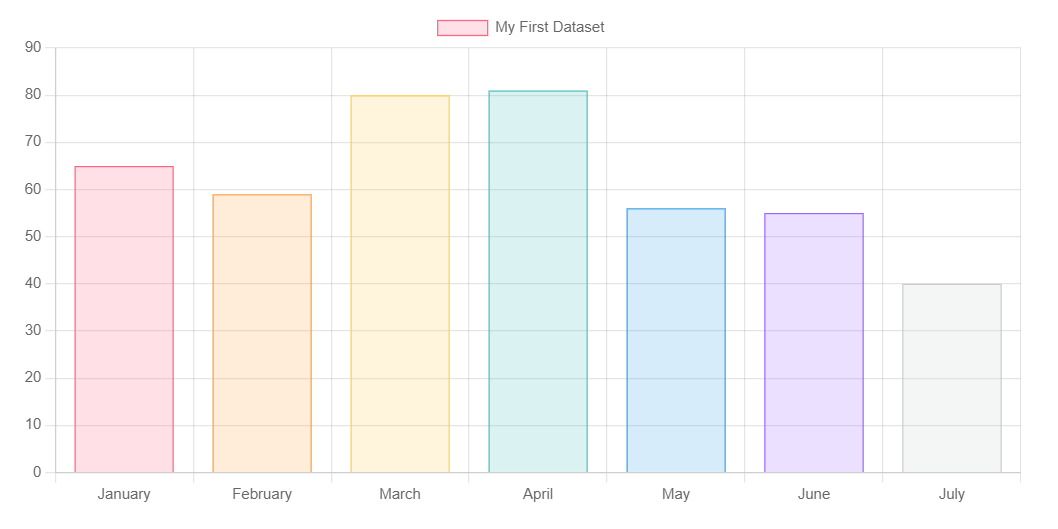
**Fig. 2-7 Rapoarte grafice de tip linie**

*Sursa:* [*https://www.chartjs.org/docs/latest/charts/line.html*](https://www.chartjs.org/docs/latest/charts/line.html)

Un grafic de tip linie, reprezintă o modalitate de a reprezenta punctele de date pe o linie. Este folosit frecvent pentru a ilustra tednințile în date sau pentru a compara două seturi de date distincte.

* + 1. **Grafice de tip bară**

Un grafic de tip bară, oferă o modalitate de a prezenta valorile datelor

reprezentate sub forma de bare verticale. Uneori este utilizat pentru a ilustra tendințele daatelor și pentru a compara mai multe seturi de date.

**Fig. 2-8 Rapoarte grafice de tip bară**

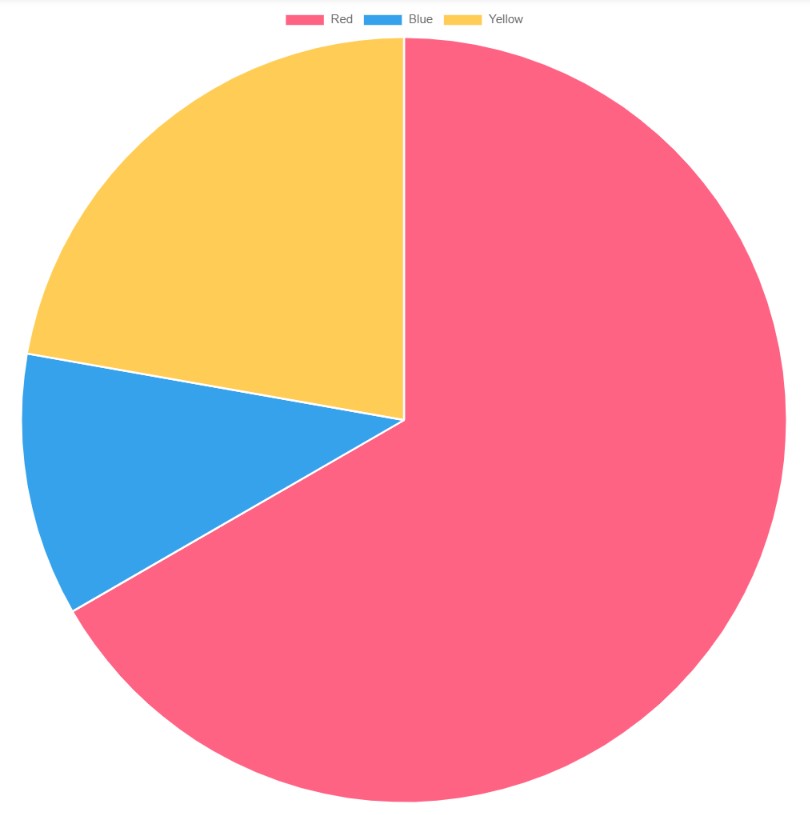
*Sursa:* [*https://www.chartjs.org/docs/latest/charts/bar.html*](https://www.chartjs.org/docs/latest/charts/bar.html)

* + 1. **Grafice de tip plăcintă**

Diagrame de tip Pi (placintă) și Doughnut, sunt recunoscute pe scară

largă ca fiind două dintre cele mai utilizate tipuri de grafice. Puterea lor constă în capacitatea de a afișa eficient proporțiile relative între diferitele puncte de date. Un grafic de tip plăcintă compară părți ale unui întreg. Acesta afișează o distribuire procentuală, astfle încât, întreaga plăcintă reprezintă datele totale și fiecare segment reprezintă o categorie anume din întreg. O diagramă circulară cu o zonă circulară goală în mijloc se numește diagrama de tip Doughnut (gogoș).

Graficele de tip Pi sunt utilizate pe scară largă pentru a compara categotii atunci când avem seturi de date mici. Totuși, acestea grafice nu sunt potrivite pentru a afișa mai multe serii de date și ar trebui evitate atunci când avem mai multe de câteva categorii.



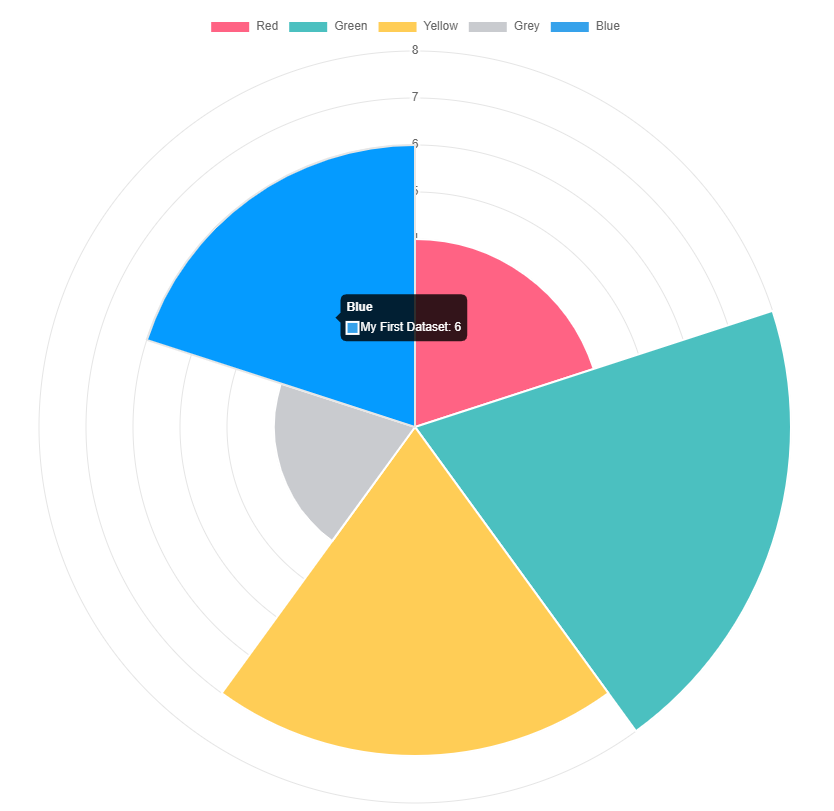
**Fig. 2-9 Rapoarte grafice de tip plăcintă**

*Sursa:* [*https://www.chartjs.org/docs/latest/charts/doughnut.html#doughnut*](https://www.chartjs.org/docs/latest/charts/doughnut.html%23doughnut)

* + 1. **Grafice de tip polar**

Graficele de tip polar, prezintă asemănări cu graficele de tip tort (Pi), cu

excepția faptuli că fiecare segment are un unghi cinstant. Acest tip de grafice se dovedește foarte avantajos atunci când dorim să prezentăm date comparative similare cu un grafic de tip tort (Pi).



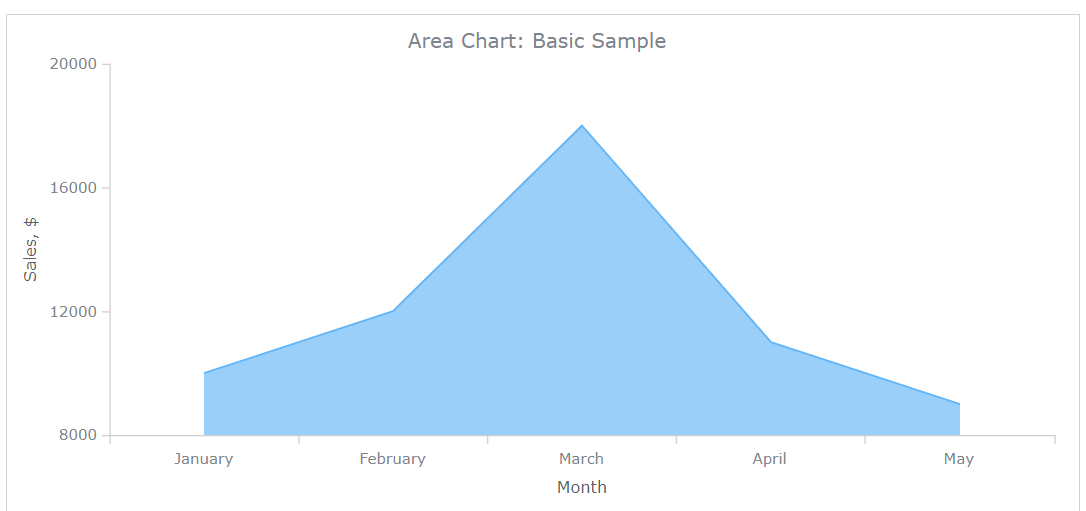
**Fig. 2-10 Rapoarte grafice de tip polar**

Sursa: <https://www.chartjs.org/docs/latest/charts/polar.html>

* + 1. **Grafice de tip arie**

Un grafic de tip zonă / arie se bazează pe graficul de linie, prezintă

informațiile ca o serie de puncte de date conectate prin segmente de lini drepte. Graficul de tip zonă evidențiază magnitudinea schimbărilor în timp și poate fi utilizat pentru a evidenția valoarea totală în cadrul unei tendenție.



**Fig. 2-11 Rapoarte grafice de tip arie / zonă**

*Sursă:* [*https://docs.anychart.com/Basic\_Charts/Area\_Chart*](https://docs.anychart.com/Basic_Charts/Area_Chart)

* + 1. **Grafice de tip bule**

**Fig. 2-12 Rapoarte grafice de tip bule**

*Sursa*[*: https://www.chartjs.org/docs/latest/charts/bubble.html*](:%20https:/www.chartjs.org/docs/latest/charts/bubble.html)

Un grafic de tip bule este utilizat pentru a afișa trei dimensiuni ale datelor

în aceleași timp. Poziția bulii este determinată de primele două dimensiuni și axele orizontale și verticale corespunzătoare. A treia dimensiune este reprezentată de mărimea bulilor.

1. **REST**
   1. **Despre REST**

Conceptul de REST a fost definit de către Roy Fielding, un informatician

American care s-a născut în anul 1965. El a prezentat principiile REST în teza sa de doctorat în anul 2000. El este unul dintre autorii principali ai specificație HTTP și creator stilului arhitectural Transferul Stării Reprezentative (REST). El este o autoritae în arhitectura rețelelor de calculatoare și a confondat proiectul Apache HTTP Server.

* 1. **Ce este REST**

REST, prescurtarea de la Transferul Stării Reprezentive, se referă la un stil

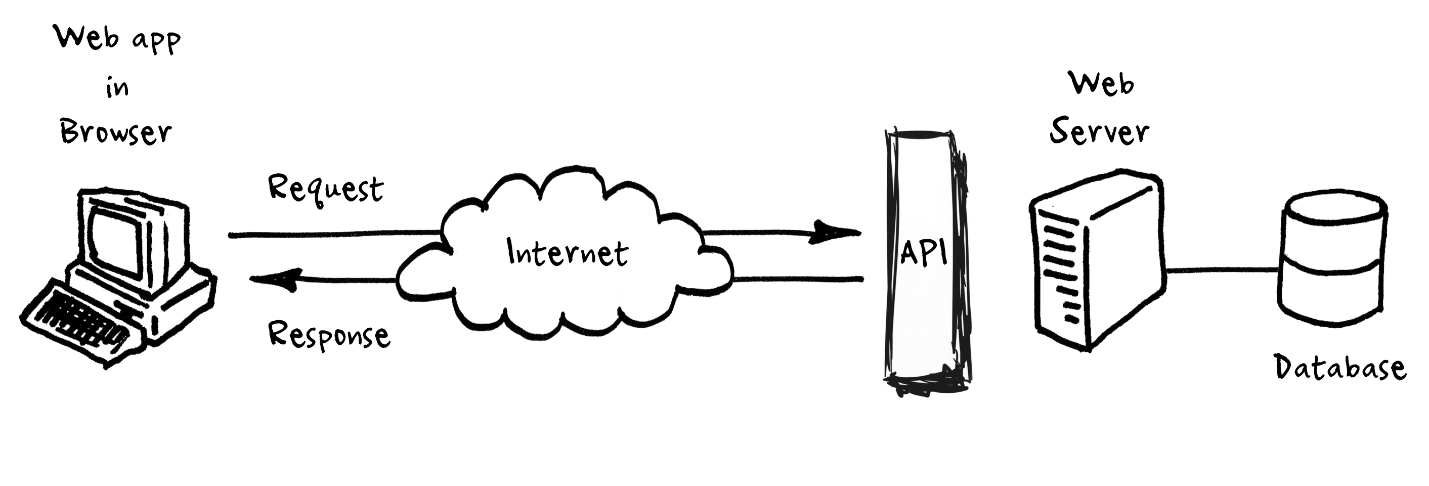
arhitectural care stabilește standardizarea între sisteme de calcul bazate pe web. Obiectivul său principal este de a facilita comunicarea fluentă între diferite sisteme. Sistemele compatibile cu REST, adesea numite sisteme RESTful, se disting prin natura lor lipsită de stare și prin separaea clară a responsabilităților între clienți și servere. Un API este o interfață de programe a aplicațiilor. Este un set de reguli care permit programelor să comunice între ele. Dezvoltatorul creează API-ul pe server și permite clientului să comunice cu acesta. REST determină modul în care arată API-ul. REST este un set de reguli pe care dezvoltatorii le urmează atunci când creează propriul API. Una dintre aceste reguli stipulează că ar trebui să puteți obține o bucată de date atunci când accesați o anumită adresă URL. Fiecare adresă URL este considerată o cerere, în timp ce datele trimise înapoi către dumneavoastră sunt numite răspunsuri. Avem posibilitate de a comunica cu serverele folosind protocolul HTTP. Prin intermediul acestor protocoale, putem efectua operațiuni de Creare, Citire, Actualizare și Stergere a datelor, cunoscute sub denumirea de operațiuni CRUD.

* 1. **Client – Server**

În stilul arhitectual REST, implementarea clientului și implementarea

severului pot fi realizate independent, fără ca unul să cunoască despre celălalt. Acest lucru înseamnă că codul de pe partea clientului poate fi modificara în orice moment fără a afecta funcționarea severului, iar codul de pe partea severului poate fi modificat fără a afecta funcționarea clientului. Atât timp cât fiecare parte știe ce format de mesaje să trimită, ele pot fi menținute modular și separate. Separarea preocupărilor referitoare la interfață utilizatorului de cele referitoare la stocarea datelor îmbunătățește flexibilitatea interfeței pe diverse platforme și îmbunătățește scalabilitatea prin simplificarea componentelor serverului. În plus, separarea permite fiecărei componente să evolueze independent.

Prin utilizarea unei interfețe REST, diferitele clienți accesează aceleași punct final REST, efectuează aceleași acțiuni și primes aceleași răspunsuri.

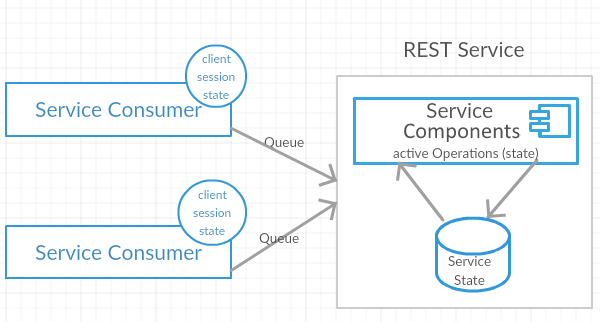


**Fig. 3-1 REST Api Client-Server**

*Sursa:* [*https://robert-drummond.com/2013/05/08/how-to-build-a-restful-web-api-on-a-raspberry-pi-in-javascript/*](https://robert-drummond.com/2013/05/08/how-to-build-a-restful-web-api-on-a-raspberry-pi-in-javascript/) *(Robert Drummond – “Javascript in the new C”)*

* 1. **Stateless**

Roy Fielding s-a inspirat din protocolul HTTP, ceea ce se referă într-o

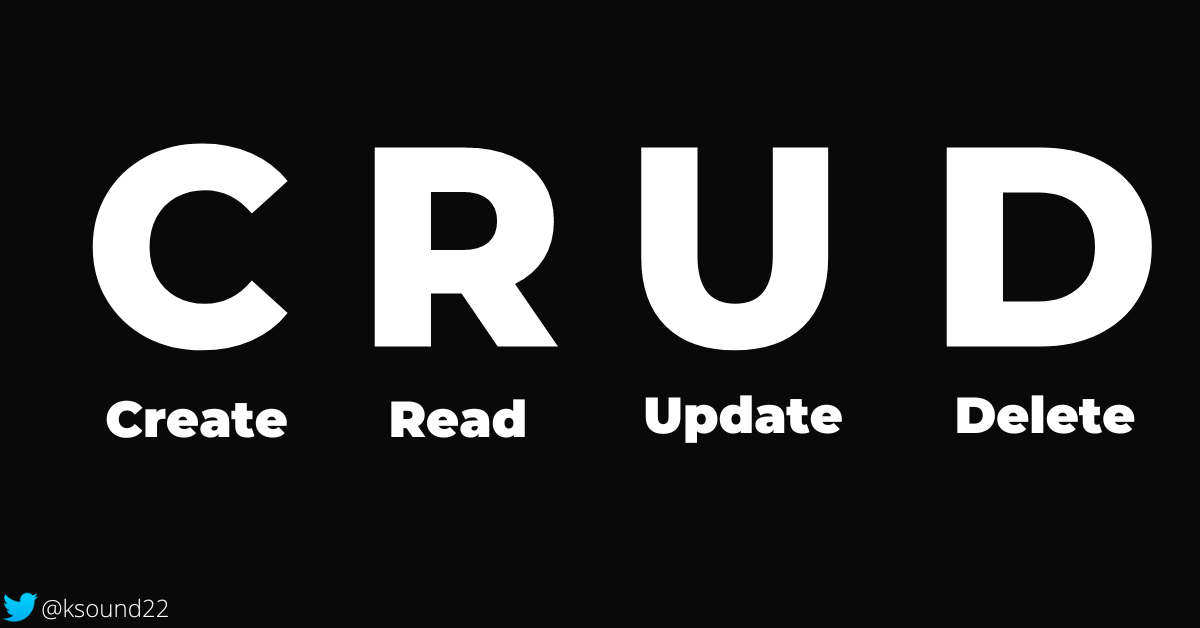
restricșie specifică: menținerea fără stare (statelessness) în toate interacțiunile client-server. Acest lucru înseamnă că serverul evită stocarea oricăror informații referitoare la cererile HTTP anterioare ale clientului. Fiecarea cerere este tratată ca fiind nouă și independentă, fără cunoștință despre interacțiunile anerioare. Ca urmare, nu există sesiuni sau date istorice păstrate. Cu toate acestea, în cazul în care aplicația client necesită un mediu cu stare pentru utilizatorii finali, unde utilizatorii se autentifică o singură dată și ulterior efectuează operații autorizate, fiecare cerere din partea clientului trebui să conține toate informațiile necesare pentru a răspunde solicitării. Acesta include detalii de autentificare și autorizare, asigurând-se astfel că serverul poate servi corespunzător cererile clientului.

**Fig. 3-2 REST API – Statelessness**

*Sursa*[*: https://www.oreilly.com/library/view/hands-on-restful-api/9781788992664/f618d9e6-1ca9-4177-a464-69ef1eec133a.xhtml*](:%20https:/www.oreilly.com/library/view/hands-on-restful-api/9781788992664/f618d9e6-1ca9-4177-a464-69ef1eec133a.xhtml)

1. **Operații CRUD**
   1. **Despre CRUD**

O aplicație softwer ar trbui să posede capacitatea de a efectua patru

operațiuni fundamentale, cunoscute sub acronimul CRUD: Creare (Create), Citire (Read), Actualizare (Update) și Ștergere (Delete). În astfel de aplicații, utilizatorii trebuie să poată genera date, să aibă acces și să vizualizeze datele în cadrul interfeței utilizator, să modifice sau să editeze datele și să le șteargă atunci când este necesar. O aplicație CRUD completă este alcătuiră în mod obișnuit din trei componente esențiale: un API sau server, o bază de date și o interfață utilizator (UI). API-ul cuprinde codul și metodele resonsabile de gestionarea cererilor, în timp ce baza de date stochează și facilitează recuperarea informațiilor solicitate de utilizatori. Interfață utilizator permite utilizatorului să interacționeze cu aplicația și să efectueze operațiunile dorite. Dezvoltarea unei aplicații CRUD este posibilă cu diverse limbaje de programare. Aplicația nu trebuie neapărat să fie full-stack, și poate să fie doar aplicație JavaScript. Fiecare literă din acronoml CRUD corespunde unei metode specifice de creare HTTP.

**Fig. 4-1 Operație CRUD**

*Sursa:* [*https://www.freecodecamp.org/news/crud-operations-explained/*](https://www.freecodecamp.org/news/crud-operations-explained/)

* 1. **Cum funcționează CRUD**

În perioada erei WEB 2.0, operațiile CRUD au constituit fundamentul

majorității site-urilor dinamice. Cu toate aceastea, trebuie să facem destincția între CRUD și verbele de acțiune HTTP. De exemplu, dacă doriți să creați o înregistrare, trebuie să utilizați metoda POST, daca doriți să actualizați o înregistrare trebuie să folosiți metoda PUT / PATCH. Dacă dotiți să ștergeți o înregistrare, veți folosi metoda DELETE.

* 1. **Beneficile CRUD**
     1. **Șanse reduse de atacuri Injection SQL**

Utilizarea limbajului SQL prezintă șanse mai mari de a întâmpina atacuri

SQL, doarce instrucțiunile SQL sunt executate pe serverele. Serverul stochează, informațiile și procedurile SQL, ceea ce poate fi fatal în care o resursă neautorizată obține accesul la acestea. Utilizarea operațiilor CRUD menține sub control posibilitatea de atac prin injectare SQL, deoarece utilizează acțiuni deja stocate și nu necesită generarea de interogări dinamice utilizând datele fuurnizate de utilizator. De asemenea, se asigură de o citire precisă a parametrilor instucțiunilor SQL.

* + 1. **Mai bună protecție îmotriva navigarii casuale**

Utilizatorii de declarații SQL ad hoc obțin permisiunea de a accesa

tebelele baze de date. Odată ce ii se acordă permisiunea, utilizatorii finali pot citi și manipula date prezente. Riscul de scurgeri de date este mereu prezent. În această situație, CRUD este util, doarece permite crearea de roluri de aplicație. Utilizând aceste roluri, se poate beneficia de un nivel ridicat de securitate integrată pentru baza de date și de control al permisiunilor de acces.

* 1. **Asemănări între REST și CRUD**

Aplicațiile REST sunt dezvoltate având la bază un anumit set de resurse.

Aceste resurse, la fel ca resursele CRUD, pot fi ușor create, citite, actualizate și șterse. În loc să se folosească termenii de creare, citire, actualizare și ștergere, în REST se utilizează PUT / POST, GET, PATCH / POST, și DELETE pentru a interacționa cu aceste resurse.

* 1. **Difernțe între REST și CRUD**

Există diferențe notabile între REST și CRUD. În primu rând REST este

recunoscut ca un cadru arhitectural, în timp ce CRUD este în primul rând un concept funțional. REST se concentrează în principal ăe utilizarea componentelor HTTP pentru gestionarea și interacțiunea cu resursele. Pe de altă parte, CRUD se axează pe manipularea datelor stocate într-o bază de date. Deși CRUD poate fi incorporat într-o implementare REST. Este important de menționat că REST poate funcșiona independent și eficient fară a se baza pe operațiile CRUD.

**Fig. 4-2 Diferența între CRUD și REST**

*Sursa:* [*https://nordicapis.com/crud-vs-rest-whats-the-difference/*](https://nordicapis.com/crud-vs-rest-whats-the-difference/)

1. **Programe folosite în realizarea website-ului**

În realizarea website-ului sunt folostie multe programe printre care sunt:

Pentru back-end:

* Visual Studio 2022 (C# / .Net)
* Postman
* MySQL (baza de date)

Pentru front-end:

* Visual Studio Code (HTML, CSS, JavaScript)

Pentru UI / UX:

* Lunacy
  1. **Despre Visual Studio 2022**

Visual Studio este un mediu integrat de dezvoltare (IDE) cuprinzător pe

care îl poți folosi pentru a scrie, edita și construi cod și apoi pentru a implementa aplicația. Visual Studio include compilatoare, instrumente de completare a codului, control al versiunilor, extensii și multe alte funcționalități care îmbunătățesc etapă a procesului de dezvoltare a software-ului.