Aplicație web pentru monitorizarea consumului de energie al electrocasnicelor

PROIECT DE DIPLOMĂ

Autor: **Ioana-Alina Hetea**

Conducător științific: SL. Dr. Ing. Mihai HULEA

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

Autor: **Ioana-Alina Hetea**

Aplicație web pentru monitorizarea consumului de energie al electrocasnicelor

1. **Enunțul temei:** *Aplicație web pentru monitorizarea consumului de energie al electrocasnicelor în locuințe*
2. **Conținutul proiectului:** *(enumerarea părților componente) Pagina de prezentare, Declarație privind autenticitatea proiectului, Sinteza proiectului, Cuprins, Titlul capitolului 1, Titlul capitolului 2,… Titlul capitolului n, Bibliografie, Anexe.*
3. **Locul documentării:** *Universitatea Tehnică din Cluj-Napoca*
4. **Consultanți:**
5. **Data emiterii temei:**
6. **Data predării:**

Semnătura autorului

Semnătura conducătorului științific

**Declaraţie pe proprie răspundere privind**

**autenticitatea proiectului de diplomă**

Subsemnatul(a) **Ioana-Alina Hetea**  , legitimat(ă) cu CI/BI seria CJ nr. 373447, CNP 6000917125808,

autorul lucrării:

APLICAȚIE WEB PENTRU MONITORIZAREA CONSUMULUI DE ENERGIE AL ELECTROCASNICELOR

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 2023 a anului universitar 2022-2023, 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 Ioana-Alina HETEA

(semnătura)

**SINTEZA**

proiectului de diplomă cu titlul:

Titlul lucrării

Autor: **Prenume NUME**

Conducător științific: **Titlu. ing. Prenume NUME**

1. Cerințele temei:

2. Soluții alese:

3. Rezultate obținute:

4. Testări și verificări:

5. Contribuții personale:

6. Surse de documentare:

Semnătura autorului

Semnătura conducătorului științific

Cuprins

[1 Introducere 2](#_Toc137300571)

[1.1 Context general 2](#_Toc137300572)

[1.2 Obiective 3](#_Toc137300573)

[1.3 Specificații 3](#_Toc137300574)

[2 Studiu bibliografic 5](#_Toc137300575)

[2.1 Prezentarea problemei și a contextului de dezvoltare 5](#_Toc137300576)

[2.1.1 Importanța monitorizării consumului de energie electrică în locuințe 5](#_Toc137300577)

[2.1.2 Impactul consumului de energie electrică asupra mediului și schimbărilor climatice 6](#_Toc137300578)

[2.1.3 Necesitatea unei soluții eficiente pentru monitorizarea consumului de energie electrică în locuințe l7](#_Toc137300579)

[2.2 Concepte de bază 8](#_Toc137300580)

[2.2.1 Arhitectura REST 8](#_Toc137300581)

[2.2.2 Arhitectura MVC 11](#_Toc137300582)

[3 Analiză 13](#_Toc137300583)

[3.1 Analiza publicului țintă 13](#_Toc137300584)

[3.2 Analiza concurenței 14](#_Toc137300585)

[3.3 Analiza tehnologiilor utilizate 17](#_Toc137300586)

[3.3.1 VSCode – mediul de lucru 17](#_Toc137300587)

[3.3.2 JavaScript – limbaj de programare 18](#_Toc137300588)

[3.3.3 MongoDB – baza de date 19](#_Toc137300589)

[3.3.4 React JS 21](#_Toc137300590)

[3.3.5 Node.js 22](#_Toc137300591)

[4 Proiectare 23](#_Toc137300592)

[4.1 Arhitectura aplicației 23](#_Toc137300584)

[4.1.1 Arhitectura client-server 23](#_Toc137300587)

[4.1.2 Componenta de client 23](#_Toc137300588)

[4.1.3 Componenta de server 23](#_Toc137300588)

[4.1.4 Comunicarea între client și server 25](#_Toc137300588)

[4.2 Detalii de proiectare 25](#_Toc137300584)

[4.2.1 Cazuri de utilizare 25](#_Toc137300587)

[4.2.2 Procesarea datelor și fluxul de comunicare al aplicației 27](#_Toc137300588)

[4.2.3 Interfața utilizatorului 31](#_Toc137300588)

[4.2.4 Comunicarea între client și server 25](#_Toc137300588)

[5 implementare 33](#_Toc137300593)

[5.1 Baza de date (MongoDB) 33](#_Toc137300594)

[5.1.1 Alegerea sistemului de gestionare a bazei de date 33](#_Toc137300587)

[5.1.2 Implementarea și proiectarea structurii bazei de date 34](#_Toc137300588)

[5.1.2.1 Colecția users 35](#_Toc137300588)

[5.1.2.2 Colecția appliances 36](#_Toc137300588)

[5.1.2.1 Colecția flats 37](#_Toc137300588)

[5.2 Backend 38](#_Toc137300595)

[5.2.1 Structura de fișiere și directoare 38](#_Toc137300600)

[5.2.1.1 Directorul handlers 39](#_Toc137300588)

[5.2.1.2 Directorul middleware 40](#_Toc137300588)

[5.2.1.3 Directorul models 40](#_Toc137300588)

[5.2.1.4 Directorul routes 40](#_Toc137300588)

[5.2.1.5 Fișierul app.js 41](#_Toc137300588)

[5.2.1.6 Fișierul .env 41](#_Toc137300588)

[5.2.2 Autentificarea și autorizarea utilizatorilor 41](#_Toc137300600)

[5.3 Frontend 42](#_Toc137300595)

[5.3.1 Structura de fișiere și directoare 42](#_Toc137300600)

[5.3.1.1 Directorul public 42](#_Toc137300588)

[5.3.1.2 Directorul src 43](#_Toc137300588)

[5.3.2 Structura de fișiere și directoare 42](#_Toc137300600)

[6 Reguli de formatare 28](#_Toc137300596)

[6.1 Formatarea paginii 28](#_Toc137300597)

[6.2 Titluri și stiluri 28](#_Toc137300598)

[6.3 Figuri, tabele și ecuații 29](#_Toc137300599)

[6.3.1 Figuri 29](#_Toc137300600)

[6.4 Tabele 29](#_Toc137300601)

[6.5 Ecuații 29](#_Toc137300602)

[6.6 Referințe bibliografice 30](#_Toc137300603)

[7 Bibliografie 31](#_Toc137300604)

# Introducere

## Context general

În ultimele decenii, conștientizarea și preocuparea pentru problemele de mediu au devenit tot mai importante în societatea contemporană. Energia electrică reprezintă un domeniu de interes major, cu o creștere semnificativă a consumului și costurilor asociate. Din acest motiv, este crucială implementarea unor soluții de monitorizare a consumului electric și de optimizare a cheltuielilor.

Importanța acestei lucrări de licență constă în faptul că abordează o nevoie actuală în domeniul energetic și al eficienței energetice la nivelul locuințelor. O astfel de aplicație web poate aduce beneficii semnificative, atât pentru utilizatori cât si pentru mediu. Utilizatorii vor obține o perspectivă mai clara asupra consumului lor individual de energie electrică și vor fi încurajați să adopte un comportament mai responsabil asupra consumului în exces.

Prin intermediul acestei aplicații, utilizatorii vor beneficia de o modalitate simplă și intuitivă de a monitoriza și gestiona consumul de energie electrică în propriile locuințe. Ei își vor putea înregistra locuințele și vor asocia fiecărei locuințe electrocasnicele folosite, permițând generarea unor grafice privind costul și consumul electric. Astfel, aplicația oferă utilizatorului posibilitatea de a lua ulterior decizii relevante în vederea reducerii consumului aferent.

Această aplicație de monitorizare a cheltuielilor de energie electrică se remarcă prin câteva aspecte teoretice și practice. Unul dintre aspectele teoretice importante este legat de importanța monitorizării și optimizării consumului de energie electrică în locuințe. Sectorul rezidențial are o contribuție semnificativă la consumul total de energie electrică, astfel, implementarea unei aplicații de monitorizare în apartamentele individuale poate avea un impact pozitiv în reducerea consumului global de energie. Conștientizarea locatarilor în urma informării acestora, îi poate încuraja să adopte practici mai sustenabile și să reducă consumul inutil de energie.

În plus, integrarea tehnologiilor React, Node.js și Express în dezvoltarea acestei aplicații web aduce câteva avantaje practice remarcabile. Utilizarea React-ului permite crearea unei interfețe moderne care oferă utilizatorului o experiență plăcută, iar Node.js și Express asigură o performanță excelentă, cu timp de răspuns rapid. Așadar, aceste tehnologii permit dezvoltarea rapidă și flexibilă a aplicației, facilitând extinderea și adaptarea sa ulterioară.

Monitorizarea consumului de energie al electrocasnicelor facilitează identificarea aparatelor care înregistrează consumuri mai ridicate și încurajează înlocuirea lor cu modele mai eficiente. Prin urmare, prin conștientizarea și educația în domeniul energiei se poate obține o gestionare mai bună a consumului de energie și se pot promova obiceiuri mai sustenabile în rândul populației.

## Obiective

Scopul acestei lucrări de licență constă în implementarea unei soluții de monitorizare a consumului și prețului energiei electrice asociate electrocasnicelor din locuințe. Prin intermediul aplicației web, utilizatorilor li se oferă posibilitatea de a adăuga informații despre locuințele lor și despre electrocasnicele asociate acestora, iar datele colectate sunt utilizate pentru a genera grafice. Funcționalitățile implementate vizează următoarele aspecte:

* Creare cont de utilizator
* Accesare cont (Log in)
* Adăugare locuință utilizator
* Modificare locuință utilizator
* Ștergere locuință utilizator
* Vizualizarea tuturor locuințelor utilizatorului
* Adăugare electrocasnic asociat locuinței
* Calculul automat al prețului consumului de energie în funcție de fiecare electrocasnic adăugat pe perioada selectată de către utilizator
* Modificarea electrocasnic asociat locuinței
* Ștergerea electrocasnic asociat locuinței
* Vizualizarea tuturor electrocasnicelor
* Generare grafice, vizualizare consum electric și preț consum electric

Prin implementarea acestor funcționalități, utilizatorii vor avea posibilitatea de a monitoriza și gestiona eficient consumul și costurile de energie electrică asociate electrocasnicelor din locuințele lor.

Astfel, prin intermediul acestor funcționalități s-au atins următoarele obiective:

* Crearea unei aplicații web care permite utilizatorilor posibilitatea de a monitoriza atât consumul de energie electrică al electrocasnicelor cât și prețul consumului
* Dezvoltarea unui interfețe intuitive cu ajutorul HTML, prin intermediul librăriei CSS și a bibliotecii Material-ui.
* Crearea si dezvoltarea unei baze de date pentru gestionarea și stocarea datelor precum: utilizatorii, electrocasnicele și locuințele.

## Specificații

Folosind această aplicație utilizatorul are opțiune de a se autentifica sau de a își crea un nou cont. În situația în care utilizatorul dorește să se înregistreze în aplicație, acesta trebuie sa completeze câmpurile necesare, iar pentru o mai bună validare a datelor, este necesară transmiterea atât a unui număr de telefon valid, cât și a CNP-ului. În urma creării contului, utilizatorul trebuie să își acceseze contul cu emailul și parola furnizată.

După conectarea acestuia în aplicație, utilizatorul va avea posibilitatea atât de a adăuga propriile locuințe cu detaliile aferente, cât și de a le modifica sau șterge din baza de date. În urma adăugării locuințelor, pentru a adăuga un electrocasnic asociat oricărei locuințe, va trebui să insereze: consumul electrocasnicului măsurat în watt, timpul exprimat în ore al funcționării acestuia și să selecteze perioada determinată folosirii lui. În urma completării acestor câmpuri, prețul consumatorului se calculează automat cu prețul implicit de KWH de 0.80 lei.

Formula de calcul al prețului final al electrocasnicului:

pret\_total = (putere\* nr\_ore)/1000 \* nr\_zile \* 0.80

În cazul introducerii greșite sau al nefolosirii unui electrocasnic, acesta are opțiunea de a șterge sau modifica consumatorul respectiv. Aplicația oferă posibilitatea de afișare a tuturor locuințelor înregistrate și a tuturor consumatorilor de energie electrică asociați acestora sub formă de tabel asigurând astfel o vizualizare clară și structurată.

Pe pagina principală utilizatorul va putea vizualiza un grafic care prezintă prețul total al consumului de energie electrică pentru fiecare locuință, oferind astfel o perspectivă asupra consumului electrocasnicelor. Tot pe pagina principală, utilizatorul poate observa și locuințele acestuia, cât și electrocasnicele asociate locuinței.

Interfața este intuitivă și prietenoasă cu utilizatorul, aceasta permite utilizatorului să navigheze cu ușurință prin diversele funcționalități și să acceseze informațiile relevante. Aplicația oferă o performanță rapidă și fără întârzieri, astfel încât datele sunt actualizate în timp real. Răspunsul la acțiunile utilizatorului sunt optimizate pentru a asigura o experiență plăcută și fluidă. Aplicația asigură securitatea datelor utilizatorilor, precum autentificarea acestora și criptarea parolei.

# Studiu bibliografic

## Prezentarea problemei și a contextului de dezvoltare

### Importanța monitorizării consumului de energie electrică în locuințe

Monitorizarea consumului de energie electrică din locuințe joacă un rol important în promovarea economisirii costurilor și în conștientizarea utilizatorilor asupra reducerii impactului acesteia asupra mediului. Prin gestionarea și evaluarea adecvată a consumului de energie electrică, utilizatorii pot contribui la crearea unei societăți mai durabile și la promovarea unei utilizări responsabile a resurselor energetice.

Prin colectarea datelor precise privind consumul de energie electrică la nivelul aparatelor electrocasnice, utilizatorii pot identifica sursele principale de consum și pot lua măsuri specifice pentru a îmbunătăți eficiența electrică. Acesta poate implica înlocuirea aparatelor vechi și ineficiente cu unele mai performante energetic, ajustarea programelor de utilizare sau adoptarea unor practici mai responsabile în utilizarea energiei.

Un aspect important legat de monitorizarea consumului de energie electrică în locuințe este economisirea costurilor. Prin identificarea aparatelor care consumă în mod excesiv sau care prezintă defecțiuni, utilizatorii pot lua măsuri pentru a reduce facturile lunare de energie electrică. Astfel, aceasta permite utilizatorilor să fie mai informați și să ia decizii bine fundamentate în ceea ce privește consumul și costurile asociate. Prin vizualizarea și analiza datelor, utilizatorii devin conștienți de cantitatea de energie consumată de diferitele aparate și pot identifica zonele în care pot face schimbări. Conștientizarea aceasta poate încuraja adoptarea unui comportament mai responsabil și sustenabil în ceea ce privește consumul de energie.

Articolul intitulat “Determinants with Impact on Romanian Consumers’ Energy-Saving Habits” [1] este concentrat pe analiza și identificarea factorilor care influențează comportamentul de economisire a energiei electrice al românilor. Acesta investighează mai multe aspecte, inclusiv nivelul de conștientizare a consumatorilor în ceea ce privește importanța economisirii energiei, influența factorilor socio-demografici și a percepțiilor referitoare la măsurile de eficiență energetică. Concluziile studiului subliniază importanța informării și educației în promovarea comportamentului de economisire a energiei, necesitatea de a oferi măsuri concrete și accesibile pentru eficiența energetică în locuințe și importanța implicării tuturor părților interesate, inclusiv guvernul, organizațiile non-guvernamentale și furnizorii de energie, în promovarea unei culturi a economisirii energetice în România. Toate acestea afectează in mod pozitiv indivizii în încurajarea implementării acestui obicei de economisire a energiei în viața lor. Monitorizarea este un aspect important, deoarece furnizează informații valoroase, iar acest lucru permite utilizatorului să identifice și să înțeleagă modul în care energia este utilizată într-o locuință.

Un alt studiu subliniază importanța înțelegerii factorilor care influențează comportamentul consumatorilor casnici de energie electrică într-un context de piață în schimbare. Aceste constatări pot fi utilizate pentru a dezvolta strategii și politici mai eficiente în promovarea unui consum responsabil și eficient de energie electrică în gospodării. Prin creșterea conștientizării și oferirea de informații relevante, se poate încuraja adoptarea unor practici sustenabile și economisirea de energie în rândul consumatorilor casnici din România, contribuind astfel la reducerea impactului asupra mediului și la promovarea unei dezvoltări durabile. [2]

### Impactul consumului de energie electrică asupra mediului și schimbărilor climatice

Impactul consumului de energie electrică asupra mediului și schimbărilor climatice este un subiect de mare relevanță în contextul preocupărilor globale privind durabilitatea și protecția mediului înconjurător. Consumul de energie electrică contribuie în mod semnificativ la emisiile de gaze cu efect de seră, care sunt responsabile de încălzirea globală și schimbările climatice.

În primul rând, producția de energie electrică utilizând surse tradiționale precum combustibilii fosili generează emisii semnificative de dioxid de carbon (CO2) și alte gaze cu efect de seră. Aceste emisii contribuie la creșterea concentrației acestor gaze în atmosferă, ceea ce duce la creșterea temperaturilor globale și la fenomene climatice extreme precum furtuni puternice, secete sau inundații.

În al doilea rând, utilizarea intensivă a energiei electrice în diverse sectoare, cum ar fi transportul, industria și locuințele, necesită o producție crescută de energie, care în mare parte provine din surse neregenerabile. Aceasta poate duce la exploatarea excesivă a resurselor naturale și la impactul negativ asupra ecosistemelor.

De asemenea, consumul excesiv de energie electrică poate contribui la creșterea cererii de energie, ceea ce poate duce la construcția de noi centrale energetice și infrastructură asociată, generând impacturi adiționale asupra mediului, precum defrișări și poluare.

Monitorizarea și reducerea consumului de energie electrică într-o locuință au un impact semnificativ în reducerea emisiilor de gaze cu efect de seră și a amprentei de carbon. Prin intermediul tehnologiilor eficiente și a practicilor de utilizare responsabilă a energiei, utilizatorii pot reduce consumul de energie electrică, promovând astfel o dezvoltare mai sustenabilă.

Impactul consumului de energie electrică asupra mediului și schimbărilor climatice este un aspect crucial în contextul preocupărilor globale privind sustenabilitatea și protecția mediului înconjurător. Monitorizarea și reducerea consumului de energie electrică în locuințe reprezintă o măsură importantă în lupta împotriva schimbărilor climatice și în promovarea unei dezvoltări durabile.

Prin reducerea consumului de energie electrică în locuințe, demonstrăm o atitudine responsabilă față de mediu și o preocupare pentru viitorul sustenabil al planetei. Ne implicăm activ în eforturile de protecție a mediului și de construire a unei societăți durabile.

Având grijă de mediu și reducând consumul de energie electrică în locuințe, contribuim la reducerea emisiilor de gaze cu efect de seră, conservarea resurselor naturale, economisirea financiară, îmbunătățirea calității aerului și manifestăm responsabilitate civică în promovarea durabilității și protecției mediului înconjurător.

Raportul “Energy Efficiency 2021: Analysis and Outlooks to 2040” [3] oferă o analiză detaliată a eficienței energetice în domeniul electrocasnicelor. Studiul evidențiază importanța reducerii consumului de energie electrică al electrocasnicelor în gospodării și identifică o serie de oportunități și provocări în acest domeniu. Raportul subliniază că electrocasnicele reprezintă o parte semnificativă a consumului de energie în gospodării și că îmbunătățirea eficienței lor energetice poate aduce beneficii semnificative în reducerea consumului total de energie și a emisiilor de gaze cu efect de seră. Studiul evidențiază că există o gamă largă de electrocasnice disponibile pe piață, de la frigidere și mașini de spălat până la televizoare și computere, iar eficiența energetică variază considerabil între acestea. Astfel, se remarcă necesitatea promovării și adoptării de către consumatori a electrocasnicelor cu un consum redus de energie, precum și necesitatea dezvoltării și implementării standardelor și etichetelor energetice pentru a furniza informații clare și comparative consumatorilor.

În concluzie, raportul evidențiază că eficiența energetică a electrocasnicelor este un aspect crucial în reducerea consumului de energie electrică și a impactului asupra mediului. Adoptarea de electrocasnice eficiente din punct de vedere energetic și promovarea unui comportament responsabil în ceea ce privește utilizarea lor pot contribui semnificativ la atingerea obiectivelor de sustenabilitate și reducere a emisiilor de gaze cu efect de seră.

### Necesitatea unei soluții eficiente pentru monitorizarea consumului de energie electrică în locuințe

Necesitatea unei soluții eficiente pentru monitorizarea consumului de energie electrică în locuințe este evidențiată în contextul creșterii importanței eficienței energetice și a sustenabilității. Această secțiune examinează motivele și beneficiile unei astfel de soluții, precum și implicarea sa practică în promovarea utilizării responsabile a energiei.

Monitorizarea consumului de energie electrică în locuințe devine esențială pentru mai multe motive. În primul rând, permite conștientizarea consumului individual și colectiv de energie, oferind informații relevante despre modul în care sunt utilizate resursele energetice. Astfel, utilizatorii pot identifica și înțelege modelele de consum și pot lua măsuri pentru a-l optimiza și reduce.

În al doilea rând, o soluție eficientă de monitorizare poate servi ca instrument de gestionare a energiei, ajutând utilizatorii să-și urmărească și să-și controleze consumul

în timp real. Aceasta permite identificarea consumatorilor energetici ineficienți sau comportamentelor de utilizare a energiei care pot fi îmbunătățite. Utilizatorii pot lua decizii informate și pot implementa strategii de economisire a energiei pentru a reduce facturile de utilități și a contribui la conservarea resurselor.

În plus, monitorizarea consumului de energie electrică în locuințe este importantă și în contextul schimbărilor climatice. Prin conștientizarea și reducerea consumului de energie, se poate contribui la scăderea emisiilor de gaze cu efect de seră și la protejarea mediului înconjurător. De asemenea, promovarea unei utilizări eficiente a energiei poate stimula adoptarea surselor de energie regenerabilă și a practicilor sustenabile în cadrul locuințelor.

Implementarea unei soluții eficiente pentru monitorizarea consumului de energie electrică în locuințe are implicații practice semnificative. Aceasta presupune dezvoltarea și implementarea unor tehnologii avansate de măsurare și monitorizare a consumului, precum și furnizarea de instrumente și aplicații accesibile utilizatorilor pentru a-și gestiona și controla consumul de energie. De asemenea, implică educația și conștientizarea utilizatorilor cu privire la importanța și beneficiile utilizării responsabile a energiei în locuințe.

Necesitatea unei soluții eficiente pentru monitorizarea consumului de energie electrică în locuințe este justificată de importanța eficienței energetice, sustenabilității și responsabilității față de mediul înconjurător. Aceasta oferă utilizatorilor posibilitatea de a conștientiza, gestiona și reduce consumul de energie electrică într-un mod practic și eficient, contribuind la atingerea obiectivelor de economisire a energiei și protejare a mediului.

Articolul [4] evidențiază că monitoarele de energie oferă informații în timp real și vizualizări intuitive ale consumului de energie, ceea ce permite gospodăriilor să-și monitorizeze și să-și înțeleagă mai bine modelele de consum. Acest lucru duce la o mai mare conștientizare a impactului lor asupra mediului și motivează utilizatorii să adopte comportamente mai eficiente din punct de vedere energetic.

În concluzie, articolul evidențiază importanța feedback-ului în timp real privind consumul de energie oferit de monitoarele inteligente și impactul acestuia asupra comportamentului de consum de energie în gospodării. Studiul sugerează că astfel de tehnologii pot juca un rol crucial în încurajarea consumului sustenabil de energie și în reducerea impactului asupra mediului.

## Concepte de bază ale aplicației

### Arhitectura REST

REST (Representational State Transfer) este un stil de arhitectură pentru a dezvolta servicii web, care utilizează protocolul HTTP ca interfață de comunicație pentru a transfera date prin metode HTTP [5]. Este un concept important în dezvoltarea aplicațiilor web moderne care facilitează construirea lor eficient și scalabil.



Figura 2‑1 Diagramă arhitectură REST

REST a fost propus inițial ca stil arhitectural software pentru sisteme distribuite în teza de doctorat a lui Roy Fielding (Fielding, 2000) [6]. Se poate obține o arhitectură robustă și ușor de întreținut, prin respectarea următoarelor constrângeri fundamentale:

* Modelul client-server: această constrângere presupune delimitarea evidentă dintre client și server. Responsabilitatea serverului este de a gestiona resursele, în timp ce clientul interacționează cu acestea prin intermediul cererilor și al răspunsurilor. Toate acestea facilitând dezvoltarea independentă a acestor componente.
* Fără stare(stateless): această abordare favorizează scalabilitatea și flexibilitatea serviciilor web prin faptul că serverul nu păstrează starea între cererile consecutive provenite de la același client.
* Cache: reprezintă utilizarea mecanismelor de stocare temporară a răspunsurilor de la server la nivelul clientului sau serverului, cu scopul de a reduce traficul de rețea și a îmbunătăți performanța sistemului
* Interfață uniformă: prin aplicarea acestei constrângeri, se facilitează interoperabilitatea, dezvoltarea independentă a componentelor client și server, precum și evitarea dependențelor strânse între acestea prin faptul că implică definirea unei interfețe standardizate între client și server
* Stratificare(layered system): permite adăugarea de straturi intermediare între server și client fără afectarea comunicării dintre cele două
* Cod la cerere: posibilitatea clientului de a trimite cod sau funcții personalizate către server pentru a fi executate

Scopul arhitecturii REST este de a oferi un model simplu și scalabil pentru dezvoltarea aplicațiilor și serviciilor web. Prin promovarea principiilor bazate pe resurse, operațiuni HTTP și transferul de stare reprezentativă, REST facilitează dezvoltarea eficientă și interoperabilitatea între diferite aplicații și platforme, oferind în același timp flexibilitate și scalabilitate necesare în mediul web.

Protocolul HTTP este un protocol fundamental în arhitectura WWW și servește ca mecanism de bază pentru comunicarea între client și server în transferul resurselor web. Acesta definește și diversele coduri de stare pentru a indica rezultatul cererii, precum 200 (OK), 404 (Not Found), 500 (Internal Server Error) etc. HTTP este bazat pe principiile arhitecturale REST și utilizează metodele de solicitare GET, POST, PUT, DELETE pentru a manipula resursele web. Clientul inițiază o cerere HTTP către server, specificând metoda de solicitare, URL-ul resursei și alte antete opționale care pot conține informații suplimentare despre cerere.

Serverul procesează cererea și returnează un răspuns HTTP care conține codul de stare, antetele și, opțional, un conținut asociat.

Metodele de solicitare ale HTTP-ului prin intermediul cărora un client interacționează cu serverul:

* GET: metodă folosită pentru a solicita resurse de la server (obținerea informațiilor)
* POST: această metodă este folosită pentru trimiterea datelor către server pentru a fi procesate. Este utilizată de obicei pentru crearea unei noi resurse.
* PATCH: aceasta permite modificarea parțială a resursei existente. Prin intermediul acestei metode, clientul poate trimite o reprezentare a modificărilor dorite, iar serverul va aplica acele modificări asupra resursei.
* DELETE: metodă utilizată pentru a șterge o resursă existentă de pe server. Prin intermediul acestei metode, clientul poate solicita ștergerea resursei specificate.

API-urile RESTful (Interfețele de Programare a Aplicațiilor) respectă principiile REST și utilizează HTTP ca protocol de comunicare. Prin intermediul API-urilor, dezvoltatorii pot accesa și utiliza funcționalități și resurse din alte aplicații sau servicii, fără a fi nevoie să dezvolte acele funcționalități de la zero. API-urile permit transmiterea și primirea de date între diferite aplicații, oferind astfel posibilitatea de a integra și interconecta diverse sisteme și servicii.

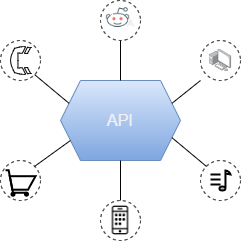


Figura 2‑2 Structura API

API-urile pot fi de diferite tipuri, cum ar fi API-uri web, API-uri de sistem de operare, API-uri de baze de date etc. API-urile web sunt printre cele mai comune și sunt folosite pentru a permite comunicația între aplicații web și alte aplicații sau servicii web.

Ele sunt bazate pe protocoale precum HTTP și permit transferul de date în formate standardizate precum JSON sau XML.

Arhitectura REST promovează separarea clară a responsabilităților între client și server, facilitând dezvoltarea aplicațiilor web distribuite, orientate pe resurse și adaptate la nevoile schimbătoare ale pieței și ale utilizatorilor.

### Arhitectura MVC

Arhitectura Model-View-Controller (MVC) este un model arhitectural utilizat în dezvoltarea aplicațiilor software, care urmărește separarea clară a responsabilităților între diferitele componente ale sistemului. Prin utilizarea acestei arhitecturi, se urmărește obținerea unei structuri modulare, ușor de întreținut și reutilizabilă.

Arhitectura MVC este alcătuită din trei componente principale: Model, View și Controller.

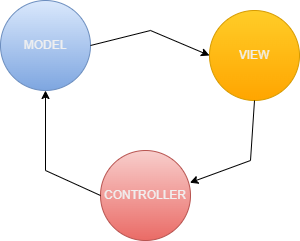


Figura 2‑3 Arhitectura MVC

Modelul este componenta centrală a arhitecturii MVC și reprezintă partea responsabilă de logica de afaceri a aplicației și gestionarea datelor. Aceasta implică manipularea datelor, inclusiv extragerea, salvarea, actualizarea și ștergerea acestora. Modelul implementează, de asemenea, regulile de afaceri specifice aplicației, asigurând validarea și consistența datelor. Vizualizarea este componenta care se ocupă de prezentarea datelor utilizatorului și de interfața grafică a aplicației. Aceasta primește datele de la controller și le afișează într-un mod adecvat utilizatorului. Vizualizarea poate fi o pagină HTML, o interfață grafică sau un șablon de afișare.

Modelul este conceput pentru a fi independent de interfața utilizatorului și de modulul de afișare, concentrându-se exclusiv pe procesarea datelor și interacțiunea cu baza de date sau alte sisteme externe. Acesta utilizează adesea tehnologii specifice, cum ar fi ORM (Object-Relational Mapping) pentru interacțiunea cu baza de date, servicii web pentru comunicarea cu alte sisteme sau biblioteci și framework-uri pentru gestionarea datelor și logicii de afaceri.

Modelul poate include, de asemenea, componente suplimentare, cum ar fi clase de acces la date, care facilitează interacțiunea cu baza de date și gestionarea operațiilor CRUD (Create, Read, Update, Delete), și clase de servicii care încapsulează logica de afaceri complexă și funcționalitățile specifice aplicației.

Controlorul gestionează interacțiunea dintre utilizator și sistem, precum și fluxul de date între model și vizualizare. Acesta primește solicitările utilizatorului, apelează metodele corespunzătoare în model pentru a obține și manipula datele necesare, apoi pregătește datele pentru a fi afișate în vizualizare. Un aspect important al controlorului este rutarea solicitărilor. Acesta determină care acțiuni trebuie executate în funcție de solicitarea primită, direcționând fluxul de control către metodele relevante din model. Controlorul gestionează, de asemenea, aspecte cum ar fi validarea datelor introduse de utilizator și gestionarea erorilor sau excepțiilor care pot apărea în timpul procesării.

Prin intermediul controlorului, se realizează separarea clară între logica de afaceri și interfața utilizatorului. Controlorul nu conține logică de afișare, ci se concentrează exclusiv pe gestionarea datelor și interacțiunea cu modelul și vizualizarea. Această separare permite o dezvoltare modulară a aplicației, facilitând testarea unitară și reutilizarea componentelor.

Vizualizarea în arhitectura MVC reprezintă partea responsabilă de prezentarea datelor utilizatorului și interacțiunea cu acesta. Aceasta este responsabilă pentru afișarea informațiilor și interfața grafică a aplicației, oferind utilizatorului posibilitatea de a interacționa cu datele și de a transmite comenzi către controlor și model.

Un aspect important al vizualizării este faptul că aceasta nu conține logica de afaceri sau manipularea datelor. În schimb, vizualizarea se concentrează pe prezentarea coerentă și estetică a informațiilor utilizatorului. Aceasta poate include elemente grafice, interfețe intuitive și funcționalități specifice pentru a facilita interacțiunea și înțelegerea datelor.

Deși vizualizarea primește datele de la controlor, aceasta nu interacționează direct cu modelul. Vizualizarea este o componentă pasivă a arhitecturii MVC, care primește datele și le afișează într-un mod corespunzător, dar nu le modifică sau le manipulează direct. Orice acțiune sau modificare a datelor se face prin intermediul controlorului.

Prin adoptarea arhitecturii MVC, dezvoltatorii pot beneficia de o structură mai organizată a aplicațiilor, ușurând procesul de dezvoltare, întreținere și extindere a acestora. Arhitectura MVC promovează dezvoltarea paralelă a componentelor și permite echipei de dezvoltare să lucreze independent asupra fiecărei părți a sistemului, optimizând astfel eficiența și calitatea dezvoltării software.

# Analiză

## Analiza publicului țintă

Analiza publicului țintă în cadrul acestei aplicații de monitorizare a consumului electric are o importanță majoră, având impact atât la nivel individual, prin satisfacția și economiile obținute de utilizatori, cât și la nivel social și ambiental, prin promovarea eficienței energetice și a unui stil de viață sustenabil.

În ceea ce privește vârsta, publicul țintă poate cuprinde indivizi din diverse grupe de vârstă, cu toate acestea predominanța este probabil să se regăsească în rândul adulților, care dețin locuințe și au responsabilitatea gestionării consumului de energie electrică. În ceea ce privește locația, aplicația poate fi destinată utilizatorilor din diferite regiuni geografice, fiind important să se ia în considerare particularitățile locale și reglementările referitoare la consumul de energie electrică.

Publicul țintă se caracterizează prin interesul pentru monitorizarea și gestionarea consumului de energie electrică, fiind motivați de economisirea energiei și reducerea costurilor asociate prin intermediul acestor practici. De asemenea, utilizatorii ar trebui să manifeste o familiaritate cu tehnologia și să fie predispuși să utilizeze aplicația pentru a accesa și gestiona informațiile legate de consumul de energie electrică.

În ceea ce privește aspectele psihologice, utilizatorii prezintă o preocupare pentru mediu și impactul consumului de energie electrică asupra acestuia. Drept urmare, aceștia pot fi motivați să reducă consumul de energie pentru a contribui la protejarea mediului înconjurător. De asemenea, orientarea către economisirea financiară poate reprezenta o trăsătură semnificativă a publicului țintă, cu dorința de a economisi bani prin reducerea consumului de energie electrică și prin adoptarea de comportamente eficiente energetic.

Publicul vizat este format din persoane adulte, care dețin/locuiesc într-o locuință și au responsabilitatea gestionării consumului de energie electrică. Aceste persoane pot fi încadrate într-o gamă largă de grupe de vârstă, dar un accent deosebit este pus pe persoanele adulte. Acestea, în general, au dobândit o anumită experiență și maturitate în gestionarea resurselor, inclusiv a energiei electrice. Ei pot fi mai receptivi la informațiile și sfaturile practice referitoare la reducerea consumului de energie și la adoptarea unui comportament mai sustenabil.

Așadar, prin identificarea categoriilor de utilizatori, cum ar fi proprietarii de locuințe, chiriașii, eco-entuziaștii și proprietarii de afaceri preocupați de economisirea consumului de energie electrică, se poate adapta aplicația la cerințele specifice ale fiecărei grupări. Prin promovarea unui stil de viață sustenabil și a practicilor eficiente energetic, aplicația poate avea un impact pozitiv atât la nivel individual, cât și la nivelul comunităților și mediului înconjurător.

Prima categorie care ar putea dezvolta un interes pentru această aplicație dezvoltată sunt proprietarii de locuințe.

Această categorie de utilizatori este reprezentată de persoanele care sunt proprietari de case sau apartamente și doresc să monitorizeze și să optimizeze consumul și costul energetic al locuințelor lor. Motivațiile acestor utilizatori pot fi diverse, cum ar fi identificarea aparatelor cu consum energetic ridicat, înlocuirea acestora cu aparate mai eficiente energetic sau implementarea de măsuri pentru a îmbunătăți eficiența energetică în general. Prin intermediul aplicației, utilizatorii pot obține date și analize referitoare la consumul de energie al fiecărui aparat în urma cărora pot gestiona situația personal.

O altă categorie care ar putea prezenta un interes în folosirea aplicației este formată din persoane care locuiesc în locuințe închiriate și doresc să monitorizeze consumul de energie și costurile asociate apartamentelor sau caselor în care locuiesc. Acești utilizatori sunt interesați să înțeleagă modul în care diversele aparate și activități influențează consumul de energie și costurile facturilor energetice. Aplicația poate oferi o prezentare clară și informații detaliate referitoare la consumul de energie, permițându-le să identifice sursele de consum ineficient și să adopte măsuri pentru a reduce cheltuielile cu energia.

De asemenea, există și categoria formată din persoane preocupate de protecția mediului și de conservarea energiei(eco-entuziaști). Acești utilizatori sunt motivați să reducă impactul negativ asupra mediului și să adopte un stil de viață mai sustenabil. Aplicația poate fi un instrument util pentru a oferi o perspectivă detaliată asupra consumului de energie și pentru a-i ajuta să identifice sursele de consum energetic ineficient. Prin intermediul analizelor și recomandărilor personalizate, acești utilizatori pot lua măsuri pentru a reduce consumul de energie și pentru a adopta practici mai prietenoase cu mediul.

În ansamblu, analiza publicului țintă în cadrul aplicației de monitorizare a consumului electric are o importanță majoră, având impact atât la nivel individual, prin satisfacția și economiile obținute de utilizatori, cât și la nivel social și ambiental, prin promovarea eficienței energetice și a unui stil de viață sustenabil.

## Analiza concurenței

În cadrul analizei concurenței pentru această aplicația web de monitorizare a consumului curentului electric al electrocasnicelor, prima etapă esențială constă în identificarea și delimitarea competitorilor direcți și indirecți. Competitorii direcți se referă la companii și produse care oferă servicii similare și concurează direct în același segment de piață. În schimb, competitorii indirecți reprezintă entități care pot furniza alternative sau soluții complementare în cadrul aceluiași domeniu, dar nu se concentrează exclusiv pe monitorizarea consumului de energie al electrocasnicelor.

Din diversele opțiuni disponibile utilizatorilor dornici să folosească o astfel de aplicație de monitorizare a consumului electric al electrocasnicelor dintr-o locuință, am identificat punctele forte și diferențele dintre acestea. Majoritatea aplicațiilor oferă funcționalități asemănătoare, fiecare diferențiindu-se prin interfață și prin caracteristicile care sunt puse în evidență. Unele aplicații oferă rapoarte detaliate sau posibilitatea de a salva șabloanele si de a le exporta.

În ceea ce privește competitorii direcți am identificat următoarele aplicații [7]:

1. „Electricity Cost Calculator‏‏‎” – este o aplicație web care determină energia consumul de energie al electrocasnicelor introduse. Trebuie introduse

detaliile aparatului precum și puterea de utilizare, prețul KWh și câte ore este utilizat. După adăugarea acestor informații se efectuează calculele și se obțin informații precum KWh/zi, cost/oră, cost/zi, cost/lună și cost/an.

1. „Electrical Cost‏‏‎” – Această aplicație se dovedește utilă pentru a monitoriza cantitatea de energie consumată în funcție de puterea utilizată de aparatele electrocasnice din gospodărie. Prin intermediul acestei aplicații, este posibil să se efectueze calculul consumului de energie și costul acestuia pe perioade de timp precum săptămâna, luna și anul. De asemenea, aplicația permite salvarea de șabloane și exportul de fișiere text. Utilizatorii au posibilitatea de a alege între încărcările predefinite și introducerea manuală a parametrilor. Astfel, se poate determina cu exactitate care dintre aparatele electrocasnice a consumat cantitatea cea mai mare de energie.
2. „EvoEnergy-Electricity cost calculator‏‏‎” - Această aplicație facilitează calculul costului consumului de energie electrică în gospodărie generat de diferite aparate sau dispozitive electrice și contribuie la reducerea consumului de energie. Prin intermediul acestei aplicații, putem identifica aparatul cu cel mai mare consum energetic și obține informații detaliate despre consumul de energie al fiecărui aparat. De asemenea, aplicația permite determinarea consumului de energie pe unități de timp precum oră, zi, lună și an. Aceasta acceptă diverse tipuri de unități de măsură pentru putere și curent, acoperind toate tensiunile de curent utilizate global. Funcționalitățile suplimentare ale aplicației includ posibilitatea de a configura simbolul monedei utilizate, introducerea ratingului curent al aparatului în amperi sau miliamperi și puterea în kilowați.
3. Energy Tracker – O aplicație mobilă intuitivă și ușor de utilizat, care permite utilizatorilor să monitorizeze consumul de energie al electrocasnicelor dintr-un mediu rezidențial. Aceasta oferă notificări în timp real cu privire la consumul excesiv și sugestii pentru optimizarea eficienței energetice.

În ceea ce privește competitorii indirecți ai acestei aplicații de monitorizare, se regăsesc următorii:

1. Companii de distribuție de energie electrică - Companiile active în domeniul distribuției de energie electrică pot oferi propriile aplicații web sau platforme, care permit utilizatorilor să monitorizeze în timp real consumul de energie și să primească facturi detaliate.
2. Producători de electrocasnice - Anumite întreprinderi producătoare de electrocasnice dezvoltă produse inteligente, care integrează funcționalități de monitorizare a consumului de energie. Acestea sunt accesibile utilizatorilor prin intermediul aplicațiilor mobile sau a platformelor web asociate.
3. Platforme de automatizare a casei inteligente - Platformele specializate în automatizarea caselor inteligente pot furniza funcționalități avansate, inclusiv monitorizarea consumului de energie și controlul electrocasnicelor. Aceste funcționalități sunt disponibile prin intermediul aplicațiilor web sau mobile asociate platformelor respective.

Diferența majoră a acestor aplicații este designul și interfața pentru utilizator. Acestea pot varia de la o aplicație la alta, unele sunt mai intuitive, ușor de utilizat și personalizabile, în timp ce altele pot avea o curba de învățare mai abruptă sau pot fi mai puțin intuitive. Unele aplicații pot oferi un nivel ridicat de personalizare și flexibilitate, permițând utilizatorilor să configureze setări individuale pentru fiecare electrocasnic sau să creeze profiluri personalizate. Altele pot avea opțiuni limitate și să ofere funcționalități standardizate pentru toate dispozitivele.

De asemenea, nivelul de securitate și confidențialitate al datelor poate varia între aplicațiile web. Unele aplicații pot implementa măsuri puternice de securitate și respectarea confidențialității datelor utilizatorilor, în timp ce altele pot avea lacune în acest sens. Utilizatorii trebuie să fie atenți la politicile de securitate și confidențialitate ale aplicației înainte de a utiliza serviciul.

Aplicația „Electricity Cost Calculator‏‏‎”, care este una dintre competitorii direcți, este cea mai asemănătoare cu aplicația realizată în cadrul acestei lucrări. Este o aplicație intuitivă și ușor de folosit, se remarcă prin simplicitate. Interfața pentru utilizator nu este una extravagantă, se axează doar pe principalele funcționalități cum ar fi: calculul consumului electric într-o zi/săptămână/lună/an/5 ani/10 ani, precum și costul acestuia.

Diferența dintre această aplicație pentru calculul consumului și al prețului electricității consumate de electrocasnice și aplicația realizată în cadrul acestei licențe este în primul rând interfața grafică. În aplicația dezvoltată se pot observa graficele atât pentru consumul de energie al fiecărui electrocasnic cât și al prețului fiecăruia. Datele precum locuințele și electrocasnicele sunt salvate într-o bază de date, pe când aplicația Electricity Cost Calculator nu le salvează, ci calculează în momentul introducerii datelor.

Aplicația oferă utilizatorilor posibilitatea de a accesa și gestiona datele privind consumul de energie electrică al electrocasnicelor din orice locuință deținută de aceștia. Datele sunt centralizate și stocate într-o bază de date, iar aplicația calculează atât costul, cât și consumul de energie electrică al fiecărui electrocasnic înregistrat.

Interfața aplicației este concepută pentru a fi practică, intuitivă și ușor de utilizat, facilitând introducerea rapidă a datelor de către utilizatori. Utilizatorii pot obține rezultatele sub formă de grafice, care le oferă o perspectivă vizuală asupra consumului de energie electrică al fiecărui electrocasnic și a costurilor asociate. Aceste informații ajută utilizatorii să ia decizii informate în ceea ce privește optimizarea consumului de energie și reducerea costurilor.

## Analiza tehnologiilor utilizate

Analiza tehnologiilor folosite într-o aplicație web este esențială pentru a înțelege infrastructura și arhitectura tehnologică a aplicației, pentru a asigura performanța, securitatea, scalabilitatea și extensibilitatea acesteia și pentru a facilita un proces de dezvoltare eficient.

### VSCode (Visual Studio Code) – mediul de lucru

Visual Studio Code (VS Code) este un editor gratuit de cod sursă dezvoltat de Microsoft și a câștigat popularitate considerabilă în rândul dezvoltatorilor software. Acesta oferă un mediu de dezvoltare integrat (IDE) eficient, versatil și ușor de utilizat pentru o varietate de limbaje de programare și platforme.

Unul dintre avantajele majore ale folosirii Visual Studio Code este flexibilitatea sa. Acesta suportă o gamă largă de limbaje de programare, precum C++, Python, Java, JavaScript, HTML, CSS și multe altele. Astfel, dezvoltatorii pot lucra pe multiple proiecte sau pot explora noi limbaje de programare într-un singur mediu coerent.

Un alt avantaj notabil al VS Code este ecosistemul său extensibil. Acesta dispune de o piață bogată de extensii create de comunitatea dezvoltatorilor, care îi permit să-și personalizeze și să-și extindă funcționalitatea în funcție de nevoile lor specifice. Fie că este vorba de adăugarea de instrumente de debug, integrarea cu sisteme de control al versiunilor, suport pentru framework-uri specifice sau orice altceva, utilizatorii pot găsi cu ușurință extensii adecvate pentru a-și îmbunătăți experiența de dezvoltare.

Un alt aspect important al VS Code este performanța sa excelentă. Este un editor rapid și reactiv, care gestionează cu ușurință proiecte mari și oferă sugestii de cod și completare automată în timp real fiind foarte folositoare utilizatorilor. De asemenea, are suport integrat pentru terminal, ceea ce permite dezvoltatorilor să execute și să testeze codul direct în editor, fără a fi nevoie să treacă la alte aplicații.

Interfața simplă și intuitivă a Visual Studio Code îl face accesibil și ușor de utilizat pentru dezvoltatorii de toate nivelele de experiență. Dispune de o varietate de caracteristici utile, cum ar fi navigarea rapidă prin proiecte, indentarea automată, evidențierea sintaxei și multe altele, care contribuie la creșterea productivității. Acesta are multe avantaje [8] din care se poate observa alegerea acestuia ca mediu de lucru.

Visual Studio code integrează și platforma GitHub, care oferă dezvoltatorilor un flux de lucru fără probleme pentru gestionarea controlului de versiune, colaborarea cu colegii și implementarea codului în “repositories”.

În concluzie, Visual Studio Code reprezintă o alegere excelentă pentru dezvoltatorii software datorită flexibilității sale, ecosistemului extensibil, performanței excelente și interfeței intuitive. Acesta oferă un mediu de dezvoltare puternic și prietenos, care facilitează scrierea, testarea și depanarea codului într-un mod eficient și plăcut.

Site-ul oficial [9] oferă mai multe informații noilor utilizatori.

### JS (JavaScript) – limbajul de programare

JavaScript reprezintă un limbaj de programare interpretat, de tip scripting, orientat pe obiecte, utilizat în mod obișnuit în dezvoltarea aplicațiilor web. Deși a fost inițial conceput pentru a fi utilizat într-un mediu de browser, JavaScript a evoluat și poate fi utilizat într-o varietate de contexte. [10]

Principala caracteristică a JavaScript constă în combinarea elementelor de programare orientată pe obiecte, programare procedurală și programare funcțională. Această combinație permite dezvoltatorilor să scrie cod flexibil și puternic, adaptat nevoilor specifice ale aplicațiilor web. Un aspect important al limbajului este faptul că este slab tipizat, ceea ce înseamnă că nu este necesară specificarea tipului de date al unei variabile înainte de utilizare. JavaScript suportă diverse tipuri de date, precum numere, șiruri de caractere, booleeni, obiecte și funcții.

Deoarece JavaScript este interpretat, codul este executat linie cu linie de către un interpretor în timpul rulării, fără a fi necesară o etapă de compilare prealabilă. Această caracteristică oferă dezvoltatorilor posibilitatea de a testa și dezvolta rapid aplicații, reducând timpul necesar pentru ciclul de dezvoltare.

Un aspect distinctiv al JavaScript îl reprezintă capacitatea sa de a interacționa cu elementele paginii web în timp real. Acest lucru este posibil datorită modelului Document Object Model (DOM), care permite accesul și modificarea elementelor HTML și CSS ale unei pagini web. Prin intermediul JavaScript, dezvoltatorii pot manipula și controla dinamic elementele DOM, facilitând crearea de interactivitate și dinamicitate în aplicațiile web. Interacțiunea dintre JavaScript, HTML și CSS este esențială în dezvoltarea aplicațiilor web moderne. JavaScript este utilizat pentru manipularea și controlul elementelor DOM, gestionarea evenimentelor și aplicarea logicii de afișare sau de interactivitate. HTML definește structura și conținutul paginii, în timp ce CSS definește aspectul și stilul vizual al acestor elemente.

Pentru a extinde funcționalitatea limbajului, JavaScript dispune de o gamă largă de biblioteci și framework-uri, cum ar fi jQuery, AngularJS, ReactJS și Node.js. Aceste biblioteci și framework-uri oferă un set de abstracții puternice, gestionarea evenimentelor, comunicare asincronă cu serverele și alte funcționalități avansate, contribuind astfel la dezvoltarea de aplicații web complexe și eficiente.

În concluzie, JavaScript este un limbaj de programare versatil și indispensabil în dezvoltarea aplicațiilor web moderne. Capacitatea sa de a interacționa în timp real cu elementele paginii web și de a oferi interactivitate îl fac esențial pentru dezvoltatorii web. Prin combinarea diferitelor paradigme de programare și prin intermediul unui set bogat de biblioteci și framework-uri, JavaScript oferă un mediu puternic și flexibil pentru crearea de aplicații web sofisticate și performante.

### MongoDB – baza de date

MongoDB este o bază de date NoSQL frecvent utilizată în aplicațiile web datorită caracteristicilor sale distincte. Una dintre caracteristicile cheie ale MongoDB este flexibilitatea în modelarea datelor. În locul unui model relațional rigid, datele sunt stocate sub formă de documente JSON, permițând adăugarea și eliminarea dinamică a câmpurilor fără a afecta schema globală. Aceasta oferă dezvoltatorilor o libertate mai mare în gestionarea datelor și adaptarea lor la nevoile schimbătoare ale aplicației.

Performanța este un alt aspect important al utilizării MongoDB în aplicațiile web. Baza de date este proiectată pentru a oferi o performanță rapidă și scalabilitate orizontală. Prin distribuirea datelor pe mai multe noduri și prin suportul pentru indexare și replicare, MongoDB poate gestiona eficient volume mari de trafic și oferi un timp de răspuns redus. Aceasta este o caracteristică valoroasă în aplicațiile web care necesită un acces rapid și eficient la date.

Flexibilitatea interogărilor este un alt avantaj al MongoDB. Limbajul de interogare oferă o gamă largă de opțiuni, inclusiv interogări de căutare, agregări, sortare și filtrare. Acest lucru permite dezvoltatorilor să obțină datele dorite într-un mod flexibil și eficient.

Scalabilitatea este, de asemenea, un punct forte al MongoDB. Clusterul MongoDB poate fi extins prin adăugarea de noduri suplimentare pentru a gestiona creșterea volumului de date și a numărului de utilizatori. Funcționalitățile native de fragmentare și replicare facilitează scalabilitatea orizontală și asigură redundanța datelor.

O altă caracteristică semnificativă a MongoDB este suportul nativ pentru interogări geospațiale. Aceasta permite stocarea și interogarea datelor bazate pe locație, fiind utilă în aplicațiile web care necesită funcționalități legate de geolocație, cum ar fi hărți interactive sau căutări în funcție de proximitate.

MongoDB și bazele de date relaționale prezintă diferențe semnificative în mai multe aspecte. Un aspect cheie este modelul de date pe care îl utilizează fiecare. MongoDB, ca bază de date NoSQL, folosește un model de date orientat pe documente, stocând informațiile sub formă de documente flexibile în format JSON sau BSON.

În schimb, bazele de date relaționale se bazează pe un model relațional, organizând datele în tabele cu rânduri și coloane și folosind chei străine pentru a defini relațiile între tabele.

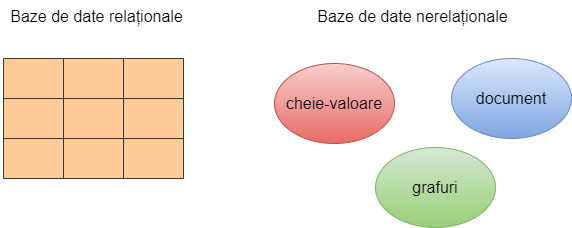


Figura 3 1 Diferențe dintre baza de date relațională și nerelațională

O altă diferență notabilă este flexibilitatea schemei. MongoDB permite o schemă flexibilă, unde documentele pot avea câmpuri diferite și structuri variabile, facilitând adaptabilitatea și modificările ulterioare ale structurii datelor. Pe de altă parte, bazele de date relaționale au o schemă rigidă, în care structura tabelului și tipurile de date trebuie definite în prealabil și respectate de fiecare înregistrare.

Limbajul de interogare reprezintă un alt aspect distinctiv. MongoDB utilizează un limbaj de interogare numit Query Language (QL), care oferă expresivitate și flexibilitate pentru interogări complexe și agregări de date. În schimb, bazele de date relaționale utilizează SQL (Structured Query Language) pentru interogări și manipularea datelor.

Scalabilitatea este și ea diferită între cele două tipuri de baze de date. MongoDB este proiectată pentru a permite scalabilitatea orizontală, adică gestionarea creșterii volumului de date prin adăugarea de noduri suplimentare într-un cluster. Pe de altă parte, bazele de date relaționale se scalează de obicei vertical, prin îmbunătățirea capacității hardware a unui singur server. [11]

### React JS

React JS este o bibliotecă open-source de JavaScript, dezvoltată inițial de către Facebook, care se concentrează pe crearea și gestionarea interfețelor utilizator (UI) interactive în aplicațiile web. Arhitectura React JS este construită în jurul conceptului de componente, adică unități individuale de cod care reprezintă elementele fundamentale ale unei interfețe. Aceste componente pot fi reutilizabile, modularizate și pot fi integrate într-o varietate de contexte de dezvoltare. [12]

Unul dintre principalele avantaje ale React JS este reprezentat de mecanismul său eficient de reconciliere virtuală (virtual DOM reconciliation). Acesta se bazează pe un arbore virtual de elemente DOM, care reprezintă o reprezentare abstractă a structurii interfeței utilizator. Atunci când starea unei componente se modifică, React JS folosește

algoritmi eficienți pentru a determina și actualiza doar porțiunile relevante ale arborelui virtual, evitând astfel actualizările costisitoare ale întregului DOM. Această abordare minimizează impactul asupra performanței aplicației și optimizează timpul de răspuns al interfeței utilizator.

De asemenea, React JS promovează un flux de date unidirecțional și o gestionare eficientă a stării. Componentele React pot primi proprietăți (props) de la componente părinte, iar starea internă a unei componente poate fi modificată doar prin intermediul unui mecanism specific de actualizare. Această abordare asigură o gestionare clară și predictibilă a datelor și permite dezvoltatorilor să creeze aplicații cu o arhitectură coerentă și ușor de înțeles.

Reutilizarea componentelor reprezintă o altă caracteristică esențială a React JS. Prin intermediul acestui mecanism, dezvoltatorii pot crea componente independente, generice și modulare, care pot fi utilizate în diferite contexte și aplicații. Această abordare promovează practici de dezvoltare eficiente, deoarece funcționalitățile comune pot fi implementate o singură dată și reutilizate în mai multe proiecte, reducând astfel duplicarea codului și accelerând ciclul de dezvoltare.

Pe lângă biblioteca de bază, React JS beneficiază de un ecosistem extins de biblioteci și instrumente suplimentare. Acestea adaugă funcționalități suplimentare și îmbunătățesc capabilitățile dezvoltării în React. De exemplu, React Router oferă un sistem robust de gestionare a rutelor în aplicațiile web de tip Single-Page (SPA), în timp ce Redux furnizează un mecanism de gestionare a stării aplicației pentru aplicații mai complexe.

În concluzie, React JS reprezintă o bibliotecă puternică și versatilă de JavaScript, care a câștigat o mare popularitate în dezvoltarea aplicațiilor web datorită abordării sale modulare, gestionării eficiente a stării și eficienței în actualizarea interfeței utilizator. Prin adoptarea conceptului de componente reutilizabile și printr-un ecosistem extins de biblioteci, React JS oferă dezvoltatorilor o platformă solidă pentru crearea de aplicații web moderne și interactive.

### Node.JS

Node.js [13] este un mediu de execuție open-source, construit pe platforma JavaScript, care permite dezvoltatorilor să ruleze codul JavaScript pe server. A fost creat cu scopul de a extinde capabilitățile JavaScript dincolo de mediu de browser și de a permite construirea de aplicații server-side scalabile și eficiente.

Node.js se bazează pe motorul JavaScript V8, dezvoltat de Google, care oferă o performanță remarcabilă în interpretarea și rularea codului JavaScript. Acesta utilizează un model de evenimente și I/O neliniar, cunoscut și sub numele de "non-blocking" sau "asynchronous", care permite gestionarea eficientă a unui mare număr de conexiuni simultane fără blocarea execuției.

Unul dintre principalele avantaje ale Node.js constă în faptul că utilizează același limbaj de programare pentru atât partea de client (browser), cât și partea de server. Aceasta permite dezvoltatorilor să împărtășească și să reutilizeze codul între aplicații client-side și server-side, reducând complexitatea dezvoltării și accelerând ciclul de dezvoltare.

Node.js este construit pe un model de module, care facilitează organizarea și reutilizarea codului. Există un ecosistem vast de module disponibile prin intermediul NPM (Node Package Manager), care oferă o gamă largă de funcționalități și biblioteci predefinite care pot fi utilizate în aplicații Node.js.

Prin intermediul Node.js, dezvoltatorii pot crea rapid și eficient aplicații web, aplicații de rețea, API-uri, microservicii și multe altele. Node.js oferă o abordare scalabilă și performantă pentru gestionarea cererilor multiple și a conexiunilor simultane, permițând construirea de aplicații în timp real și de mare viteză.

De asemenea, Node.js a devenit un mediu popular pentru dezvoltarea de aplicații bazate pe arhitectura serverless, unde codul este rulat în funcții individuale care sunt declanșate în funcție de evenimente specifice. Acest lucru permite scalabilitatea automată și gestionarea eficientă a resurselor în funcție de necesitățile aplicației.

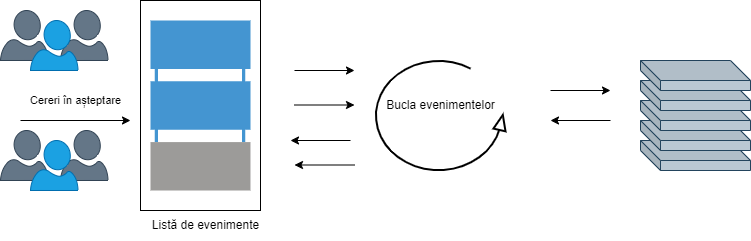


Figura 3 2 Arhitectura Node.js

Aceste tehnologii, în sinergie, permit dezvoltarea unei aplicații web moderne, eficiente și scalabile, cu o performanță excelentă și un timp de dezvoltare optimizat. Prin utilizarea React, dezvoltatorii pot construi interfețe utilizator atractive și interactive, utilizând componente reutilizabile. Node.js oferă un mediu de execuție performant pentru aplicații server-side, permițând gestionarea cererilor HTTP și conectarea la baze de date, iar integrarea cu MongoDB aduce flexibilitate și scalabilitate în gestionarea datelor. Utilizarea limbajului de programare JavaScript pe întreaga stivă tehnologică facilitează reutilizarea codului și împărțirea logică a aplicației între client și server.

# Proiectare

## Arhitectura aplicației

### Arhitectura client-server

Arhitectura unei aplicații web este o abordare structurală și organizatorică a dezvoltării software care implică împărțirea aplicației în componente interconectate, fiecare având un rol specific în furnizarea funcționalităților și a serviciilor. Acest model implică separarea funcționalității aplicației în două componente principale: clientul și serverul.

### Componenta de client

Partea de client este responsabilă pentru crearea și gestionarea interfeței utilizatorului. Aceasta este responsabilă de interacțiunea directă cu utilizatorul și de afișarea interfeței utilizator (UI). Această parte a fost implementată prin utilizarea framework-ului React facilitând construirea interfeței de utilizator prin construirea componentelor UI reutilizabile și reactive. Prin intermediul React, se pot defini stări ale componentelor și se pot reactualiza în mod eficient interfața utilizatorului în funcție de modificările stării.

Tot în componenta de client au fost folosite tehnologii precum HTML pentru structurarea conținutului și CSS pentru stilizarea și aspectul vizual al interfeței, precum și JavaScript pentru logica interactivă și manipularea evenimentelor. Partea de client este responsabilă și pentru trimiterea cererilor către server, de primirea și afișarea datelor obținute de la server și de gestionare a interacțiunii utilizatorului cu interfața. Aceasta implică validarea datelor de intrare, efectuarea acțiunilor solicitate de utilizator și prezentarea rezultatelor sau informațiilor relevante utilizatorului.

### Componenta de server

Serverul în arhitectura unei aplicații web reprezintă componenta responsabilă de gestionarea și furnizarea resurselor și serviciilor către clienți. Acesta primește cererile de la clienți și răspunde cu date și informații relevante. Este responsabil pentru prelucrarea și manipularea datelor, accesul la baza de date și furnizarea de funcționalități către utilizatori. Această parte a fost dezvoltată utilizând Node.js împreună cu framework-ul Express. Node.js fiind un mediu de execuție JavaScript, care permite rularea codului în afara browserului, pe server. Node.js beneficiază de asincronism și de un model de evenimente, ceea ce îl face potrivit pentru gestionarea unui număr mare de cereri și a procesării paralele a acestora.

Express permite dezvoltarea unei aplicații într-un mod structurat și eficient, permițând definirea rutelor și să implementarea logică de manipulare a cererilor într-un mod modular.

Baza de date, MongoDB, este o bază de date NoSQL care permite stocarea și gestionarea datelor într-un mod flexibil. În loc de tabele și înregistrări, aceasta stochează datele sub formă de documente JSON flexibile.

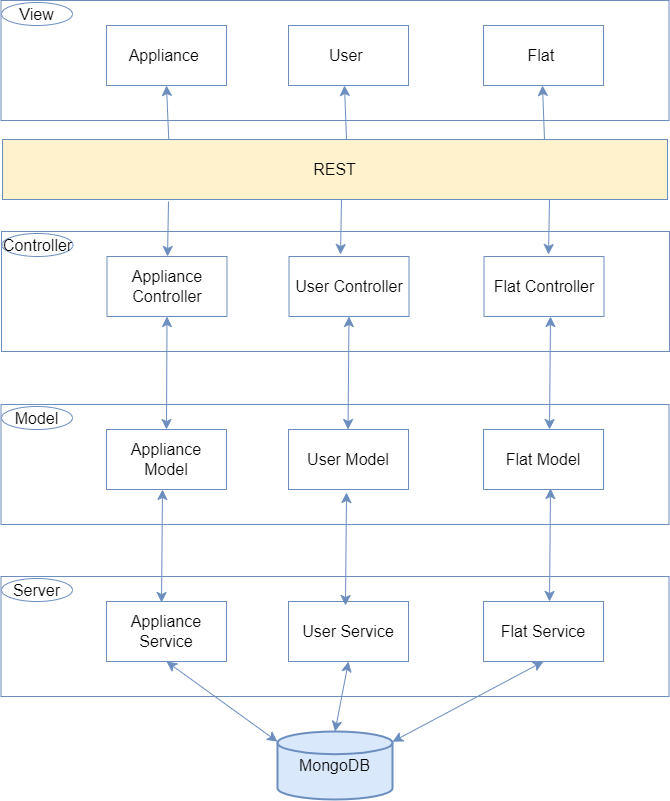


Figure 4-1 Arhitectura aplicației

### Comunicarea între client și server

Legatura dintre client (frontend) și server (backend) este realizată prin intermediul serviciilor web, prin folosirea protocoalelor HTTP, iar pentru transferul de date JSON. Clientul trimite cereri către server, iar acesta le procesează și trimite înapoi un răspuns.

Prin intermediul comunicării între client și server, se realizează schimbul de informații necesare pentru a actualiza și a afișa datele, asigurând astfel funcționarea corectă și sincronizată a aplicației.

## Detalii de proiectare

### Cazuri de utilizare

În cadrul acestei aplicații de monitorizare a consumului de curent electric al electrocasnicelor, sunt identificate și analizate mai multe cazuri de utilizare, care prezintă interacțiunea între utilizatori și aplicație. Aceste cazuri de utilizare evidențiază funcționalitățile și fluxurile de lucru principale ale aplicației, oferind o perspectivă asupra modului în care utilizatorii pot interacționa cu aceasta.

Actorul principal al aplicației este utilizatorul, deoarece majoritatea funcționalităților sunt construite pentru acesta.

Având în vedere funcționalitățile realizate, se identifică următoarele cazuri de utilizare destinate utilizatorilor.

* Înregistrare
* Autentificare
* Adăugare locuințe
* Editare locuințe
* Ștergere locuințe
* Adăugare electrocasnice
* Editare electrocasnice
* Ștergere electrocasnice
* Vizualizare locuințe și electrocasnicele asociate
* Vizualizare consum electric electrocasnice
* Vizualizare preț consum electrocasnice
* Generare grafice corespunzătoare prețului și consumului electrocasnicelor

Prin intermediul acestor funcționalități s-a dezvoltat o aplicație web destinată utilizatorilor interesați de monitorizarea consumului de energie electrică al electrocasnicelor.

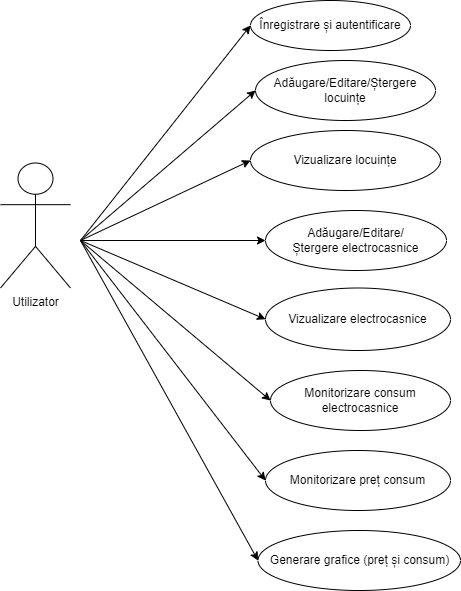


Figure 4‑2 Diagrama cazurilor de utilizare a utilizatorilor

Având în vedere cazurile de utilizare, se va detalia cel mai complex caz, adăugarea unui electrocasnic.

* Adăugarea unui electrocasnic

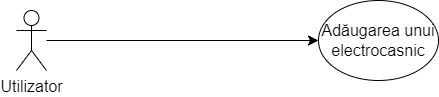


Figure 4‑3Diagrama cazului de adăugare a unui electrocasnic

Scenariu: Utilizatorul dorește să adauge un electrocasnic asociat unei locuințe

Flux de lucru: În momentul în care utilizatorul dorește să adauge un electrocasnic, acesta trebuie să completeze un formular care conține următoarele câmpuri:

* Numele electrocasnicului
* Puterea electrocasnicului în Watt (din cartea tehnică a electrocasnicului utilizatorul va introduce puterea acestuia)
* Selectarea datei de început și final al utilizării electrocasnicului
* Introducerea numărului de ore/zi al funcționării electrocasnicului
* Selectarea locuinței asociate electrocasnicului

După completarea acestor câmpuri, se va calcula automat prețul total al consumului electrocasnicului. Prețul unui kWh este prestabilit la 0.80 lei/kWh, după noile norme [14]. Prețul este calculat cu formula: pret\_total = (putere\* nr\_ore)/1000 \* nr\_zile \* 0.80. Prețul total va fi afișat în câmpul “price”, care mai apoi va fi salvat în baza de date.

Ulterior, cu toate aceste date vor fi generate atât graficele pentru preț pentru fiecare locuință cât și pentru consumul fiecare electrocasnic din locuințele asociate lor.

### Procesarea datelor și fluxul de comunicare al aplicației

Procesarea datelor este esențială pentru a gestiona informațiile și a le transforma într-o formă utilizabilă. Prin intermediul procesării datelor, componenta de server preia cererile primite de la componenta de client și le interpretează, extrăgând și manipulând datele necesare. Acest proces implică interacțiunea cu baza de date pentru a obține și actualiza informațiile relevante.

Importanța fluxului aplicației constă în faptul că definește modul în care utilizatorii interacționează cu aplicația și cum sunt gestionate cererile și răspunsurile între componenta de client și componenta de server. Un flux bine definit și eficient permite utilizatorilor să navigheze ușor prin aplicație, să acceseze și să actualizeze datele fără întârzieri semnificative și să beneficieze de o interfață receptivă și intuitivă.

Prin intermediul diagramelor secvențiale, se poate evidenția fluxul de date într-un mod vizual și ușor de înțeles. Acestea oferă o perspectivă clară asupra modului în care obiectele interacționează și cum se desfășoară procesele în cadrul aplicației. Astfel, diagrama secvențială ajută la identificarea și înțelegerea pașilor implicați în procesarea datelor și la evidențierea eventualelor probleme sau neclarități în fluxul de informații.

Prin utilizarea diagramelor secvențiale în cadrul unei aplicației de monitorizare a consumului de curent electric al electrocasnicelor, se poate evidenția modul în care datele de consum sunt colectate de la electrocasnice, cum sunt transmise la server pentru procesare, cum sunt stocate și cum pot fi accesate ulterior de către utilizatori. Această prezentare vizuală facilitează înțelegerea fluxului de date și contribuie la dezvoltarea unei aplicații bine structurate și eficiente.

Așadar, se va exemplifica printr-o diagramă secvențială fluxul de comunicare și procesarea datelor pentru crearea contului de utilizator.

Una dintre cele mai importante funcționalități ale acestei aplicații web este crearea unui cont de utilizator. Este necesar ca utilizatorul să fie înregistrat, deoarece acestă aplicație ține cont de anumite validări precum vârsta necesară (minim 18 ani), deoarece se consideră faptul că persoanele majore au interes mai aparte pentru monitorizarea consumului de curent și pentru faptul că acestea dețin locuințe sau locuiesc într-o chirie.

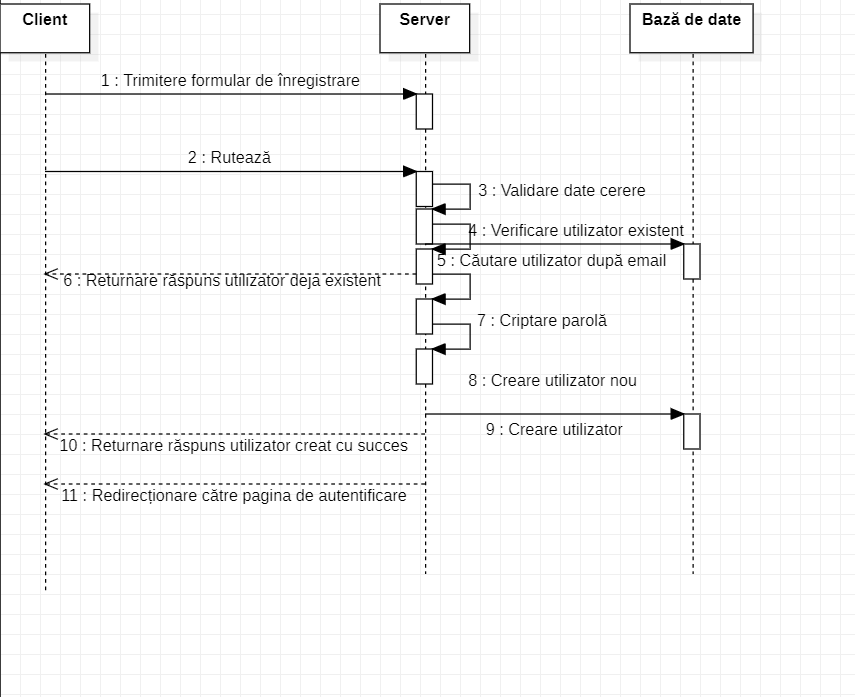


Figura 4-4 Diagrama secvențială a creării contului de utilizator

Utilizatorul completează formularul de înregistrare, furnizând detaliile necesare pentru crearea contului de utilizator, cum ar fi:numele, prenumele, adresa de email, parola, confirmarea parolei, vârsta, cnp și numărul de telefon. Adresa de email, parola, vârsta, cnp-ul și numărul de telefon au validări corespunzătoare acestora, fiind necesar ca utilizatorul să le respecte pentru crearea contului. În urma completării formularului, utilizatorul trimite o cerere HTTP de tip POST către ruta “/user/signup” de pe server.

La primirea cererii, serverul inițiază procesul de validare a datelor primite. În primul rând, verifică în baza de date dacă există deja un utilizator înregistrat cu adresa de email specificată în cerere. Aceasta implică o interogare a bazei de date pentru a căuta înregistrări care corespund adresei de email.

Serverul returnează un răspuns către client, comunicând dacă utilizatorul este deja înregistrat sau nu. Dacă se găsește un utilizator cu adresa de email respectivă în baza de date, se va trimite un răspuns negativ către client, indicând că utilizatorul este deja înregistrat și că trebuie să folosească o altă adresă de email.

În cazul în care utilizatorul nu este înregistrat, serverul continuă cu următoarea etapă a procesului. Serverul efectuează criptarea parolei primite de la client folosind un algoritm adecvat, cum ar fi bcrypt. Prin criptarea parolei, se asigură că aceasta este stocată într-o formă securizată în baza de date.

Apoi, serverul creează un nou utilizator în baza de date, folosind detaliile furnizate de client, inclusiv parola criptată. Aceasta implică crearea unei noi înregistrări în tabela utilizatorilor din baza de date, cu valorile corespunzătoare pentru fiecare câmp menționat anterior.

Serverul returnează un răspuns de succes către client, indicând că utilizatorul a fost creat cu succes.

La primirea răspunsului, clientul redirecționează utilizatorul către pagina de autentificare, unde acesta poate introduce adresa de email și parola pentru a accesa contul creat recent.

Această diagramă secvențială detaliată ilustrează fluxul de comunicare și interacțiunea între client, server și baza de date în procesul de creare a contului de utilizator, evidențiind rutele, acțiunile și schimbul de informații dintre aceste componente.

O altă funcționalitate importantă în cadrul acestei lucrări este autentificarea utilizatorului, fiind următorul pas după înregistrarea acestuia.

Acestă diagramă prezintă fluxul de informații și procesarea datelor în urma autentificării utilizatorului.

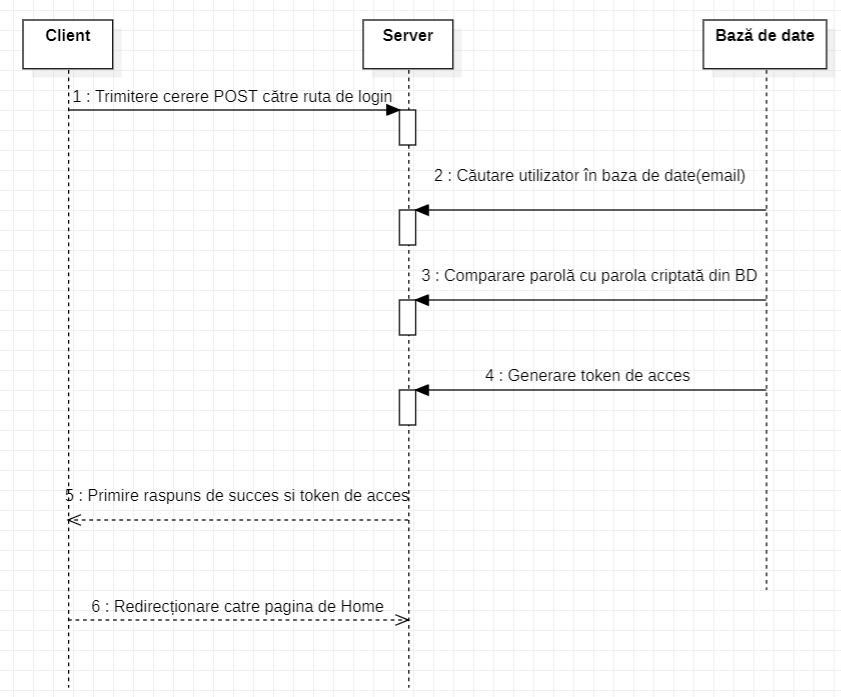


Figura 4-5 Diagrama secvențială a autentificării utilizatorului în aplicație

Această diagramă secvențială include legătura dintre client, server și baza de date în procesul de autentificare.

După ce clientul trimite o cerere POST către ruta /user/login de pe server, acesta primește cererea și începe procesul de autentificare.

Serverul se conectează la baza de date și caută utilizatorul cu adresa de email specificată în cerere.

Serverul compară parola trimisă de client cu parola criptată a utilizatorului din baza de date.

Dacă autentificarea reușește, serverul generează un token de acces folosind secretul ACCESS\_TOKEN\_SECRET și îl returnează împreună cu un răspuns de succes către client.

La primirea răspunsului de succes, clientul este redirecționat către pagina de Home.

Această diagramă secvențială evidențiază legătura între client, server și baza de date în procesul de autentificare și schimbul de informații între aceste componente.

### Interfața utilizatorului

Designul interfeței este un aspect important în dezvoltarea acestei aplicații deoarece implică crearea unei interfețe vizuale atractive, intuitiv de utilizat și eficientă din punct de vedere al experienței utilizatorului.

Aceasta are o structură clară și organizată, cu elementele plasate într-un mod coerent și ușor de înțeles. Setul de culori ales predominant (albastru și alb) împreună cu fontul și elementele grafice creează o experiență plăcută și o atmosferă adecvată aplicației. Această aplicație de monitorizare a consumului electric al electrocasnicelor trimite cu gândul implicit la culoarea de albastru, iar graficele generate sunt construite având culori contrastante pentru a facilita citirea și pentru asigurarea structurii logice.

În urma alegerii infrastructurii designului, s-a dezvoltat partea de User Interface.

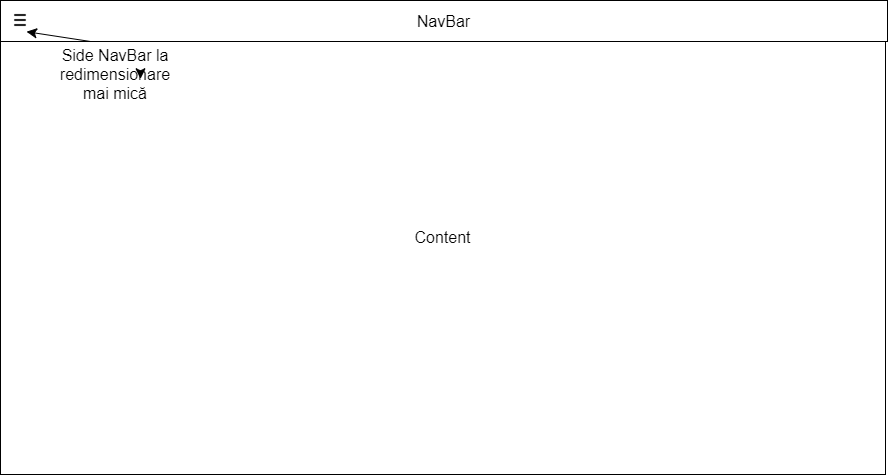


Figure 4-6 Infrastructura interfeței

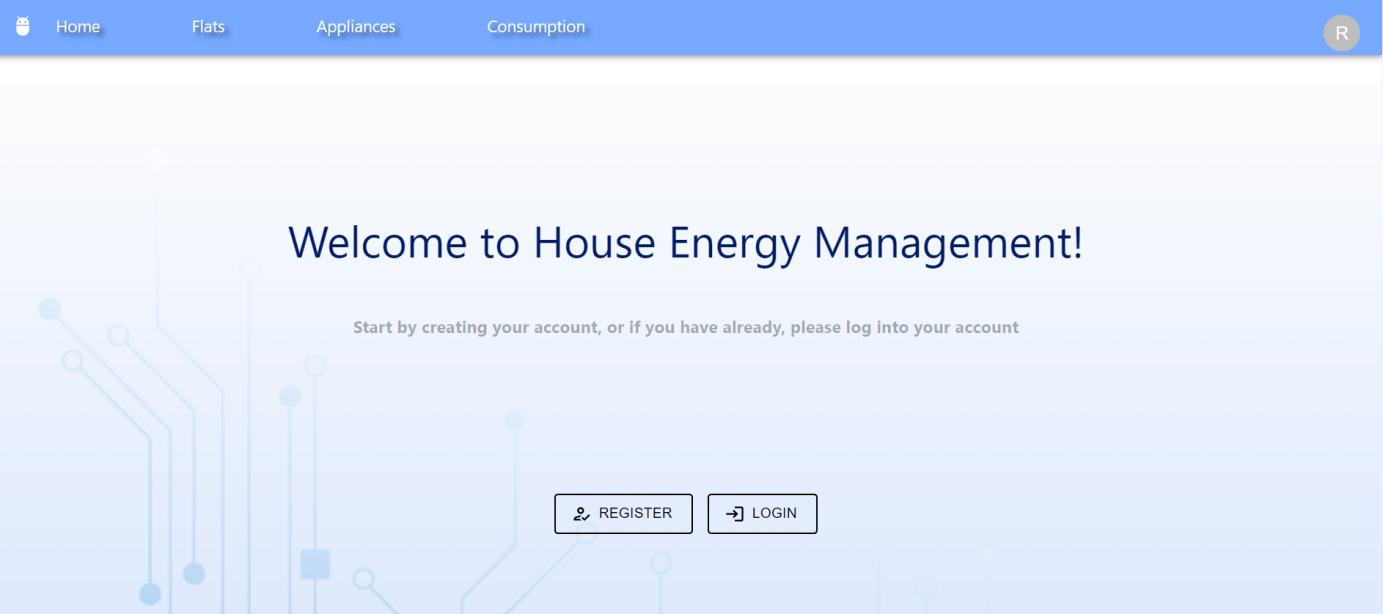


Figura 4-7 Pagina de Dashboard

Pentru crearea interfeței se observă faptul că aceste culorile asociate oferă utilizatorului un mediu prietenos dar în același timp denotă și un mediu de seriozitate. Primul lucru pe care utilizatorul trebuie sa îl facă este să se înregistreze sau să se autentifice pentru a putea dispune de funcționalitățile acestei aplicații de monitorizare a consumul de curent electric al electrocasnicelor asociate locuințelor.

# Implementare

Prin această etapă în care conceptele abstracte devin realitate și ideile se transformă în cod sursă, aplicația de monitorizare a consumului de energie electrică a electrocasnicelor s-a dezvoltat fiind structurată în trei părți:

* Baza de date (MongoDB)
* Backend (Node.js + Express)
* Frontend (React)

Baza de date a fost dezvoltată folosind MongoDB, deoarece este o soluție puternică și flexibilă pentru stocarea și gestionarea datelor în aplicații web, acesta fiind o baza de date nerelațională care folosește obiecte de tip JSON.

Implementarea backendului cu Node.js și frameworkul Express aduce flexibilitate, performanță și o abordare ușoară a dezvoltării aplicațiilor web. Aceste tehnologii permit dezvoltarea unei aplicații scalabile, eficiente și ușor de întreținut.

În dezvoltarea frontendului s-a ales frameworkul JavaScript, React, care este un framework modern. Folosirea acestuia are multe avantaje precum reutilizarea componentelor, gestionarea stării și compatibilitatea cu tehnologiile alese pentru backend.

Arhitectura REST stă la baza acestei aplicații, fiind necesară dezvoltarea eficientă și coerentă a aplicației. Acesta oferă avantaje precum interoperabilitate, scalabilitatea, separarea clară între frontend și backend, precum și performanță și eficiență.

## Baza de date (MongoDB)

### Alegerea sistemului de gestionare a bazei de date

Alegerea sistemului de gestionare a bazelor de date este un pas crucial în implementarea aplicației. În cazul specific al acestei aplicației am optat pentru MongoDB ca sistem de gestionare a bazelor de date. Am luat această decizie în urma evaluării cerințelor proiectului și a analizei caracteristicilor oferite de MongoDB. structura flexibilă de tip document, scalabilitatea orizontală, suportul pentru replicare și toleranță la defecțiuni, capacitatea avansată de indexare și funcționalitățile puternice de agregare și interogare au fost factori determinanți în alegerea MongoDB.

De asemenea, am ținut cont de compatibilitatea și integrarea cu celelalte tehnologii utilizate în proiect. Prin alegerea MongoDB, am asigurat un sistem robust și eficient pentru stocarea și manipularea datelor.

### Implementarea și proiectarea structurii bazei de date

Baza de date a acestei aplicații, MongoDB, este o baza de date nerelațională care oferă o structură flexibilă și scalabilitate orizontală. Acesta implică definirea a trei colecții pentru diferite entități: “appliances”,”flats” și “users”. Entitățile fiecărei colecții sunt transformate sub formă de JSON, iar fiecare entitate este generată printr-un identificator unic ușurând accesarea lor.

Baza de date MongoDB permite, de asemenea, gestionarea relațiilor între entități prin includerea referințelor între documente. Astfel, un document al unei entități poate avea un câmp care face referire la un alt document dintr-o altă colecție.

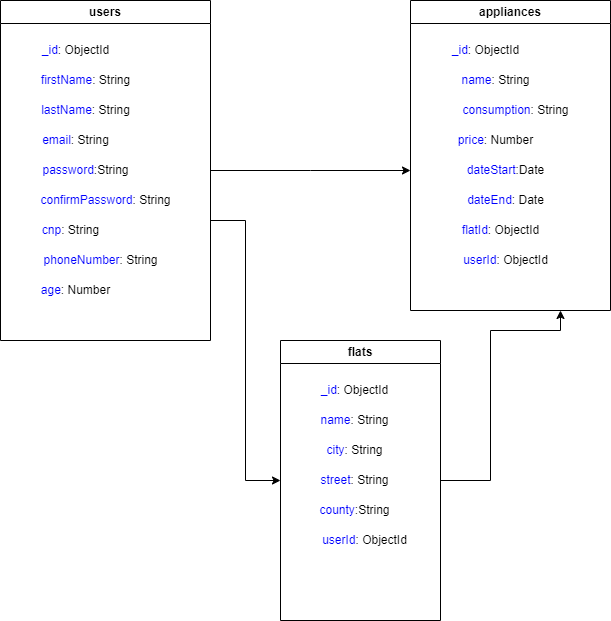


Figura 5-1 Diagrama bazei de date

În figura 5-1, s-a prezentat diagrama bazei de date. În cadrul acesteia se poate observa comunicarea dintre colecțiile appliances, flats și users. Conexiunea dintre users și flats este reprezentată prin câmpul de \_id din users și userId din flats, iar legătura dintre flats și appliances este dată to de \_id din flats și flatId din appliances. Asemenea este reprezentată și legătura dintre users și appliances.

#### **Colecția users**

Acestă colecție din baza de date este folosită pentru stocarea informațiilor utilizatorilor în momentul în care aceștia se înregistrează în aplicație. Colecția dispune de următoarele câmpuri:

* firstName:
  + Reprezintă numele de familie al utilizatorului
  + Câmpul este de tip String
  + Completarea lui este obligatorie
* lastName:
  + Reprezintă prenumele utilizatorului
  + Câmpul este de tip String
  + Completarea lui este obligatorie
* email:
  + Emailul trebuie să fie unic
  + Câmpul este de tip String
  + Completarea lui este obligatorie
* password:
  + Reprezintă parola aleasă de către utilizator
  + Câmpul este de tip String
  + Completarea lui este obligatorie
* confirmPassword:
  + Utilizatorul trebuie să reintroducă parola pentru a fi validată ulterior
  + Câmpul este de tip String
  + Completarea lui este obligatorie
* cnp:
  + Reprezintă CNP-ul utilizatorului
  + Câmpul este de tip String
  + Completarea lui este obligatorie
* phoneNumber:
  + Reprezintă numărul de telefon al utilizatorului
  + Câmpul este de tip String
  + Completarea lui este obligatorie
* age:
  + Reprezintă vârsta utilizatorului
  + Câmpul este de tip Number
  + Completarea lui este obligatorie

#### **Colecția appliances**

Acestă colecție din baza de date este folosită pentru stocarea informațiilor electrocasnicelor în momentul în care utilizatorul deja înregistrat dorește să introducă un electrocasnic asociat locuinței lui.

* name:
  + Reprezintă numele electrocasnicului
  + Câmpul este de tip String
  + Completarea lui este obligatorie
* consumption:
  + Reprezintă puterea electrică a electrocasnicului
  + Câmpul este de tip String
  + Completarea lui este obligatorie
* price:
  + Câmpul este de tip Number
  + Completarea lui este obligatorie
* dateStart:
  + Reprezintă data de început a utilizării electrocasnicului aleasă de către utilizator
  + Câmpul este de tip Date
  + Completarea lui este obligatorie
* dateEnd:
  + Reprezintă data de sfârșit a utilizării electrocasnicului aleasă de către utilizator
  + Câmpul este de tip Date
  + Completarea lui este obligatorie
* flatId:
  + Reprezintă referința către colecția “flats”
  + Câmpul este de tip ObjectId
  + Completarea lui este obligatorie
* userId:
  + Reprezintă referința către colecția “users”
  + Câmpul este de tip ObjectId
  + Completarea lui este obligatorie

#### **Colecția flats**

Acestă colecție din baza de date este folosită pentru stocarea informațiilor locuințelor utilizatorului deja înregistrat.

* name:
  + Reprezintă numele locuinței (ex: Apartament 56)
  + Câmpul este de tip String
  + Completarea lui este obligatorie
* city:
  + Reprezintă numele orașului în care se află locuința
  + Câmpul este de tip String
  + Completarea lui este obligatorie
* street:
  + Reprezintă detaliile străzii locuinței utilizatorului
  + Câmpul este de tip String
  + Completarea lui este obligatorie
* county:
  + Reprezintă numele județului
  + Câmpul este de tip String
  + Completarea lui este obligatorie
* userId:
  + Reprezintă referința către colecția “users”
  + Câmpul este de tip ObjectId
  + Completarea lui este obligatorie

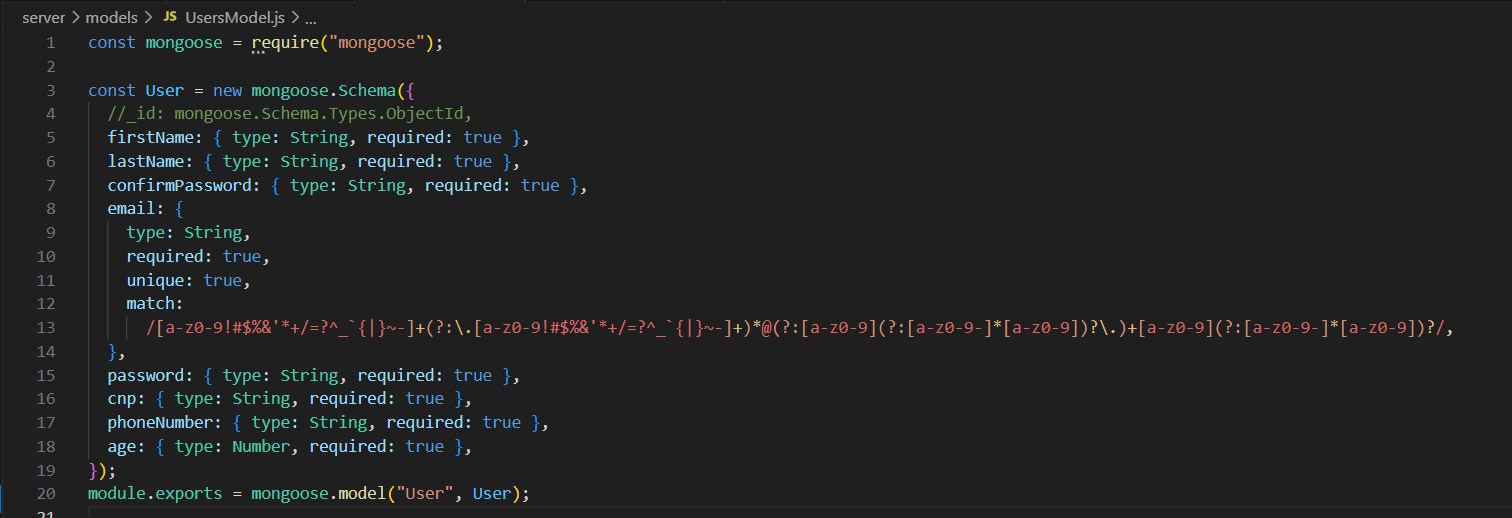


Figure 5-2 Crearea colecției User

## Backend

Structurarea modulului de backend este esențială pentru dezvoltarea eficientă și întreținerea ușoară a unei aplicații. O structură bine definită facilitează înțelegerea codului și permite reutilizarea componentelor în diferite părți ale aplicației sau în alte proiecte.

### Structura de fișiere și directoare

Backendul este motorul aplicației și este responsabil cu procesarea datelor și comunicarea cu baza de date MongoDB.

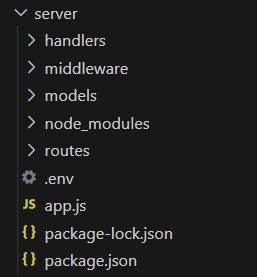


Figure 5-3 Structura Backend

#### **Directorul handlers**

Directorul "handlers" conține fișierele în care este implementată logica aplicației pentru fiecare colecție din baza de date. Fiecare fișier reprezintă un handler pentru o anumită colecție și conține funcții care gestionează cererile și răspunsurile asociate acelei colecții. Aceste funcții includ operațiile CRUD (Create, Read, Update, Delete), precum și alte operații specifice pentru manipularea și procesarea datelor din colecție.

Prin separarea logicii în fișiere separate, se obține o organizare mai clară a codului și o gestionare mai eficientă a funcționalităților specifice fiecărei colecții. Această structură permite o dezvoltare modulară și ușor de întreținut a backend-ului aplicației.

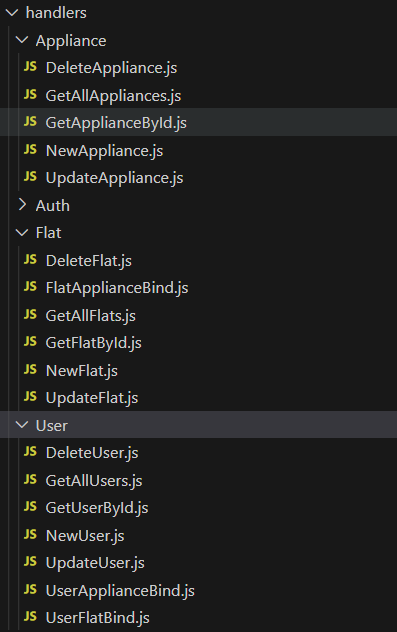


Figure 5-4 Directorul “handlers”

#### **Directorul middleware**

Acest director conține o funcție middleware numită validateToken care este utilizată pentru validarea și autentificarea tokenului JWT (JSON Web Token).

Funcția primește trei parametri: req (obiectul cerere), res (obiectul răspuns) și next (funcția middleware următoare în lanț).

Dacă tokenul este valid și decodificarea sa este reușită, obiectul decodat conține informațiile utilizatorului autentificat. Aceste informații sunt atașate obiectului cerere (req.user) pentru a fi utilizate în funcționalitățile ulterioare.

În final, se apelează funcția next() pentru a permite trecerea la următorul middleware în lanț. Dacă tokenul lipsește sau este invalid, se aruncă o eroare și se returnează un răspuns de eroare corespunzător.Acest middleware este utilizat în rutele care necesită autentificare, asigurând că doar utilizatorii autorizați pot accesa aceste resurse.

#### **Directorul models**

În cadrul acestui director, fiecare fișier reprezintă un model de date specific, de exemplu, FlatModel.js. În interiorul fișierului prin utilizarea frameworkului Mongoose se definesc schemele și comportamentul modelelor. Schema modelelor definește câmpurile și tipurile acestora, restricțiile, validările și orice alte proprietăți specifice.



Figure 5-5 Directorul “models”

#### **Directorul routes**

Directorul routes conține fișierele care gestionează rutele și rutele API ale aplicației backend. Aceste fișiere definesc comportamentul și logica pentru fiecare rută disponibilă în aplicație.

În cadrul acestui director, fiecare fișier reprezintă un grup de rute legate de o anumită entitate a aplicației. De exemplu, aveam un fișier numit flats-routes.js care gestionează rutele legate de operațiile CRUD (Create, Read, Update, Delete) pentru locuințe.

În fișierele din directorul routes, se utilizează framework-ul Express pentru a defini rutele și a le asocia cu funcțiile corespunzătoare. Aceste funcții sunt responsabile de preluarea cererilor HTTP, apelarea logicii necesare (cum ar fi interogarea bazei de date sau validarea datelor) și generarea răspunsurilor HTTP corespunzătoare.

#### **Fișierul app.js**

Acesta este fișierul principal al acestei aplicații deoarece este responsabil de configurarea și pornirea serverului utilizând Express. Aici sunt adăugate și configurate toate componentele necesare pentru funcționarea corectă a serverului și pentru gestionarea cererilor primite de la client, precum și conectarea la baza de date MongoDB.

#### **Fișierul .env**

În cadrul aplicației, fișierul .env conține un token de acces. Un access token este un identificator criptografic utilizat pentru a autentifica și autoriza cererile între client și server.

### Autentificarea și autorizarea utilizatorilor

Se utilizează modulele precum "express", "mongoose", "bcrypt" și "jsonwebtoken" pentru gestionarea cererilor HTTP, comunicarea cu baza de date MongoDB, criptarea parolelor și generarea token-urilor de autentificare.

Prin intermediul funcției "User.findOne()", se verifică dacă utilizatorul există deja în baza de date. Parola utilizatorului este criptată utilizând funcția "bcrypt.hash()" înainte de a fi salvată în baza de date. În cele di urmă se compară parola introdusă cu cea stocată în baza de date utilizând bcypt.compare().

În caz de succes, se generează un token JWT pentru autorizare.

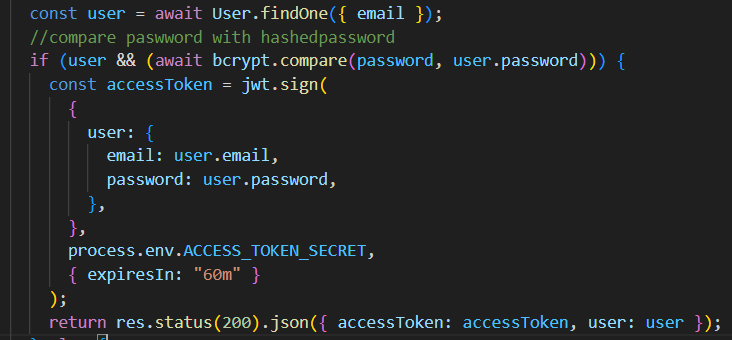


Figura 5-6 Criptare parolă și generare de token

## Frontend

În cadrul implementării frontend-ului aplicației, s-a ales utilizarea framework-ului React, datorită avantajelor pe care le oferă în dezvoltarea interfeței utilizatorului.

### Structura de fișiere și directoare

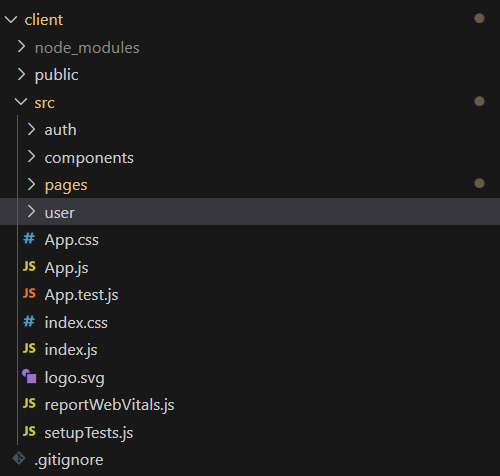


Figura 5-7 Structura Frontend

#### **Directorul public**

În directorul public se regăsesc imaginile care sunt folosite pentru fundalul aplicației. Atunci când se construiește aplicația, conținutul directorului "public" este copiat în directorul de ieșire al aplicației finale, astfel încât aceste fișiere să fie disponibile publicului în timpul rulării aplicației. De exemplu, avem o imagine numită "fundal.jpg" și putem accesa această imagine în codul CSS prin specificarea căii relative către aceasta, fără a fi necesară o rută specifică a serverului.

Fișierele din directorul "public" sunt accesibile publicului și pot fi accesate direct prin URL-uri.

#### **Directorul src**

Directorul "src" este un punct de plecare pentru găsirea și navigarea prin codul sursă al aplicației, iar modificările și extinderile vor avea loc, în principal, în acest director și în subdirectoarele sale.

Acesta este directorul principal în care se găsesc toate fișierele sursă ale aplicației.

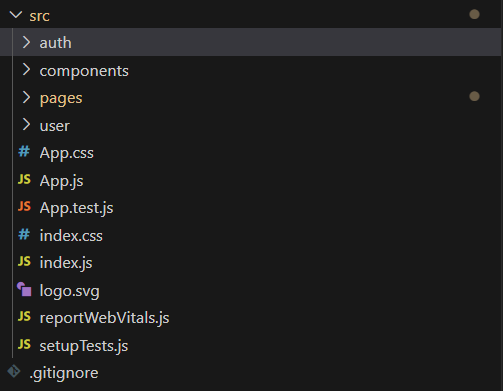


Figura 5-7 Directorul “src”

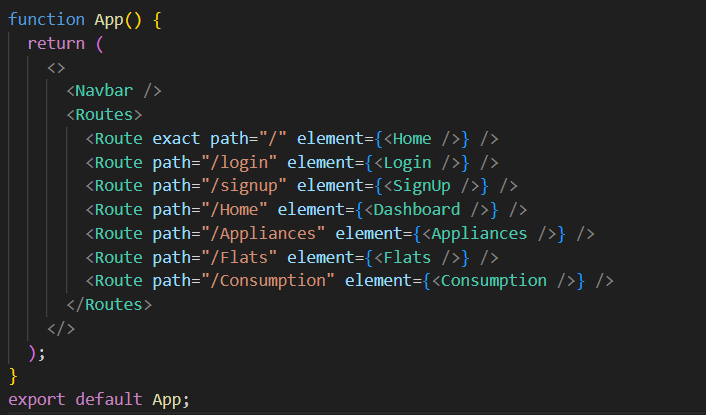
* components: conține componenta de Navbar
* pages:
  + Home
  + Dashboard
  + Flats
  + Appliances
  + Consumption
* App.js: Acesta definește componenta principală a aplicației și rutele asociate.
* Index.js: configurează și pornește aplicația React, rendează componenta principală "App" în cadrul unui "BrowserRouter" și utilizând instanța "root" pentru rendare în elementul HTML cu ID-ul "root".

Figura 5-1 Fișierul App.js

# Concluzii

## Rezultate obținute

Evidențiați toate rezultatele pe care le-ați obținut și trageți concluzii din ele. Puteți prezenta o analiză critică a ceea ce ați realizat comparativ cu alte lucrări/studii anterioare.

Includeți o listă a contribuțiilor pe care le-ați avut în domeniul temei abordate.

## Direcții de dezvoltare

Descrieți direcțiile posibile de dezvoltare.

# Reguli de formatare

## Formatarea paginii

* + Dimensiunea paginii: A4
  + Margini: 2.5 cm (sus, jos, stânga, dreapta)
  + Antet și subsol: 1.27 cm de la marginea paginii
  + În antetul paginii (header): titlul capitolului, centrat, stil: Header\_style
  + În subsolul paginii: numărul paginii, centrat

## Titluri și stiluri

Titlurile capitolelor și subcapitolelor se marchează cu stilurile Heading 1 – 4, conform documentului model anexat în format Word. Descrierea stilurilor utilizate în document este prezentată în Tabelul 5.1.

Tabelul 5.1. Stiluri utilizate în acest document

| Nr. | Stil | Utilizat pentru | Format |
| --- | --- | --- | --- |
| 1 | Normal | Text normal | Font: (Default) Cambria, 12 pt, Justified, Line spacing: Multiple 1.1 li, Space After: 6 pt |
| 2 | Titlu | Titlul proiectului, prima pagină | Font: 24 pt, Small caps, Centered Line spacing: single, Space Before: 126pt, After: 0 pt, |
| 3 | Titlu2 | Titlul proiectului, pagina de prezentare | Font:14pt, Bold, Centered |
| 4 | Heading 1 | Titlurile capitolelor (nivel 1) | Font: 24 pt, Indent: Left: 0 cm Hanging: 0.76 cm, Space Before: 24pt, After: 12pt |
| 5 | Heading 2 | Titlurile subcapitolelor (nivel 2) | Font: 14 pt, Bold, Indent: Left: 0 cm  Hanging: 1.02 cm, Space Before: 18pt, After: 12pt |
| 6 | Heading 3 | Titlurile secțiunilor (nivel 3) | Font: Bold, Indent: Left: 0 cm Hanging: 1.27 cm, Space Before: 6 pt, After: 6pt |
| 7 | Heading 4 | Titlurile secțiunilor (nivel 4) | Font: Italic, Indent: Left: 0 cm Hanging: 1.52 cm, Space Before: 2 pt, After: 0 pt |
| 8 | Caption | Legenda figurilor și tabelelor | Font: Italic, Font color: Text 1, Line spacing: single, Space After: 10 pt, |
| 9 | Header\_style | Antetul paginii | Font: 10 pt, Italic, Centered, Border: Bottom: (Single solid line, Background 1, 0.5 pt Line width) |

## Figuri, tabele și ecuații

### Figuri

Figurile se inserează în text centrate, cu etichetă de numerotare și legendă (Caption) în partea de jos a figurii. Numărul figurii include și numărul capitolului, după exemplul prezentat în Figura 5.1.



Figura 5‑1. Figură exemplu, stil: Caption

## Tabele

Tabelele se inserează în text centrate, cu etichetă și legendă (Caption) în partea de sus a tabelului, aliniată la stânga. Numărul tabelului include și numărul capitolului, după cum este prezentat, de exemplu, în Tabelul 5.1.

## Ecuații

Ecuațiile se inserează în text centrate, cu numerotare în partea dreaptă. Numărul ecuației include și numărul capitolului, conform exemplului din relația (5.1).

|  |  |
| --- | --- |
|  | (5.1) |

## Referințe bibliografice

Se recomandă ca citarea referințelor bibliografice să fie făcută în formatul IEEE.

În secțiunea Bibliografie sunt prezentate exemple pentru: o citare a unui capitol dintr-o carte [12], un articol publicat într-o revistă [13] și un articol publicat la o conferință [14].

Detalii cu privire la formatul citării diverselor tipuri de referințe pot fi găsite în [15] sau [16].

Referințele bibliografice se pot insera în text utilizând facilitățile Word de a adăuga surse și bibliografie unui document (References -> Citations & Bibliography). Dacă formatul IEEE pentru bibliografie nu este instalat implicit în Word, se poate descărca gratuit de la:

<https://bibword.codeplex.com/wikipage?title=Styles&referringTitle=Home>

Instrucțiunile de instalare pentru diferite versiuni de Word se pot obține de la aceeași adresă.

# Bibliografie

|  |  |
| --- | --- |
| [1] | M. V. Dincă, M. Busu și Z. Nagy-Bege, „Determinants with Impact on Romanian Consumers’”. |
| [2] | A. Maxim, „Explaining the behavior of Romanian household electricity”. |
| [3] | I. E. A. (IEA), „Energy Efficiency 2021: Analysis and Outlooks to 2040,” 2021. |
| [4] | H.M. Røpke, M.J. Howarth și E. Heiskanen, „Making energy visible: A qualitative field study of how householders interact with feedback from smart energy monitors”. |
| [5] | [Interactiv]. Available: https://treewebsolutions.com/ro/articole/bazele-rest-api-68. |
| [6] | Bruno Costa, Paulo F. Pires, Flavia C. Delicato și Paulo Merson , „Evaluating REST architectures—Approach, tooling and guidelines”. |
| [7] | „https://forumautomation.com/t/5-best-electricity-cost-and-electricity-consumption-calculator-apps-for-android-and-their-features/8623,” [Interactiv]. |
| [8] | „https://ro.education-wiki.com/3958588-what-is-visual-studio-code,” [Interactiv]. |
| [9] | [Interactiv]. Available: https://code.visualstudio.com/. |
| [10] | [Interactiv]. Available: https://javascript.info/. |
| [11] | [Interactiv]. Available: https://www.mongodb.com/compare/relational-vs-non-relational-databases. |
| [12] | P. Nume, „Titlul capitolului,” în *Titlul cartii*, Oras, Editura, 2016, pp. 1-24. |
| [13] | P. Nume, „Titlul articolului,” *Titlul revistei,* vol. 1, nr. 2, pp. 22-30, 2016. |
| [14] | P. Nume, „Titlul articolului,” în *Numele conferintei*, Oras, 2015. |
| [15] | „IEEE Citation Reference,” 2009. [Interactiv]. Available: https://www.ieee.org/documents/ieeecitationref.pdf. |
| [16] | „IEEE Editorial Style Manual,” 2016. [Interactiv]. Available: https://www.ieee.org/documents/style\_manual.pdf. |
| [17] | „"Making energy visible: A qualitative field study of how householders interact with feedback from smart energy monitors”. |