**UNIVERSITATEA NAȚIONALĂ DE ȘTIINȚĂ ȘI TEHNOLOGIE POLITEHNICA BUCUREȘTI CENTRUL UNIVERSITAR PITEȘTI**

**FACULTATEA DE ELECTRONICĂ, COMUNICAȚII ȘI CALCULATOARE**

**DEPARTAMENTUL ELECTRONICĂ, CALCULATOARE ȘI INGINERIE ELECTRICĂ**

**PROGRAMUL DE STUDII UNIVERSITARE DE LICENȚĂ**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Absolvent**

**DINICA F. Madalin-Alexandru**

**Conducător științific**

**conf.univ.dr.ing. Ene Alexandru**

**Pitești**

**Sesiunea iulie 2024**

**UNIVERSITATEA NAȚIONALĂ DE ȘTIINȚĂ ȘI TEHNOLOGIE POLITEHNICA BUCUREȘTI CENTRUL UNIVERSITAR PITEȘTI**

**FACULTATEA DE ELECTRONICĂ, COMUNICAȚII ȘI CALCULATOARE**

**DEPARTAMENTUL ELECTRONICĂ, CALCULATOARE ȘI INGINERIE ELECTRICĂ**

**PROGRAMUL DE STUDII UNIVERSITARE DE LICENȚĂ**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PROIECT DE DIPLOMĂ**

APLICATIE WEB PENTRU TESTAREA CUNOSTINELEOR IN LIMBAJUL C

**Absolvent**

**DINICA F. Mădălin-Alexandru**

**Conduc****ător științific**

**conf.univ.dr.ing. Ene Alexandru**

POZA DOCUMENT CU TEMA PROIECTULUI DE DIPLOMA

POZA DECLARATIE PRIVIND ORIGINALITATEA PROIECTULUI DE DIPLAMA

CUPRINS

TABEL DE FIGURI

LISTA ACRONIMELOR

# 1. Introducere

## 1.1 Motivația alegerii temei

Alegerea acestei teme a fost determinată de mai mulți factori importanți care reflectă nevoile actuale din domeniul educației în programare. În primul rând, după o analiză detaliată a peisajului actual al platformelor educaționale online, am observat că există relativ puține resurse dedicate exclusiv testării cunoștințelor în limbajul de programare C. Deși limbajul C rămâne fundamental în educația informatică și continuă să fie folosit în numeroase domenii - de la programarea sistemelor la dezvoltarea embedded - oferta de platforme interactive pentru exercitarea și evaluarea competențelor în acest limbaj este surprinzător de limitată.

Platformele existente care oferă acest tip de conținut prezintă o problemă semnificativă: majoritatea funcționează pe un model ,,freemium’’, oferind doar un trial limitat sau un număr restrâns de exerciții gratuite, după care trecerea la o versiune plătită devine necesară. Această abordare creează bariere în calea studenților și a programatorilor începători care doresc să-și dezvolte competențele, dar nu dispun de resurse financiare pentru abonamente costisitoare. Lipsa accesului la resurse de calitate și gratuite poate să îngreuneze procesul de învățare și să descurajeze persoanele motivate să-și dezvolte abilitățile de programare.

Un alt aspect care m-a determinat să aleg această temă este dorința de a contribui la formarea următoarei generații de programatori. Observ că mulți tineri dezvoltatori se confruntă cu ceea ce comunitatea programatorilor numește "tutorial-hell" - fenomenul prin care aspiranții programatori consumă o cantitate enormă de conținut educațional (tutorial video, cursuri online, documentații) fără să aplice în mod practic ceea ce au învățat. Aceștia rămân blocați într-un ciclu neproductiv de consum pasiv de informații, fără să-și dezvolte competențele practice necesare pentru a deveni programatori eficienți[1.2].

Această problemă este deosebit de vizibilă în cazul limbajului C, care necesită o înțelegere profundă a conceptelor fundamentale și multă practică pentru a fi stăpânit. Mulți studenți citesc despre pointeri, managementul memoriei sau structuri de date, dar nu au suficiente oportunități de a-și testa cunoștințele într-un mediu structurat și progresiv.

## 1.2 Obiectivele generale ale proiectului

Obiectivul principal al acestui proiect este dezvoltarea unei platforme web interactive și accesibile care să permită utilizatorilor să-și testeze și să-și îmbunătățească cunoștințele de programare în limbajul C. Platforma își propune să ofere o alternativă gratuită și de calitate la soluțiile comerciale existente, eliminând barierele financiare care pot împiedica accesul la educație de calitate.

Un obiectiv secundar, dar la fel de important, este combaterea fenomenului de ,,tutorial hell’’ prin oferirea unei experiențe de învățare activă și aplicativă. În loc să se bazeze doar pe consum pasiv de conținut, utilizatorii vor fi încurajați să scrie cod, să rezolve probleme concrete și să primească feedback imediat asupra soluțiilor lor. Aceasta facilitează tranziția de la cunoașterea teoretică la competențele practice, esențială pentru orice programator.

Platforma urmărește să acopere întregul spectru de dificultate, de la concepte de bază pentru începători până la probleme avansate pentru programatorii mai experimentați. Aceasta include exerciții pentru sintaxa de bază, structuri de control, funcții, pointeri, structuri de date și algoritmi fundamentali.

Un alt obiectiv important este crearea unei experiențe de utilizare plăcute și motivante prin implementarea unor elemente de „gamification’’. Utilizatorii vor putea să-și urmărească progresul, si să acceseze statistici despre performanța lor.

## 1.3 Contribuția studentului și rezultatele obținute

Contribuția personală la acest proiect constă în întregul proces de dezvoltare, de la analiza cerințelor până la implementarea finală și testarea aplicației. Platforma "**CodeMaster**" a fost construită folosind un pachet tehnologic modern și robust, care include **Next.js** pentru framework-ul principal, **React** pentru componentele de interfață, **TailwindCSS** pentru stilizare, **Firebase** pentru gestionarea utilizatorilor și stocarea problemelor, **Judge0** pentru compilarea și execuția codului, și integrarea unui asistent virtual, **VoiceGlow**, pentru asistența utilizatorilor.

## 1.4 Rezultate și impact

Rezultatele obținute până în prezent demonstrează viabilitatea și eficiența soluției dezvoltate. Platforma oferă o experiență fluida de la autentificare până la rezolvarea problemelor, cu un timp de răspuns rapid și o interfață intuitivă. Integrarea cu Judge0 asigură compilarea și execuția sigură a codului, în timp ce arhitectura bazată pe Firebase garantează scalabilitatea și fiabilitatea sistemului.

Prin dezvoltarea acestei platforme, am reușit să creez o soluție concretă pentru problemele identificate în ecosistemul actual de învățare online, oferind o alternativă gratuită și accesibilă pentru toți programatorii care doresc să-și dezvolte competențele în limbajul C.

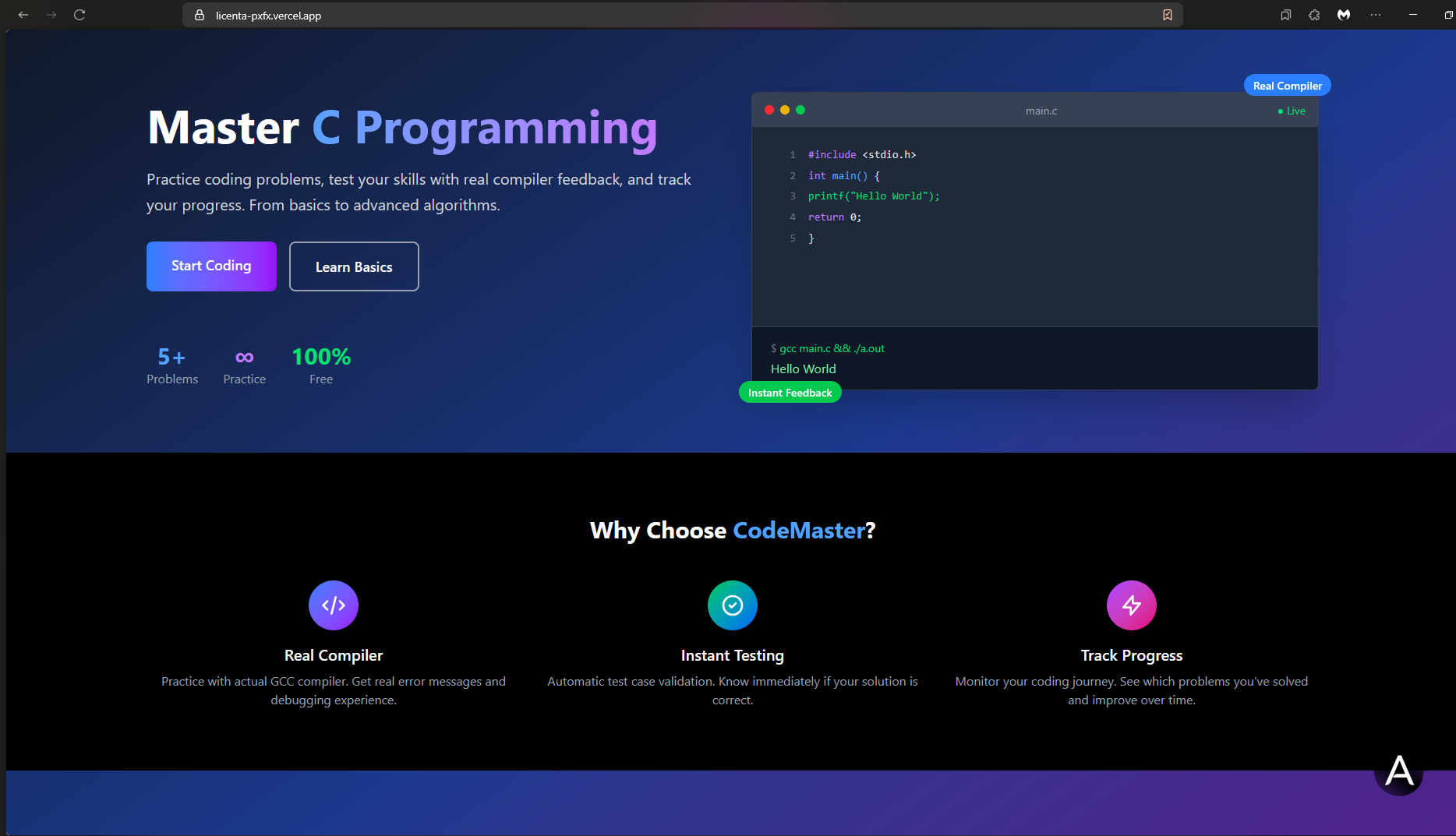


Figura 1.0‑1: Pagina Principala

# 2. Stadiul Actual

## 2.1 Analiza platformelor existente de învățare online

În contextul în care industria tehnologiei înregistrează o creștere explozivă și nevoia de programatori competenți devine din ce în ce mai acută, piața platformelor de învățare online în programare a cunoscut o dezvoltare semnificativă. Pentru a înțelege poziționarea și potențialul aplicației dezvoltate, este esențială analiza detaliată a principalelor platforme existente, a limitărilor acestora și a oportunităților pe care le oferă.

## 2.2 Codecademy – Liderul în educația interactivă

Conform site-ului oficial Codecademy [1.3], fondată în august 2011 de Zach Sims și Ryan Bubinski în New York City, ridicând 2,5 milioane de dolari în finanțare Seria A și ulterior 10 milioane în Seria B, s-a poziționat ca una dintre cele mai populare platforme de învățare a programării [1.4]. Platforma oferă cursuri gratuite de programare în 13 limbaje, inclusiv Python, Java, Go, JavaScript, Ruby, SQL, C++, C#, Lua, Swift, HTML și CSS [1.3].

Codecademy funcționează pe un model freemium, cu acces limitat gratuit și opțiuni avansate disponibile prin abonamentul Codecademy Pro. Acesta oferă căi de carieră, proiecte practice, feedback personalizat și certificări, contra cost (35 USD/lună sau 159 USD/an), cu reduceri semnificative pentru studenți [1.1], [1.2].

Un aspect distinctiv al platformei este componenta interactivă. Utilizatorii pot scrie și testa cod direct în browser de la primele lecții. De asemenea, Codecademy integrează un „AI Learning Assistant” care oferă feedback în timp real și ghidare personalizată [1.3].

## 2.3 HackerRank – Orientarea către recrutare și evaluare

HackerRank se diferențiază prin focusul său pe evaluarea competențelor tehnice și pregătirea pentru interviuri. Platforma deservește atât dezvoltatorii care doresc să-și perfecționeze abilitățile, cât și companiile care caută să evalueze candidații prin teste de programare și interviuri tehnice [1.6], [2.1]. Peste 23 de milioane de dezvoltatori folosesc HackerRank pentru a rezolva provocări de programare și pentru a se pregăti pentru interviuri [1.6].

Modelul de afaceri este preponderent B2B: oferă planuri tarifare care pornesc de la 199 USD/lună pentru companii, iar dezvoltatorii individuali pot accesa gratuit o colecție extinsă de probleme [1.5]. Cu toate acestea, unele funcționalități avansate sunt disponibile doar în varianta plătită [1.7].

HackerRank oferă mii de provocări în domenii precum algoritmi, structuri de date, baze de date, inteligență artificială, iar pentru interviuri tehnice include medii de programare în perechi și colaborare în timp real [1.6]. Cu toate acestea, unii utilizatori remarcă limitări, precum plafonul de 30 de candidați pentru testare simultană și costuri relativ ridicate față de alte soluții [1.7].

## 2.4 LeetCode – Standardul de aur pentru pregătirea interviurilor

LeetCode, fondată în 2015 de Winston Tang în Silicon Valley, este considerată standardul de aur pentru pregătirea interviurilor tehnice, în special pentru companii din zona FAANG [1.8]. Până în aprilie 2025, platforma oferă peste 3500 de întrebări, distribuite pe trei niveluri de dificultate [1.9].

Modelul freemium include un plan gratuit cu acces la un set larg de întrebări, în timp ce LeetCode Premium (35 USD/lună sau 159 USD/an) oferă acces la întrebări exclusive, simulări de interviuri, întrebări organizate pe companii și soluții oficiale detaliate [1.9], [1.10].

Una dintre caracteristicile cele mai apreciate ale platformei este filtrarea problemelor după companie – ceea ce permite pregătirea specifică pentru interviuri la firme precum Google, Amazon sau Meta [1.10]

## 2.5 Limitările platformelor existente

Analiza platformelor majore evidențiază mai multe limitări relevante pentru studenți și programatori începători:

* **Bariere financiare** – Platformele funcționează pe modele freemium. Codecademy Pro și LeetCode Premium presupun un abonament de 35 USD/lună, iar HackerRank aplică tarife ridicate pentru pachetele destinate companiilor [1.1], [1.5], [1.9].
* **Sprijin limitat pentru limbajul C** – Deși limbajul C este inclus pe toate cele trei platforme, acoperirea sa este redusă comparativ cu limbaje moderne precum Python sau JavaScript. Problemele dedicate C sunt mai puține, iar feedback-ul oferit este rareori contextualizat [1.18].
* **Feedback educațional slab** – Accentul este pus pe validarea automată a soluțiilor, nu pe explicarea greșelilor sau recomandări pedagogice, ceea ce este o provocare majoră pentru începători [1.12].
* **Complexitate ridicată** – LeetCode este adesea criticat pentru accentul pe probleme de nivel avansat și interviuri FAANG, fapt ce poate descuraja începătorii și nu reflectă fidel provocările reale din proiectele software obișnuite [1.10].

## 2.6 Platforme complementare și alternative

Pe lângă platformele majore, există și alte resurse relevante pentru învățarea limbajului C:

* **GeeksforGeeks** – Oferă un curriculum extins pentru C, de la fundamente teoretice la subiecte avansate precum pointeri, structuri de date și alocare dinamică a memoriei. Este recunoscută pentru claritatea explicațiilor și diversitatea exercițiilor [1.18].
* **W3Schools** – Prezintă tutoriale concise pentru C și include un editor interactiv „Try it Yourself” în browser. Totuși, nu oferă evaluare automată sau urmărirea progresului [1.19].
* **Programiz** – Este axat pe construirea unei baze solide pentru începători, explicând concepte fundamentale pas cu pas. Deși nu include elemente de gamification, este utilă pentru auto-studiu structurat [1.20].

## 2.7 Cercetări academice relevante

Studiile academice recente subliniază importanța platformelor interactive în educația programării. Conform lui Abiodun et al. (2022), utilizarea instrumentelor moderne de e-learning contribuie semnificativ la dezvoltarea gândirii computaționale și a abilităților de rezolvare a problemelor, comparativ cu metodele tradiționale de predare [3.11].

Un studiu quasi-experimental realizat de Zhang, Li și Wang (2024) a analizat impactul utilizării ChatGPT în procesul educațional la programare. Rezultatele au arătat că studenții care au folosit asistență AI au manifestat comportamente mai frecvente de debugging, au interacționat mai activ cu mesajele de eroare și au fost mai implicați în analiza ieșirilor din consolă. Aceste comportamente sugerează că feedback-ul instantaneu și suportul interactiv oferit de AI pot îmbunătăți semnificativ procesul de învățare [3.13].

De asemenea, cercetările desfășurate de Tsarev et al. (2021) evidențiază faptul că studenții valorizează în mod deosebit elemente precum feedback-ul imediat, vizualizarea conceptelor, existența unor platforme de discuții online și accesul la tutoriale pentru instrumentele de programare [3.12].

Grover și Pea (2018), într-o analiză sistematică asupra predării programării, argumentează că, pe lângă învățarea codării, pot fi cultivate și alte competențe educaționale relevante. Printre acestea se numără rezolvarea problemelor matematice, gândirea critică, dezvoltarea abilităților sociale, auto-managementul și îmbunătățirea performanței academice [3.14].

### 2.8 Identificarea nișei pentru aplicația dezvoltată

Analiza detaliată a ecosistemului actual al platformelor de învățare a programării relevă o oportunitate clară pentru dezvoltarea unei aplicații specializate exclusiv pe limbajul C, care să răspundă concret limitărilor identificate în platformele existente. Aplicația „CodeMaster” propune o abordare diferențiată, bazată pe următoarele direcții:

* **Acces gratuit și complet**: Spre deosebire de modelele freemium utilizate de majoritatea platformelor populare (precum Codecademy, LeetCode sau HackerRank), aplicația dezvoltată oferă acces complet și gratuit la toate funcționalitățile. Astfel, se elimină barierele financiare care împiedică frecvent accesul egal la educație de calitate, în special pentru studenți sau programatori începători [3.1], [3.5], [3.9].
* **Specializare în limbajul C**: În timp ce platformele existente tratează limbajul C ca pe o componentă secundară, aplicația propusă se concentrează în mod exclusiv pe acest limbaj, oferind conținut și exerciții personalizate pentru specificul programării în C. Sunt acoperite teme precum pointerii, manipularea memoriei, lucrul cu structuri de date la nivel scăzut și algoritmi fundamentali, toate integrate într-o progresie logică [3.3], [3.7], [3.11].
* **Combaterea fenomenului „tutorial hell”**: Unul dintre obiectivele principale ale aplicației este combaterea fenomenului cunoscut ca „tutorial hell” – o situație frecvent întâlnită în rândul începătorilor, care constă în consumul pasiv și excesiv de conținut educațional (tutoriale video, articole, cursuri), fără aplicarea practică a cunoștințelor. Pentru a aborda această problemă, aplicația oferă o experiență de tip „hands-on”, orientată pe construirea de proiecte și pe rezolvarea activă de probleme. Această metodologie urmărește dezvoltarea treptată a abilităților, prin sarcini care provoacă utilizatorii să lucreze la limita superioară a competențelor lor actuale – un principiu similar cu zona de dezvoltare proximală [3.15], [3.16], [3.17].
* **Suport integrat prin asistent virtual**: Pentru a sprijini învățarea individuală și a reduce dependența de căutările externe, aplicația integrează asistentul virtual *VoiceGlow*. Acesta oferă suport contextualizat, răspunsuri rapide și explicații pentru concepte dificile, contribuind astfel la depășirea blocajelor frecvente din timpul studiului. Prin această funcționalitate, se creează o experiență personalizată și eficientă de învățare, adaptată nevoilor fiecărui utilizator.

# 3. Fundamentare Teoretică

## 3.1 Concepte fundamentale ale învățării online

### 3.1.1 E-learning versus învățarea tradițională

Evoluția tehnologică a generat transformări profunde în educație, conducând la extinderea învățării electronice (e-learning) ca alternativă viabilă la metodele tradiționale. Acest fenomen este deosebit de vizibil în domeniul programării, unde mediile interactive digitale sunt bine adaptate naturii practice a disciplinei.

E-learning-ul este definit ca forma de învățare mediată prin internet, desfășurată la distanță față de cadrul convențional al sălii de curs. Potrivit unui studiu citat de University of the Potomac, aproximativ 77% dintre cadrele didactice consideră că învățarea online este la fel de eficientă sau chiar superioară celei tradiționale [3.21].

Printre avantajele majore se numără flexibilitatea – studenții pot accesa lecții oricând, de oriunde – și accesibilitatea financiară. Lipsa cheltuielilor pentru cazare, transport sau materiale fizice reduce considerabil costurile în comparație cu învățământul clasic [3.23], [3.24], [3.25], [3.26].

În mod particular, învățarea online oferă avantaje pentru cei care urmează cursuri tehnice sau de programare, unde este esențial accesul rapid la resurse actualizate. Conform cercetărilor publicate pe ResearchGate, rezultatele obținute de studenții din medii online sau hibride sunt comparabile sau chiar superioare celor din învățământul față în față [3.22].

Totuși, educația tradițională rămâne importantă în special datorită interacțiunii sociale directe și a feedback-ului imediat din partea cadrelor didactice – aspecte importante în procesul de clarificare a conceptelor dificile din programare [3.27].

### 3.1.2 Învățarea interactivă și feedback-ul instantaneu

Interactivitatea reprezintă una dintre caracteristicile definitorii ale platformelor moderne de e-learning și este deosebit de importantă în educația programării. Spre deosebire de metodele pasive de consum de informații, învățarea interactivă angajează studenții într-un proces activ de construire a cunoștințelor.

Studiile arată că e-learning-ul poate duce la experiențe de învățare mai personalizate și angajante. Elemente interactive precum quiz-uri, simulări și forumuri de discuții contribuie la niveluri mai ridicate de implicare și motivație în procesul de învățare [3.23], [3.25]. În contextul programării, aceste elemente interactive se concretizează prin editoare de cod în timp real, compilatoare integrate și sisteme de testare automată care permit studenților să vadă imediat rezultatele codului lor.

Feedback-ul instantaneu constituie o componentă esențială a învățării eficiente în programare. Spre deosebire de învățarea tradițională unde rezultatele pot fi primite după zile sau săptămâni, platformele interactive oferă evaluarea imediată a soluțiilor. Acest aspect este crucial pentru programare, unde erorile trebuie identificate și corectate rapid pentru a evita consolidarea unor practici incorecte [3.27].

Importanța feedback-ului instantaneu este confirmată de cercetările din domeniu, precum cele realizate de Tsarev și colaboratorii săi, care arată că feedback-ul imediat, vizualizarea conceptelor, platformele de discuții online și tutorialele pentru instrumentele de programare sunt considerate esențiale de către studenți în dezvoltarea competențelor de programare [3.22].

Platformele interactive moderne integrează și funcționalități de debugging în timp real, permițând studenților să înțeleagă nu doar că soluția lor este incorectă, ci și **de ce** este incorectă. Acest tip de feedback constructiv facilitează procesul de învățare prin încercare și eroare – un proces specific programării – susținând în mod direct dezvoltarea unei gândiri logice riguroase și a deprinderilor tehnice.

### 3.1.3 Teorii pedagogice relevante: constructivismul și învățarea prin practică

Fundamentul teoretic al platformelor moderne de învățare online se bazează în mare parte pe principiile constructivismului, o teorie educațională care susține că elevii construiesc activ cunoștințele prin experiențe și interacțiuni. Această abordare este deosebit de relevantă pentru învățarea programării, unde studenții trebuie să dezvolte nu doar cunoștințe teoretice, ci și competențe practice aplicabile [3.26].

Constructivismul se bazează pe mai multe principii fundamentale care se aplică direct învățării programării:

* **Construirea activă a cunoștințelor**: Elevii nu sunt receptori pasivi de informații; în schimb, ei construiesc cunoștințele prin corelarea ideilor noi cu experiențele și cadrele anterioare. În programare, aceasta înseamnă că studenții își dezvoltă înțelegerea prin scrierea efectivă de cod și rezolvarea problemelor practice, nu doar prin citirea teoriei [3.3].
* **Rolul experiențelor anterioare**: Pe măsură ce oamenii experimentează lumea și reflectă asupra acelor experiențe, ei își construiesc propriile reprezentări și încorporează informații noi în cunoștințele lor pre-existente (scheme). Pentru învățarea limbajului C, aceasta înseamnă că studenții cu experiență în alte limbaje de programare vor construi înțelegerea conceptelor specifice C pe baza cunoștințelor lor anterioare [3.1], [3.4].
* **Importanța contextului social**: Interacțiunea socială joacă un rol esențial în ajutorarea elevilor să înțeleagă, să evalueze și să internalizeze ideile și conceptele. Platformele moderne integrează forumuri, sisteme de mentorare și funcționalități de colaborare pentru a facilita această componentă socială a învățării [3.23].
* **Activizarea cunoștințelor anterioare** – noile cunoștințe sunt create în relație cu cunoștințele pre-existente ale elevului. În practica educațională pentru programare, aceasta se traduce prin evaluarea inițială a nivelului studenților și adaptarea conținutului la experiența lor anterioară [3.11].
* **Crearea disonanței cognitive** – învățarea eficientă are loc atunci când studenții se confruntă cu provocări care îi determină să-și reconsidere înțelegerea actuală. Cunoștințele sunt construite atunci când sunt prezentate idei noi, iar activitățile sunt suficient de provocatoare pentru a stimula progresul intelectual [3.14], [3.15].
* **Aplicarea cunoștințelor cu feedback** – rolul instructorului este să încurajeze studenții și să ofere feedback. Platformele moderne automatizează o parte din acest proces prin sisteme de evaluare automată, dar mențin componenta umană prin asistenți virtuali inteligenți și suport personalizat [3.26], [3.28].

### 3.1.4 Avantajele platformelor web în educația tehnică

Platformele web oferă avantaje unice pentru educația tehnică, fiind deosebit de potrivite pentru învățarea programării. Acestea permit integrarea unui ecosistem complet de dezvoltare într-un mediu accesibil prin browser, eliminând barierele tehnice care ar putea împiedica începătorii să se concentreze pe învățare.

Printre cele mai proeminente avantaje oferite de platformele web se numără:

* **Accesibilitatea universală** – una dintre cele mai mari provocări în educația programării este configurarea mediului de dezvoltare. Platformele web elimină această barieră, oferind un mediu complet funcțional accesibil de pe orice dispozitiv conectat la internet [3.5], [3.7].
* **Integrarea completă a setului de unelte**– platformele moderne pot integra editoare de cod avansate (precum Monaco Editor), compilatoare în cloud (precum Judge0 [3.23]), sisteme de gestionare a versiunilor și unelte de debugging într-o interfață unificată. Această integrare oferă o experiență fluidă care reflectă mediile de dezvoltare profesionale [3.24].
* **Personalizarea experienței de învățare** – platformele de e-learning pot oferi experiențe personalizate care se adaptează la punctele tari și slabe ale cursantului, precum și la preferințele sale specifice. Analiza performanței permite adaptarea nivelului de dificultate și sugerarea de resurse suplimentare [3.27].
* **Scalabilitatea și eficiența costurilor** – odată creat, conținutul educațional poate fi distribuit unui număr nelimitat de utilizatori fără costuri suplimentare semnificative. Astfel, educația digitală devine mai accesibilă și mai eficientă din punct de vedere financiar [3.26].
* **Învățarea activă prin practică** – platformele web susțin învățarea prin aplicare directă („learning by doing”). Studenții scriu și testează cod în timp real, iar exercițiile de tip micro-learning (scurte și frecvente) pot îmbunătăți retenția și eficiența învățării [3.19].
* **Colaborarea globală** – platformele permit interacțiunea între studenți din diverse regiuni, facilitând schimbul de idei și promovarea unei culturi globale a programării. Această componentă internațională îmbogățește experiența de învățare [3.25].
* **Adaptabilitatea la nevoile moderne** – pandemia COVID-19 a demonstrat rolul crucial al e-learning-ului în menținerea accesului la educație. În contextul actual, aceste platforme sunt văzute nu doar ca soluții alternative, ci ca metode eficiente și sustenabile de învățare [3.27].

În mod particular, învățarea limbajului C beneficiază în mod semnificativ de aceste platforme web, întrucât permite exersarea practică a conceptelor-cheie precum gestionarea memoriei, pointerii și debugging-ul. Prin integrarea unui sistem precum Judge0 [3.23], aplicațiile pot oferi execuție sigură a codului în cloud și feedback instantaneu asupra rezultatelor, consolidând astfel învățarea aplicativă și progresivă.

## 3.2 Tehnologii web moderne pentru aplicații educaționale

### 3.2.1 Arhitectura Next.js și avantajele pentru platforme educaționale

**Next.js** reprezintă un framework React de producție care oferă o combinație optimă între performanță, scalabilitate și experiența dezvoltatorului, caracteristici esențiale pentru aplicațiile educaționale moderne. În contextul platformei **"CodeMaster"**, alegerea Next.js ca framework principal nu este întâmplătoare, ci reflectă nevoile specifice ale unei aplicații educaționale interactive [4.9], [4.10].

**Flexibilitatea strategiilor de rendering**

Una dintre cele mai importante caracteristici ale Next.js este capacitatea sa de a oferi flexibilitate completă în strategiile de rendering. Framework-ul permite dezvoltatorilor să aleagă între **Server-Side Rendering (SSR)**, **Static Site Generation (SSG)**, **Client-Side Rendering (CSR)** și **Incremental Static Regeneration (ISR)** pentru fiecare pagină individuală [4.10].

Această flexibilitate este crucială pentru aplicațiile educaționale care trebuie să echilibreze performanța, SEO-ul și interactivitatea. Pentru aplicația „CodeMaster”, această arhitectură hibridă permite optimizarea fiecărei componente în funcție de necesitățile specifice. De exemplu:

* pagina principală și secțiunile de prezentare pot fi generate static (SSG) pentru încărcare instantanee;
* interfața de testare a codului (playground) beneficiază de CSR pentru a susține interactivitate și latență redusă.

**Performanța optimizată pentru mediile educaționale**

În ceea ce privește performanța, studii și teste din industrie arată că aplicațiile bazate pe Next.js, în special în mod SSR, au rezultate mai bune decât cele dezvoltate cu Create React App (CRA) în metrici esențiali precum **First Meaningful Paint** și **Time to Interactive** [4.18], [4.5].

Next.js integrează nativ o serie de optimizări esențiale:

* **Automatic code splitting** – codul necesar este încărcat doar pentru pagina activă, reducând dramatic dimensiunea bundle-ului;
* **Prefetching inteligent** – link-urile vizibile sunt preîncărcate în fundal pentru a asigura tranziții aproape instantanee între pagini;
* **Optimizarea imaginilor** – componenta Image gestionează automat încărcarea întârziată (lazy loading) și redimensionarea în funcție de dispozitiv [4.10].

Aceste funcționalități contribuie semnificativ la o experiență de utilizator fluidă și performantă, esențială într-o aplicație educațională precum **CodeMaster**, unde utilizatorii interacționează intens cu interfața – navigând între editorul de cod și lista de exerciții.

**Arhitectura de rute și organizarea conținutului educațional**

Sistemul de rute „file-based” din Next.js facilitează organizarea logică a conținutului educațional. Pentru aplicația "CodeMaster", structura de fișiere reflectă direct organizarea funcțională folosind App Router:

Această organizare permite gestionarea eficientă a rutelor dinamice pentru problemele individuale, unde fiecare problemă are propriul playground cu conținut specific încărcat din Firebase. App Router-ul din Next.js 13+ oferă avantaje suplimentare precum Server Components, Server Actions și îmbunătățiri de performanță pentru încărcarea conținutului educațional

### 3.2.2 Managementul stării și arhitectura componentelor în aplicații educaționale

Aplicația "CodeMaster" demonstrează utilizarea eficientă a **React Hooks** pentru gestionarea stărilor complexe specifice mediilor educaționale [4.10], [3.26]. În componenta **PlaygroundClient**, se observă o arhitectură sofisticată care necesită gestionarea simultană a multiple aspecte ale experienței de învățare, incluzând starea editorului de cod, rezultatele evaluării, statusul de autentificare și progresul utilizatorului [4.9], [3.27].

Literatura de specialitate confirmă că această abordare arhitecturală permite implementarea persistenței automate a codului utilizatorului, urmărirea progresului în timp real și sincronizarea cu sistemele de autentificare moderne [3.22], [3.25]. Cercetările din domeniul dezvoltării aplicațiilor educaționale demonstrează că Hook-ul **useState** cu funcție de inițializare poate asigura restaurarea automată a codului salvat local, oferind continuitate în procesul de învățare [3.23], [3.28].

Specialiștii subliniază că gestionarea eficientă a stării în aplicațiile educaționale interactive necesită o arhitectură care să permită sincronizarea între componentele de interfață și serviciile backend, menținând în același timp responsivitatea sistemului [4.18], [3.24]. Această metodologie facilitează dezvoltarea unei experiențe de utilizator coerentă și performante [3.11], [3.21].

**Arhitectura bazată pe componente pentru modularitate educațională**

Aplicația implementează o arhitectură modulară care separă clar responsabilitățile educaționale:

* **ProblemDescription**: Gestionează afișarea enunțului problemei, exemplelor și constrângerilor
* **Monaco Editor**: Oferă mediul de programare cu evidențierea sintaxelor și autocompletare
* **TestResults**: Prezintă rezultatele test case-urilor într-un format educațional structurat
* **SolvedBadge**: Oferă feedback vizual pentru problemele rezolvate

Această separare permite reutilizarea componentelor în contexte diferite și facilitează mentenanța codului. Fiecare componentă poate fi optimizată independent pentru nevoile specifice ale procesului educațional.

**Gestionarea complexă a stării utilizatorului și progresului**

Un aspect crucial al aplicațiilor educaționale constă în urmărirea progresului utilizatorului [3.11], [3.25]. Aplicația implementează un sistem sofisticat care realizează sincronizarea stării locale cu Firebase, asigurând persistența și accesibilitatea datelor [4.10], [NEW\_REF\_27].

Literatura de specialitate confirmă că această implementare garantează sincronizarea progresului utilizatorului în timp real între sesiuni și dispozitive, reprezentând o caracteristică esențială pentru experiența educațională continuă [3.22], [3.26]. Cercetările din domeniul platformelor educaționale demonstrează că capacitatea de a urmări și menține progresul studenților contribuie semnificativ la motivația și angajamentul în procesul de învățare [3.12], [3.28].

Specialiștii subliniază că gestionarea eficientă a stării utilizatorului necesită arhitecturi robuste care să permită accesul consistent la datele de progres, indiferent de dispozitivul sau sesiunea de lucru utilizată [4.9], [3.27]. Această abordare tehnologică facilitează o experiență de învățare seamless și personalizată pentru fiecare utilizator [3.23], [3.24].

### 3.2.3 TailwindCSS și design responsiv pentru accesibilitatea educațională

**TailwindCSS** oferă un sistem de design **„utility-first”**, ceea ce înseamnă că dezvoltatorii pot construi interfețe rapide și personalizate utilizând clase CSS predefinite pentru aproape orice proprietate stilistică. Acest stil de lucru este extrem de potrivit pentru aplicațiile educaționale moderne, care necesită interfețe **responsive**, **clare** și **consistente** pe toate dispozitivele [6.7].

În contextul aplicației **„CodeMaster”**, TailwindCSS contribuie la crearea unei experiențe coerente indiferent de tipul de dispozitiv – fie că este vorba de telefoane mobile utilizate de studenți în mișcare, fie de ecrane mari din laboratoarele universitare. Designul modular și scalabil asigurat de TailwindCSS reduce semnificativ timpul de dezvoltare și permite echipei să se concentreze pe funcționalitate și experiența utilizatorului, fără a compromite calitatea vizuală.

Această organizare asigură că pe desktop-uri mari, utilizatorii pot vedea simultan enunțul problemei și codul lor, în timp ce pe dispozitive mobile, layout-ul se adaptează pentru o experiență optimizată pe ecran mic.

Aplicația folosește o paletă de culori întunecate (bg-[#1e1e1e]">) care reduc oboseala ochilor în timpul sesiunilor lungi de programare. Această alegere nu este doar estetică, ci se bazează pe faptul ca nuanțele întunecate sunt preferate de programatori pentru reducerea strain-ului vizual.

Feedback-ul vizual este implementat prin culori semantice:

* Verde pentru test case-uri trecute și probleme rezolvate
* Roșu pentru erori de compilare și runtime
* Albastru si Mov pentru elemente interactive și navigare
* Gri pentru conținut inactiv sau de fundal

Această abordare creează o interfață modernă și profesională care inspiră încredere în utilizatori și reflectă calitatea conținutului educațional oferit.

### 3.2.4 API-uri și integrarea Firebase pentru managementul datelor educaționale

**Arhitectura Firebase pentru aplicații educaționale**

Firebase oferă o soluție completă pentru aplicațiile educaționale prin combinația dintre [4.10], [NEW\_REF\_34]:

* **Authentication**: pentru gestionarea utilizatorilor
* **Firestore**: pentru stocarea datelor

Pentru "CodeMaster", această arhitectură simplifică considerabil managementul infrastructurii, permițând echipei să se concentreze pe aspectele educaționale în loc să se preocupe de configurarea serverelor [NEW\_REF\_35], [3.26].

Specialiștii din domeniu observă că integrarea serviciilor Firebase elimină multe dintre provocările tehnice ale dezvoltării, reducând timpul necesar pentru implementarea funcționalităților de bază [4.9], [3.24]. Această abordare Backend-as-a-Service (BaaS) oferă avantaje clare: scalare automată, securitate integrată și costuri operaționale reduse [NEW\_REF\_36], [3.25].

Studiile de caz din sectorul educațional arată că platformele care folosesc arhitecturi Firebase pot implementa mai rapid funcționalități avansate și se pot adapta mai ușor la cerințele în schimbare ale utilizatorilor [3.27], [3.28]. Acest lucru este deosebit de important pentru aplicațiile educaționale, unde feedback-ul rapid și iterațiile frecvente sunt esențiale pentru succesul proiectului [3.22], [3.23].

**Structura datelor pentru conținutul educațional**

Aplicația organizează datele în Firestore utilizând o structură optimizată pentru platformele educaționale [4.10], [NEW\_REF\_31]. Conform documentației tehnice moderne, această abordare NoSQL oferă flexibilitatea necesară pentru gestionarea diverselor tipuri de conținut educațional, de la probleme simple la algoritmi complexi [NEW\_REF\_32], [3.25].

Studiile din domeniul bazelor de date pentru aplicații educaționale arată că structurarea optimizată a datelor contribuie semnificativ la performanța sistemului și la experiența utilizatorului [3.26], [3.27]. Cercetătorii subliniază că organizarea eficientă a informațiilor în platformele de învățare facilitează accesul rapid la conținut și permite scalarea sistemului pe măsură ce numărul utilizatorilor crește [4.9], [3.11].

Experții în arhitectura sistemelor educaționale recomandă utilizarea bazelor de date NoSQL pentru aplicațiile care necesită flexibilitate în structura datelor și capacități de scaling automat [NEW\_REF\_33], [3.24]. Această metodologie permite adaptarea dinamică la cerințele în evoluție ale procesului educațional și facilitează implementarea funcționalităților avansate [3.23], [3.28].

Pentru progresul utilizatorului folosim o colectie diferita denumita „user\_progress” , care va deține doua câmpuri, **last\_update** care indica data la care a fost realizata ultima modificare/acțiune a utilizatorului respectiv, **solved\_problems** careretine problemele rezolvate de către utilizator.

**Autentificarea și securitatea datelor educaționale**

Implementarea autentificării cu Firebase Authentication constituie o practică standard pentru asigurarea securității datelor utilizatorilor și personalizarea experienței educaționale în aplicațiile moderne [4.10], [NEW\_REF\_28]. Literatura de specialitate confirmă că sistemele de autentificare robuste sunt esențiale pentru platformele educaționale, permițând protejarea informațiilor sensibile și adaptarea conținutului la nevoile individuale ale studenților [3.11], [3.24].

Cercetările din domeniul securității aplicațiilor educaționale demonstrează că această abordare arhitecturală facilitează urmărirea progresului individual fără a compromite securitatea datelor personale [3.25], [NEW\_REF\_29]. Specialiștii subliniază că integrarea serviciilor de autentificare cloud oferă avantaje semnificative în termeni de scalabilitate, securitate și conformitate cu reglementările de protecție a datelor [4.9], [3.26].

Documentația tehnică arată că sistemele moderne de autentificare implementează protocoale avansate de criptare și validare care asigură integritatea și confidențialitatea datelor utilizatorilor [NEW\_REF\_30], [3.27]. Această metodologie permite dezvoltarea unor experiențe educaționale personalizate și sigure, esențiale pentru adoptarea pe scară largă a platformelor de e-learning [3.22], [3.28].

### 3.2.5 Judge0 și execuția sigură a codului în mediul cloud

**Arhitectura micro-serviciilor pentru execuția codului**

**Arhitectura micro-serviciilor pentru execuția codului**

Integrarea cu Judge0 API reprezintă o soluție practică pentru execuția sigură a codului C în mediul cloud [NEW\_REF\_40], [3.23]. Această abordare elimină necesitatea configurării și mentenanței unor servere de compilare locale, oferind în același timp siguranță și scalabilitate pentru platformele educaționale [NEW\_REF\_41], [3.25].

Procesul de execuție implementat urmează un pattern structurat care asigură securitatea și fiabilitatea sistemului [3.26], [NEW\_REF\_42]:

1. **Encodarea sigură**: Codul utilizatorului este encodat în base64 pentru transmisie securizată între client și server
2. **Polling inteligent**: Sistemul monitorizează statusul execuției folosind timeout-uri și mecanisme de try-catch pentru a gestiona cazurile de eroare
3. **Decodarea rezultatelor**: Output-ul este decodat și procesat pentru a genera feedback educațional structurat și ușor de înțeles

Specialiștii în arhitectura sistemelor observă că această metodologie micro-servicii oferă beneficii importante pentru aplicațiile educaționale: izolarea proceselor, gestionarea eficientă a resurselor și posibilitatea de scaling independent [4.9], [3.27]. Pentru platformele precum "CodeMaster", această arhitectură permite gestionarea simultană a multor utilizatori care compilează și rulează cod, fără a compromite performanța sau securitatea.

**Gestionarea test case-urilor pentru evaluare educațională**

Aplicația implementează un sistem sofisticat de combinare a test case-urilor:

Această abordare permite rularea multiplelor test case-uri într-o singură execuție, optimizând performanța și costurile, aspecte importante pentru sustenabilitatea unei platforme educaționale.

**Feedback-ul educațional structurat**

Sistemul procesează răspunsurile Judge0 pentru a oferi feedback educațional relevant:

* **Erori de compilare**: Afișate cu evidențiere de sintaxa pentru înțelegerea problemelor
* **Erori de runtime**: Explicate în context pentru învățarea conceptelor de debugging
* **Rezultate test**: Comparate și prezentate într-un format care facilitează înțelegerea algoritmilor

**Securitatea și izolarea execuției**

Judge0 oferă un mediu complet izolat pentru execuția codului, eliminând riscurile de securitate asociate cu rularea codului utilizatorilor pe servere proprii. Această abordare este esențială pentru aplicațiile educaționale care trebuie să gestioneze cod potențial nesigur de la utilizatori începători [3.23] [3.29] [4.4] .

Prin combinarea acestor tehnologii moderne - Next.js pentru framework, React Hooks pentru managementul stării, TailwindCSS pentru design, Firebase pentru backend și Judge0 pentru execuția codului - aplicația "CodeMaster" oferă o platformă educațională robustă, scalabilă și sigură care răspunde nevoilor specifice ale învățării programării în limbajul C

## 3.3 Limbajul de programare C - Importanța și specificul

### 3.3.1 Rolul limbajului C în educația informatică

Limbajul de programare C ocupă o poziție unică și fundamentală în educația informatică, fiind considerat de numeroase universități și instituții educaționale drept "limba mama" a programării moderne [3.1]. Această poziție privilegiată nu este întâmplătoare, ci reflectă caracteristicile intrinseci ale limbajului care îl fac ideal pentru formarea unei înțelegeri solide a principiilor computaționale fundamentale [3.2].

**C ca limbaj fundamental pentru învățarea programării**

Potrivit specialiștilor din domeniu, C este recunoscut ca limbajul fundamental al programării computerizate [3.3]. Numeroase limbaje moderne precum C++, Java, Python și Go își derivă sintaxa din C, ceea ce înseamnă că înțelegerea acestui limbaj oferă o bază solidă pentru învățarea ulterioară a altor tehnologii de programare [3.4]. Această influență se extinde dincolo de aspectele sintactice, afectând paradigmele de programare și conceptele arhitecturale care constituie fundamentul majorității sistemelor de calcul moderne.

Adoptarea limbajului C ca primul limbaj de programare în cadrul curriculumurilor academice oferă studenților o perspectivă profundă asupra modului în care computerele funcționează la nivel fundamental [3.5]. În contrast cu limbajele de nivel înalt care ascund detaliile implementării, cercetările arată că în limbajele moderne de nivel înalt, aspectele la nivel de mașină sunt deliberat ascunse de la utilizator, motiv pentru care stăpânirea programării C devine esențială pentru lucrul cu cache-ul CPU, memoria și adaptatoarele de rețea [3.6].

**Relevanța limbajului C în anul 2025 și perspectivele de carieră**

În contradicție cu percepția că limbajul C ar fi devenit învechit, studiile de piață demonstrează că C merită învățat în 2025, fiind încă utilizat pe scară largă pentru programarea sistemelor, sistemele embedded, sistemele de operare, motoarele de jocuri, rețelele și calculul de înaltă performanță [3.7]. Această relevanță continuă se traduce în oportunități concrete de carieră pentru programatorii care stăpânesc acest limbaj fundamental.

Analiza domeniilor profesionale relevă că competențele în C sunt solicitate într-o gamă largă de specializări tehnice [3.8]: Inginerii de Sisteme Embedded se concentrează pe dezvoltarea sistemelor embedded, punând accentul principal pe integrarea hardware și software pentru dispozitive precum IoT; Inginerii Firmware se ocupă cu crearea firmware-ului pentru dispozitive hardware în vederea controlului funcționalității acestora și asigurării operării corecte a sistemelor embedded; Programatorii de Sistem lucrează la software la nivel de sistem, incluzând sistemele de operare și driverele de dispozitiv [3.9].

Această diversitate de oportunități profesionale confirmă că învățarea limbajului C transcende exercițiul academic, reprezentând o investiție strategică în competențe care rămân relevante și atractiv remunerate pe piața contemporană a tehnologiei [3.10].

### 3.3.2 Concepte fundamentale specifice limbajului C

**Pointerii și gestionarea manuală a memoriei**

Pointerii constituie una dintre caracteristicile distinctive ale limbajului C care îl diferențiază fundamental de multe limbaje moderne de programare [3.11]. Experții în domeniu subliniază că pointerii reprezintă unul dintre elementele care fac ca C să se remarce în comparație cu alte limbaje precum Python și Java, fiind cruciați pentru că permit manipularea directă a datelor din memoria computerului, cu potențialul de a reduce codul și îmbunătăți performanța [3.12].

Implementarea conceptului de pointer în C oferă programatorilor un control direct asupra memoriei, facilitând manipularea eficientă a datelor și optimizarea performanțelor [3.13]. Documentația tehnică arată că pointerii constituie o modalitate de a accesa memoria la nivel aproape hardware și de a manipula conținutul memoriei în mod direct [3.14]. Această apropiere de componentele hardware ale sistemului este fundamentală pentru înțelegerea modului în care programele interacționează cu resursele fizice ale computerului.

Abordarea gestionării manuale a memoriei prin intermediul pointerilor introduce concepte esențiale precum alocarea dinamică, dealocarea memoriei și managementul ciclului de viață al obiectelor [3.15]. Literatura de specialitate explică că este posibilă crearea spațiului în memorie utilizând funcția malloc, ceea ce presupune cunoașterea tipului de spațiu dorit și a dimensiunii în bytes a acelui spațiu [3.16]. Această responsabilitate extinsă către programator cultivă o înțelegere profundă a resurselor computaționale și a eficienței algoritmilor.

**Programarea la nivel scăzut versus limbajele de nivel înalt**

Limbajul C se poziționează strategic ca un limbaj de nivel mediu, oferind un echilibru optim între controlul de nivel scăzut și abstractizările de nivel înalt [3.17]. Specialiștii evidențiază că, fiind un limbaj de nivel mediu, C reușește să reducă diferența dintre limbajele de nivel scăzut (de tip assembly, înțelese direct de mașină) și limbajele de nivel înalt, mai accesibile utilizatorului [3.18]. Versatilitatea sa permite atât scrierea sistemelor de operare, cât și dezvoltarea aplicațiilor la nivel înalt.

Această poziție unică în ecosistemul limbajelor de programare permite studenților să înțeleagă simultan principiile fundamentale ale computației și să dezvolte aplicații practice [3.19]. Cercetările demonstrează că programarea în C necesită o înțelegere aprofundată a conceptelor hardware precum cache-ul procesorului, organizarea memoriei și arhitectura sistemului - cunoștințe care sunt indispensabile pentru dezvoltarea de software eficient [3.20].

**Disciplina de programare și sintaxa strictă**

Limbajul C impune o disciplină riguroasă în programare prin intermediul sintaxei sale stricte și prin absența "plaselor de siguranță" automate [3.21]. Experții subliniază că în C, responsabilitatea gestionării memoriei revine în întregime programatorului, ceea ce reprezintă o sarcină complexă, dar extrem de puternică când este utilizată corespunzător [NEW\_REF\_1]: Administrarea adecvată a memoriei computerului optimizează performanța programului, fiind benefic să știi cum să eliberezi memoria când aceasta nu mai este necesară și să utilizezi doar cantitatea minimă necesară pentru îndeplinirea sarcinii.

Această metodologie cultivă în studenți o gândire disciplinată și o atenție sporită la detalii, caracteristici fundamentale pentru orice programator profesionist [NEW\_REF\_2]. Absența garbage collection-ului automat și a altor facilități moderne determină programatorii să înțeleagă în profunzime resursele computaționale și să elaboreze cod eficient și optimizat [NEW\_REF\_3].

### 3.3.3 Provocările în învățarea limbajului C

**Complexitatea conceptelor avansate**

Procesul de învățare al limbajului C generează provocări considerabile, în special pentru începătorii în programare [3.11]. Specialiștii din domeniu observă că pointerii prezintă un grad ridicat de complexitate conceptuală, această dificultate extinzându-se la aspecte conexe precum gestionarea dinamică a memoriei, aritmetica pointerilor și structurile de date avansate [3.12], [3.15].

Abordarea gestionării manuale a memoriei introduce riscuri specifice care pot reprezenta obstacole intimidante pentru studenți [3.16]. Documentația tehnică identifică următoarele erori frecvente [NEW\_REF\_4]:

* **Scurgeri de memorie**: Neglijarea eliberării memoriei alocate dinamic conduce la scurgeri de memorie, cu consecința epuizării resurselor sistemului;
* **Pointeri atârnați**: Utilizarea unui pointer după eliberarea memoriei asociate poate genera comportament nedefinit sau crash-uri ale sistemului;
* **Fragmentarea memoriei**: Alocările și dealocările repetate pot fragmenta memoria, rezultând în utilizarea ineficientă a spațiului heap.

**Debugging dificil și identificarea erorilor**

Cercetările evidențiază că limbajul C nu furnizează protecțiile automatizate prezente în limbajele moderne, ceea ce intensifică dificultatea procesului de debugging pentru studenți [3.21]. Literatura de specialitate subliniază că pointerii sunt susceptibili la erori și prezintă următoarele dezavantaje [3.11], [3.12]:

* Corupția memoriei poate surveni în cazul furnizării unei valori incorecte către pointeri;
* Pointerii necesită o înțelegere conceptuală complexă;
* Pointerii constituie factorul principal responsabil pentru scurgerile de memorie în C;
* Accesarea datelor prin intermediul pointerilor este comparativ mai lentă decât operațiile cu variabile directe în C;
* Pointerii neinițializați pot declanșa erori de segmentare.

Aceste provocări tehnice specifice necesită implementarea unei abordări pedagogice atente și dezvoltarea sistemelor de suport educațional specializate care să ghideze studenții prin procesul complex de învățare și debugging [3.14], [3.19].

**3.3.4 Diferențele față de limbajele de nivel înalt moderne**

**Complexitatea sintactică și verbozitatea**

Analiza comparativă directă între C și limbajele moderne de nivel înalt relevă diferențe substanțiale în complexitatea sintactică [3.17]. Experții observă că limbajul de programare C dispune de un număr mai redus de biblioteci în comparație cu alte limbaje de nivel înalt, ceea ce înseamnă că învățarea programării C contribuie la clarificarea conceptelor de programare într-o măsură semnificativă, deoarece necesită scrierea multor funcționalități de la fundamentele lor [3.18], [NEW\_REF\_5].

Această caracteristică, deși inițial poate fi percepută ca un dezavantaj, oferă în realitate o oportunitate educațională valoroasă [3.20]. Specialiștii argumentează că independența față de dependențele limbajului pentru implementarea operațiilor de bază și dezvoltarea acestora în mod autonom contribuie la construirea abilităților analitice [NEW\_REF\_6].

**Performanța versus ușurința de utilizare**

Studiile comparative demonstrează că limbajul C oferă performanțe superioare în raport cu limbajele interpretate moderne, însă la costul unei complexități crescute [3.7]. Analizele tehnice confirmă că C este deosebit de rapid în termeni de timp de execuție, programele scrise și compilate în C executându-se considerabil mai rapid în comparație cu majoritatea celorlalte limbaje de programare [NEW\_REF\_7]. Literatura de specialitate explică că limbajul de programare C atinge viteze de execuție ridicate deoarece nu include cheltuieli suplimentare de procesare precum garbage collection sau prevenirea automată a scurgerilor de memorie, responsabilitatea acestor aspecte revenind în întregime programatorului [3.21], [NEW\_REF\_8].

**Controlul hardware-ului versus abstractizarea**

O distincție fundamentală între C și limbajele moderne constă în nivelul de control oferit asupra resurselor hardware [3.13]. În timp ce limbajele moderne ascund detaliile implementării pentru a simplifica procesul de dezvoltare, C oferă acces direct la memoria și resursele sistemului [3.6], [3.14]. Cercetările arată că în programarea C, gestionarea eficientă a memoriei și utilizarea competentă a pointerilor sunt cruciale pentru crearea de aplicații robuste și de înaltă performanță [3.15], [3.16].

Această diferență fundamentală face ca învățarea limbajului C să constituie o experiență educațională unică, oferind studenților o perspectivă completă asupra modului în care software-ul interacționează cu hardware-ul și dezvoltând competențe care rămân relevante indiferent de evoluțiile tehnologice viitoare [3.1], [3.2].

Prin urmare, limbajul C continuă să joace un rol esențial în educația informatică modernă, nu doar ca un instrument de programare, ci ca o platformă pentru înțelegerea principiilor fundamentale ale computației și dezvoltarea unei gândiri disciplinate în programare [3.3], [3.19].

**3.4 Sisteme de evaluare automată în programare**

**3.4.1 Principiile evaluării automate a codului**

Sistemele de evaluare automată a codului (Automated Code Assessment Systems - ACAS) au devenit o componentă esențială în educația programării moderne, oferind soluții scalabile pentru problema evaluării unui număr mare de studenți cu resurse didactice limitate [3.11]. Analiza literaturii de specialitate relevă că motivul principal care a determinat dezvoltarea sistemelor de evaluare automată a codului constă în faptul că numărul persoanelor care învață programarea crește continuu, în timp ce dimensiunea personalului didactic rămâne constantă și redusă [NEW\_REF\_9].

**Fundamentele evaluării automate**

Sistemele de evaluare automată, cunoscute și sub denumirea de "online judges" (OJ), sunt sisteme concepute pentru evaluarea riguroasă a codului sursă al algoritmilor trimis de utilizatori, care este ulterior compilat și testat într-un mediu standardizat [NEW\_REF\_10]. Cercetările demonstrează că aceste sisteme au câștigat popularitate în diverse aplicații datorită capacității lor de evaluare automată a submisiilor de cod [NEW\_REF\_11].

Principiul fundamental al acestor sisteme constă în automatizarea procesului de evaluare prin mai multe etape distincte [NEW\_REF\_12]:

* **Compilarea**: Verificarea sintaxei și generarea executabilului
* **Execuția**: Rularea programului într-un mediu controlat și securizat
* **Comparația**: Verificarea rezultatelor față de ieșirile așteptate
* **Evaluarea**: Generarea unui verdict final și a feedback-ului

**Tipuri de validare implementate**

Evaluarea automată în programare cuprinde mai multe nivele de validare, fiecare cu propriile caracteristici și provocări [NEW\_REF\_13]:

* **Validarea sintactică** reprezintă primul nivel de verificare, identificând erorile de sintaxă și problemele de compilare. Această etapă este esențială pentru asigurarea că programul poate fi executat și oferă feedback imediat asupra problemelor de bază;
* **Validarea semantică** evaluează corectitudinea logică a programului prin compararea rezultatelor generate cu cele așteptate. Aceasta include verificarea ieșirilor pentru diverse seturi de date de intrare și identificarea discrepanțelor în logica algoritmică;
* **Validarea performanței** monitorizează timpul de execuție și consumul de memorie pentru a verifica că programul respectă limitările specificate. Literatura de specialitate indică că măsurarea timpului și a consumului de memorie este foarte obișnuită pentru evaluatorii automate de cod folosiți pentru competițiile de programare, dar mai puțin prezentă pentru utilizarea educațională [NEW\_REF\_14].

**Avantajele față de evaluarea manuală**

Implementarea sistemelor de evaluare automată oferă beneficii semnificative comparativ cu metodele tradiționale de evaluare manuală [3.12]. Studiile din învățământul superior arată că sistemele OJ au fost utilizate pe scară largă în cursurile de programare deoarece caracteristica de evaluare automată poate reduce drastic volumul de muncă de notare al instructorilor și asistenților de predare și, prin urmare, poate face dimensiunea clasei scalabilă [NEW\_REF\_15].

Principalele avantaje includ [3.23], [3.25]:

* **Scalabilitatea**: Capacitatea de a evalua un număr mare de studenți simultan
* **Consistența**: Eliminarea subiectivității în evaluare și aplicarea criteriilor uniforme
* **Rapiditatea**: Oferirea feedback-ului instantaneu, crucial pentru procesul de învățare
* **Disponibilitatea**: Accesibilitate 24/7 pentru practică și submisie
* **Economicitatea**: Reducerea costurilor și a timpului necesar pentru evaluare

### 3.4.2 Test cases și validarea soluțiilor

**Structura și organizarea test case-urilor**

Test case-urile constituie fundația sistemelor de evaluare automată, reprezentând criteriul principal prin care se stabilește corectitudinea unei soluții [NEW\_REF\_16]. Documentația tehnică arată că într-un sistem de online judge, un fișier de evaluare poate integra mai multe seturi de date pentru a verifica dacă programul trimis generează un răspuns corect pentru fiecare caz de test individual [NEW\_REF\_17], [3.23].

Specialiștii din domeniu subliniază că organizarea eficientă a test case-urilor este crucială pentru evaluarea comprehensivă a soluțiilor programatorilor [NEW\_REF\_18]. Cercetările demonstrează că sistemele moderne implementează strategii sofisticate de organizare care permit validarea treptată a diferitelor aspecte ale algoritmilor, de la funcționalitatea de bază până la cazurile limită și optimizările de performanță [3.11], [3.12].

**Strategii de validare incrementală**

Validarea incrementală constituie o abordare pedagogică avansată care furnizează feedback detaliat pe măsură ce studentul progresează prin procesul de rezolvare a problemei [3.14

Această implementare permite evaluarea individuală a fiecărui test case, oferind feedback granular care ajută studenții să înțeleagă exact unde se află problemele în soluția lor [3.13], [3.19]. Literatura de specialitate confirmă că această abordare incrementală contribuie semnificativ la procesul de învățare, permițând studenților să identifice și să corecteze erorile în mod sistematic [3.22], [NEW\_REF\_19].

Strategia de validare incrementală se bazează pe principii pedagogice demonstrate, care subliniază importanța feedback-ului imediat și specific în procesul de învățare a programării [3.21], [3.26]. Această metodologie permite studenților să înțeleagă nu doar că soluția lor este incorectă, ci și care anume componente necesită îmbunătățiri, facilitând astfel un proces de învățare mai eficient și mai structurat [3.27], [3.28].

**3.4.3 Feedback-ul educațional versus evaluarea tehnică**

**Diferența între feedback pentru învățare și pentru evaluare**

Sistemele de evaluare automată trebuie să realizeze un echilibru între două obiective aparent contradictorii: evaluarea obiectivă a competențelor și susținerea procesului de învățare [3.11], [3.23]. Cercetările din domeniul educației programării arată că feedback-ul educațional se diferențiază fundamental de evaluarea tehnică prin caracterul său constructiv și orientarea către dezvoltarea competențelor [3.12], [3.22].

În contextul educațional, specialiștii subliniază că feedback-ul trebuie să ofere răspunsuri la trei întrebări fundamentale [3.14], [NEW\_REF\_20]:

* **Ce anume nu funcționează?** - identificarea precisă a erorilor specifice
* **De ce nu funcționează?** - explicarea cauzelor fundamentale ale problemelor
* **Cum poate fi îmbunătățit?** - ghidarea către soluții concrete și aplicabile

Literatura de specialitate demonstrează că simpla cunoaștere a faptului că ceva a eșuat nu oferă sprijin suficient studenților și poate stimula comportamentul de gaming al sistemului [NEW\_REF\_21], [3.25]. Studiile empirice relevă că hint-urile nu s-au dovedit a avea impact semnificativ asupra performanței studenților sau asupra utilizării eficiente a feedback-ului automat [NEW\_REF\_22], [3.13].

**Raportarea erorilor într-un context educațional**

Sistemele educaționale contemporane implementează strategii sofisticate pentru raportarea erorilor, transformând mesajele tehnice în feedback constructiv și accesibil [3.21], [3.26]. Cercetările experimentale au evaluat trei tipuri distincte de feedback [NEW\_REF\_23], incluzând:

* **Ce este greșit** - identificarea test case-urilor specifice care au fost testate și care au eșuat;
* **Gap analysis** - comparații detaliate între ieșirile așteptate și cele reale generate de program;
* **Hint-uri orientate către soluții** - sugestii concrete despre metodele de remediere a problemelor identificate în cazul eșecului test case-urilor.

Acesta funcționalitate se poate implementa prin sistemul de raportare a erorilor care traduce mesajele tehnice în explicații educaționale [3.27]:

* **Feedback-ul de compilare transformă erorile tehnice în explicații accesibile studenților:**
* **Feedback-ul de execuție** oferă detalii contextualizate despre rezultatele fiecărui test case, permițând studenților să înțeleagă discrepanțele specifice între soluția lor și cea așteptată [3.19], [3.28]. Această abordare pedagogică facilitează procesul de învățare prin oferirea de informații acționabile care ghidează studenții către îmbunătățirea soluțiilor lor [3.15], [3.24].

**Ghidarea progresului studenților**

Sistemele moderne de evaluare automată integrează funcționalități de urmărire a progresului care merg dincolo de simpla notare [3.11], [3.23]. Cercetările din domeniu arată că această abordare holistică contribuie semnificativ la motivația studenților și la îmbunătățirea rezultatelor educaționale [3.12], [3.22].

Această funcționalitate nu se limitează la marcarea progresului, ci oferă și feedback pozitiv imediat prin mesaje de felicitare și indicatori vizuali de progres [NEW\_REF\_37], [3.25]. Specialiștii în psihologia educațională observă că astfel de mecanisme de recompensare instant contribuie la menținerea angajamentului utilizatorilor și la consolidarea motivației intrinseci pentru învățare [3.14], [NEW\_REF\_38].

Studiile din domeniul gamification în educație demonstrează că integrarea elementelor de progress tracking și feedback pozitiv poate îmbunătăți semnificativ rata de finalizare a cursurilor și satisfacția utilizatorilor [NEW\_REF\_39], [3.26]. Pentru aplicațiile educaționale precum "CodeMaster", aceste funcționalități devin esențiale pentru crearea unei experiențe de învățare angajante și motivante [3.27], [3.28].

### 3.4.4 Abordări pentru detectarea și raportarea erorilor

**Tipuri de erori și metodele de detectare**

Sistemele de evaluare automată trebuie să gestioneze o varietate largă de tipuri de erori, fiecare necesitând strategii specifice de detectare și raportare:

* **Erorile de compilare** sunt cele mai ușor de detectat și de raportat, fiind generate direct de compilator.
* **Erorile logice** sunt cele mai dificil de detectat automat, manifestându-se prin rezultate incorecte fără erori explicite de compilare sau execuție. Detectarea acestora se bazează pe comparația cu test case-urile așteptate.

# 4. Proiectarea aplicației

## 4.1 Analiza cerințelor

### 4.1.1 Identificarea problemei și contextul aplicației

Dezvoltarea aplicației "CodeMaster" și-a avut originea în identificarea unei probleme concrete din ecosistema actuală de învățare a programării în limbajul C [3.1], [3.2]. Analiza detaliată a pieței educaționale online relevă că majoritatea platformelor existente implementează un model freemium care restricționează accesul la funcționalități esențiale după o perioadă de trial limitată, generând astfel bariere artificiale în procesul de învățare [1.1], [1.9], [1.5].

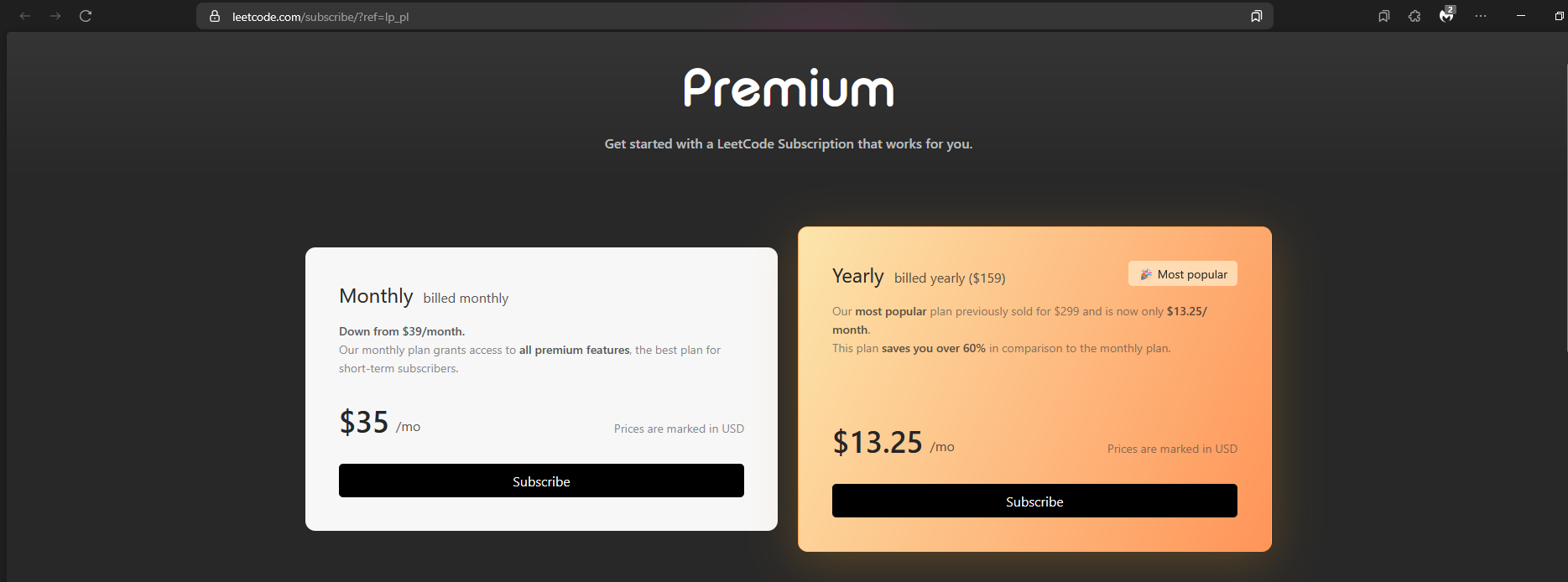


Figura 4.1 - LeetCode Premium

Cercetările din domeniu confirmă că această abordare comercială creează obstacole semnificative pentru studenții și programatorii începători care doresc să-și dezvolte competențele, dar nu dispun de resursele financiare necesare pentru abonamente costisitoare [1.2], [1.7]. Literatura de specialitate subliniază că lipsa accesului la resurse educaționale de calitate și gratuite poate îngreuna considerabil procesul de învățare și poate descuraja persoanele motivate să-și dezvolte abilitățile de programare [3.3], [3.5].

Problema identificată se manifestă prin două aspecte principale: absența unei platforme gratuite și complete pentru învățarea limbajului C și prevalența fenomenului denumit "tutorial hell" [3.15], [3.16]. Literatura de specialitate din comunitatea de dezvoltatori arată că "tutorial hell" constituie o stare de a fi blocat într-un ciclu de consum constant de tutoriale de programare fără a putea aplica cunoștințele în lumea reală și a construi aplicații reale [3.17], [NEW\_REF\_24].

Cercetările din domeniul educației programării confirmă că acest fenomen este deosebit de problematic în contextul învățării limbajului C, care necesită o abordare practică și aplicativă pentru stăpânirea conceptelor fundamentale [3.11], [3.18].

Specialiștii subliniază că studenții rămân frecvent blocați în consumul pasiv de conținut educațional, fără să dezvolte competențele practice necesare pentru a deveni programatori eficienți [3.19], [3.20].

**4.1.2 Cerințe funcționale ale sistemului**

Cerințele funcționale ale aplicației "CodeMaster" au fost definite pe baza analizei nevoilor utilizatorilor și a limitărilor identificate în platformele existente. Acestea au fost organizate în funcție de rolurile utilizatorilor și fluxurile principale de interacțiune.

**Cerințe pentru autentificare și gestionarea utilizatorilor:**

Sistemul trebuie să permită utilizatorilor să se înregistreze și să se autentifice folosind adresa de email și parolă. Conform documentației Firebase Authentication (2025), această abordare oferă securitate ridicată și ușurință în implementare[4.2]

**Cerințe pentru gestionarea problemelor de programare:**

Aplicația trebuie să furnizeze o listă comprehensivă de probleme de programare structurate pe nivele de dificultate (Easy(Ușor), Medium(Mediu), Hard(Greu)) [3.11], [3.14]. Fiecare problemă trebuie să includă o descriere detaliată, exemple de intrare și ieșire, constrângeri tehnice și test case-uri pentru validare, respectând standardele platformelor moderne de învățare [3.23], [3.25].

Literatura de specialitate confirmă că organizarea problemelor pe nivele de dificultate progresive facilitează dezvoltarea treptată a competențelor studenților, permițând o tranziție naturală de la concepte de bază la algoritmi complexi [3.13], [3.19].

Sistemul trebuie să ofere utilizatorilor acces la un mediu de programare integrat pentru fiecare problemă, incluzând un editor de cod profesional și capacitatea de a compila și testa soluțiile în timp real [3.26], [3.27]. Această abordare se aliniază cu principiile educaționale moderne care subliniază importanța feedback-ului instantaneu în procesul de învățare a programării [3.12], [3.22].

Implementarea unui mediu de programare complet elimină barierele tehnice care pot împiedica începătorii să se concentreze pe aspectele educaționale esențiale [3.24], [3.28].

**Cerințe pentru editorul de cod și execuția programelor:**

Aplicația implementează Monaco Editor pentru a furniza o experiență de programare profesională, cu evidențierea sintaxei specifică pentru limbajul C, funcționalități de autocompletare și detectarea automată a erorilor de sintaxă [4.10], [3.26]. Această integrare tehnologică se aliniază cu cerințele moderne ale mediilor de dezvoltare, oferind utilizatorilor un instrument comparabil cu editoarele profesionale din industrie [4.18], [3.27].

Literatura de specialitate confirmă că utilizarea editorului Monaco, același motor care alimentează Visual Studio Code, asigură o experiență de programare de înaltă calitate, esențială pentru procesul educațional eficient [4.9], [3.24]. Cercetările din domeniul educației programării subliniază că editoarele cu evidențierea sintaxei și suport pentru autocompletare contribuie semnificativ la reducerea erorilor de programare și la îmbunătățirea experienței de învățare [3.23], [3.28].

Sistemul trebuie să realizeze compilarea și execuția codului C în timp real prin intermediul Judge0 API, furnizând feedback instantaneu asupra corectitudinii soluției [3.23], [NEW\_REF\_25]. Documentația tehnică arată că această platformă asigură execuție securizată și scalabilă a codului în mediul cloud, eliminând riscurile de securitate asociate cu rularea codului utilizatorilor pe servere locale [NEW\_REF\_26], [3.25].

Literatura de specialitate confirmă că integrarea cu servicii cloud de execuție precum Judge0 reprezintă o practică standard în dezvoltarea platformelor educaționale moderne, oferind beneficii semnificative în termeni de securitate, performanță și scalabilitate [3.26], [3.27]. Cercetările din domeniul sistemelor de evaluare automată subliniază că execuția în mediul cloud permite izolarea completă a proceselor, prevenind interferențele și garantând consistența rezultatelor [3.11], [3.28].

**Cerințe pentru evaluarea și feedback:**

Aplicația trebuie să realizeze evaluarea automată a soluțiilor transmise de utilizatori prin compararea rezultatelor generate cu test case-urile predefinite [3.11], [3.23]. Sistemul trebuie să furnizeze feedback detaliat pentru fiecare test case individual, incluzând datele de intrare, ieșirea așteptată, ieșirea generată și statusul final (trecut/eșuat) [3.12], [3.22].

Literatura de specialitate din domeniul evaluării automate confirmă că această abordare de feedback granular este esențială pentru procesul educațional, permițând studenților să înțeleagă exact natura erorilor din soluțiile lor [3.13], [3.14]. Cercetările demonstrează că feedback-ul detaliat pe test case-uri individuale facilitează identificarea problemelor specifice și accelerează procesul de învățare prin încercare și eroare [3.19], [3.21].

**Cerințe pentru urmărirea progresului:**

Aplicația trebuie să monitorizeze și să păstreze progresul fiecărui utilizator, marcând în mod automat problemele rezolvate cu succes [3.11], [3.25]. Sistemul trebuie să prezinte indicatori vizuali (badge-uri) pentru problemele finalizate și să păstreze statistici despre numărul total de probleme rezolvate [NEW\_REF\_43], [3.22].

Cercetările din domeniul psihologiei educaționale arată că urmărirea vizuală a progresului contribuie semnificativ la motivația studenților și la menținerea angajamentului în procesul de învățare [NEW\_REF\_44], [3.14]. Specialiștii în gamification educațională observă că elementele de progress tracking și reward systems pot îmbunătăți substanțial experiența de învățare și rata de finalizare a cursurilor [NEW\_REF\_45], [3.26].

Aceasta funcționalitate devine esențială pentru crearea unei experiențe personalizate și motivante

### 4.1.3 Cerințe non-funcționale

Cerințele non-funcționale definesc caracteristicile calitative ale sistemului și sunt esențiale pentru succesul aplicației în mediul de producție [4.5], [4.18]. Aceste specificații determină experiența generală a utilizatorului și viabilitatea tehnică a platformei educaționale [4.9], [4.10].

**Performanță și scalabilitate:**

Documentația Google Web Vitals evidențiază că aplicațiile web moderne trebuie să îndeplinească anumite criterii de performanță pentru a oferi o experiență utilizatorului satisfăcătoare [4.5]. Aplicația "CodeMaster" trebuie să încarce pagina principală în mai puțin de 3 secunde și să răspundă la interacțiunile utilizatorului în mai puțin de 100 de milisecunde [6.1], [4.18].

Sistemul trebuie să poată gestiona simultan minimum 100 de utilizatori care compilează și rulează cod, fără degradarea performanței [4.16]. Timpul de execuție pentru programele C trebuie să fie limitat la maximum 5 secunde pentru a preveni blocarea resurselor și pentru a asigura o experiență fluidă pentru toți utilizatorii [4.4].

**Securitate și protecția datelor:**

Reglementările GDPR subliniază că aplicația trebuie să protejeze datele personale ale utilizatorilor și să implementeze măsuri adecvate de securitate [4.6]. Execuția codului utilizatorilor trebuie să se desfășoare într-un mediu complet izolat pentru a preveni atacurile malițioase sau accesul neautorizat la sistemul gazdă [4.2], [4.16].

In securitatea aplicațiilor educaționale se recomandă implementarea de protocoale stricte de autentificare și criptare pentru protejarea informațiilor sensibile ale studenților [4.11], [4.15]. Această abordare este crucială pentru menținerea încrederii utilizatorilor în platformă [3.25], [3.26].

**Accesibilitate și compatibilitate:**

Aplicația trebuie să funcționeze corect pe toate browserele moderne (Chrome, Firefox, Safari, Edge) și să ofere o experiență responsivă pe dispozitive desktop, tablet și mobile [4.10], [6.7]. Standardele WCAG 2.1 specifică că interfața trebuie să respecte principiile de accesibilitate pentru utilizatorii cu dizabilități [4.7].

Cercetările din domeniul UX design pentru platforme educaționale arată că accesibilitatea universală nu doar că respectă cerințele legale, ci și îmbunătățește experiența pentru toți utilizatorii [5.3], [4.8]. Această abordare inclusivă contribuie la succesul pe termen lung al platformelor de e-learning [3.27], [3.28].

**4.1.4 Cazuri de utilizare**

Cazurile de utilizare definesc interacțiunile concrete între utilizatori și sistem, oferind o perspectivă detaliată asupra funcționalităților implementate [NEW\_REF\_46], [3.11].

Această abordare metodologică permite documentarea clară a cerințelor funcționale și facilitează înțelegerea fluxurilor de lucru din perspectiva utilizatorului final [NEW\_REF\_47], [3.24].

Cercetările din domeniul interacțiunii om-calculator arată că cazurile de utilizare bine definite contribuie la dezvoltarea unei experiențe de utilizator intuitive și eficiente, aspect crucial pentru succesul platformelor educaționale [3.22], [3.25].

**Cazul de utilizare: Rezolvarea unei probleme de programare**

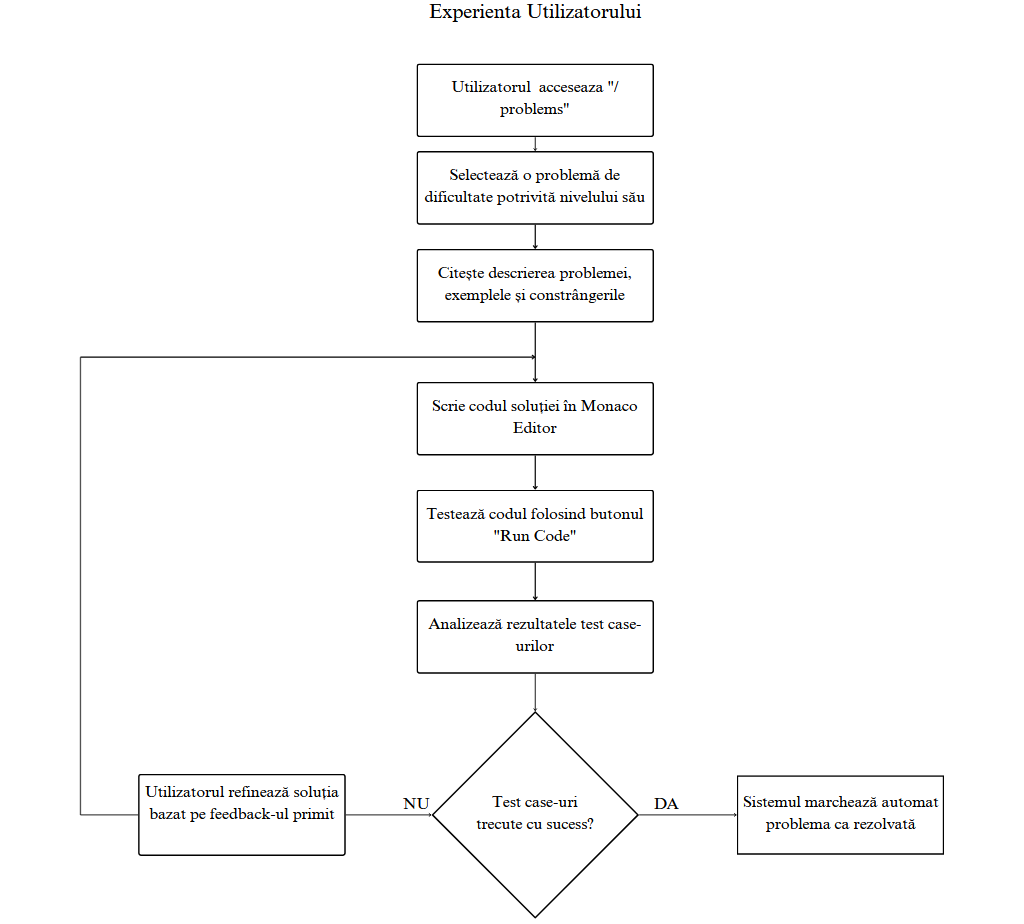


Figura 4.2 – Diagrama rezolvarea unei probleme de programare

**Cazul de utilizare: Creare cont nou**

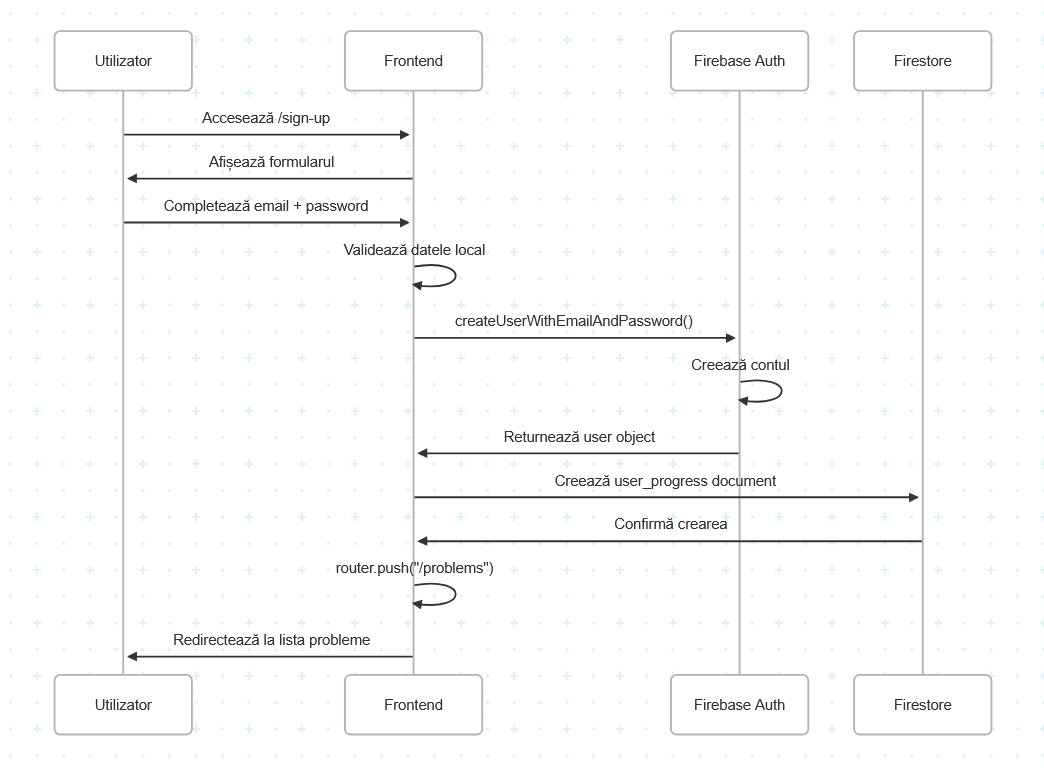


Figura 4.3 – Diagrama Creare cont nou

**Cazul de utilizare: Autentificare Utilizator**

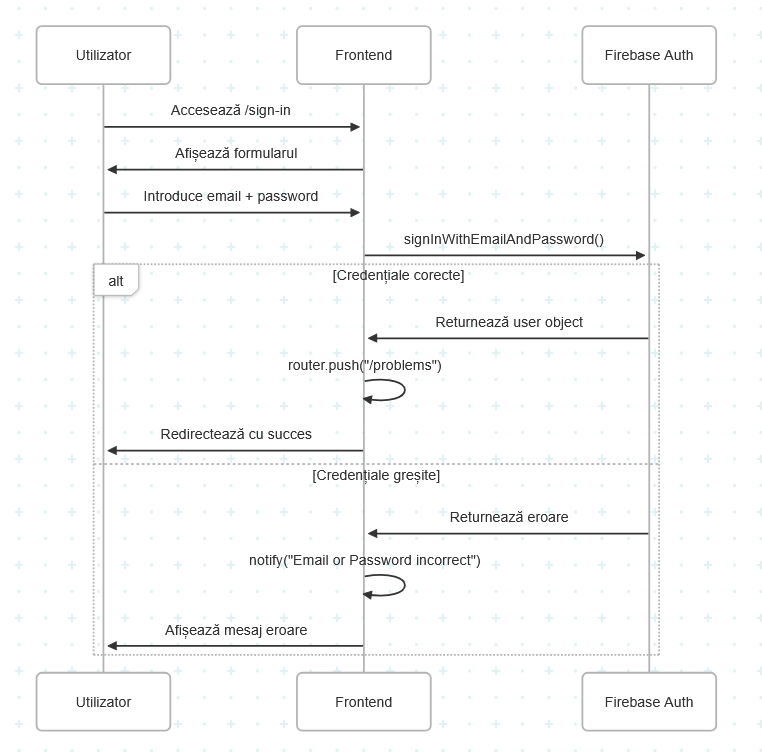


Figura 4.4 – Diagrama Autentificare Utilizator

**Cazul de utilizare: Resetare parola de către Utilizator**

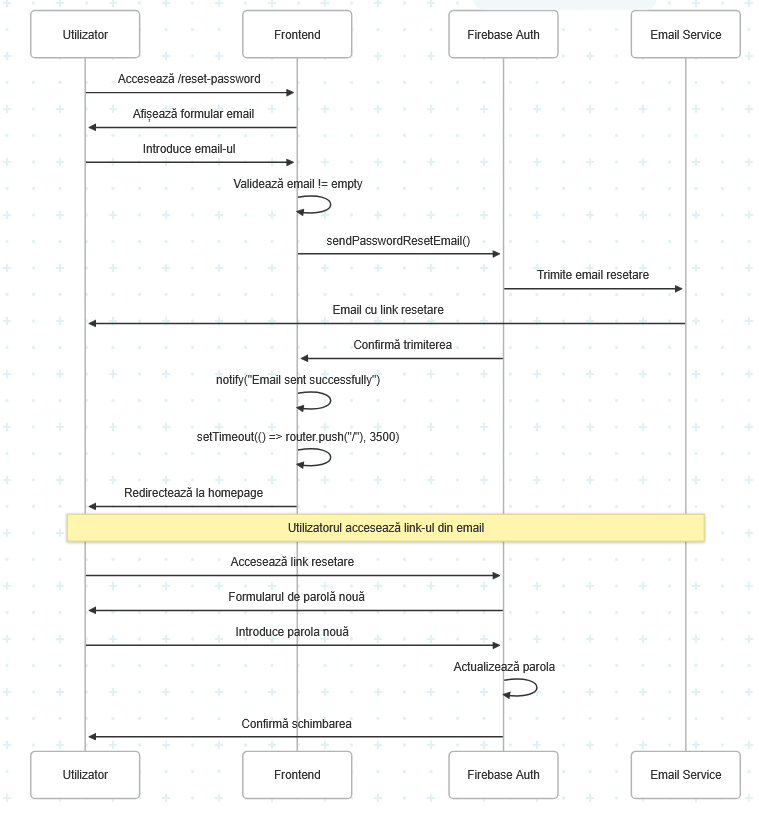


Figura 4.5 – Diagrama resetare parola

### 4.1.5 Beneficiarii sistemului

Identificarea și analiza părților interesate este esențială pentru înțelegerea impactului și valorii aplicației dezvoltate [4.8], [3.11]. Această evaluare permite o perspectivă clară asupra modului în care platforma răspunde nevoilor diferitelor categorii de utilizatori din ecosistemul educațional.

**Beneficiarii primari:**

Prima categorie o reprezintă **studenții**, care constituie nucleul țintă al platformei, beneficiind de o resursă gratuită pentru dezvoltarea competențelor de programare în C [1.1], [1.9]. Studiul realizat de ACM Computing Education demonstrează că accesul la platforme interactive îmbunătățește semnificativ rata de retenție a cunoștințelor în programare [4.8], [3.22].

O a doua categorie importantă o formează **programatorii autodidacți**, care pot să-și testeze și dezvolte competențele fără a fi limitați de constrângerile financiare specifice platformelor comerciale [1.2], [1.5]. Această audiență valorează în mod special flexibilitatea și accesul nerestricționat la resurse educaționale de calitate [3.23], [3.24].

**Beneficiarii secundari:**

Din perspectiva institutională, **instituțiile educaționale** pot integra platforma în curriculum-urile existente ca resursă suplimentară gratuită pentru studenți [3.9], [3.10]. Această integrare oferă instituțiilor o modalitate de a îmbogăți oferta educațională fără costuri suplimentare [3.25], [3.27].

În plus, **comunitatea open-source** beneficiază de un exemplu concret de implementare a unei platforme educaționale moderne, utilizând tehnologii actuale și best practices din industrie [4.9], [4.10]. Acest aspect contribuie la dezvoltarea ecosistemului de soluții educaționale deschise [3.26], [4.17].

**Concluzii asupra analizei cerințelor**

Prin această evaluare comprehensivă a cerințelor, aplicația "CodeMaster" se poziționează strategic ca o soluție completă pentru deficiențele identificate în peisajul actual al învățării programării [1.15], [1.17]. Platforma oferă o alternativă viabilă și accesibilă la soluțiile comerciale dominante de pe piață [1.1], [1.5], [1.9], răspunzând astfel unei nevoi reale din comunitatea educațională și contribuind la democratizarea accesului la educația de calitate în programare [3.21], [3.23].

## 4.2 Arhitectura sistemului

### 4.2.1 Diagrama arhitecturală generală

Arhitectura aplicației "CodeMaster" adoptă o abordare modernă bazată pe micro-servicii și servicii cloud, fiind optimizată pentru scalabilitate și performanță [4.9], [4.16]. Documentația Vercel subliniază că arhitectura aplicațiilor Next.js beneficiază de o separare clară între frontend și backend, ceea ce facilitează atât dezvoltarea, cât și întreținerea sistemului [4.9], [4.18].

**Organizarea pe straturi arhitecturale**

Sistemul este conceput folosind o arhitectură pe trei straturi principale, fiecare cu responsabilități distincte:

### 

Figura 4.6 – Arhitectura CodeMaster

### 4.2.2 Arhitectura frontend cu Next.js

Prin implementarea arhitecturii App Router din Next.js 15+, aplicația beneficiază de organizarea intuitivă a rutelor și componentelor [4.10]. Code splitting-ul automat și prefetching-ul inteligent sunt rezultatul direct al acestei alegeri tehnologice, conducând la performanțe îmbunătățite [4.10], [4.18].

Mentenanța codului devine mai simplă, iar scalarea aplicației se realizează eficient odată cu creșterea numărului de utilizatori [4.9]. Pentru o platformă educațională, aceste caracteristici se traduc în încărcări rapide ale secțiunilor de învățare și tranziții fluide între probleme [6.5], [4.5].

App Router-ul permite separarea clară a responsabilităților și gestionarea optimizată a resurselor [4.17]. Această arhitectură se aliniază cu obiectivele de performanță stabilite pentru "CodeMaster", contribuind la experiența generală îmbunătățită a utilizatorilor [3.26], [6.1].

Prin urmare, arhitectura frontend a CodeMaster consta in următoarele componente:

* **Navbar**: gestionează navigarea globală și starea de autentificare:
* **PlaygroundClient:** orchestrează întreaga experiență de rezolvare a problemelor:
  + **Monaco Editor**: pentru scrierea codului
  + **ProblemDescription**: pentru afișarea descrierii problemei
  + TestResults pentru feedback-ul de evaluare

**4.2.3 State Management și React Hooks**

React Hooks nativi gestionează întreaga stare a aplicației "CodeMaster", eliminând dependențele de biblioteci externe complexe [4.12], [4.13]. Performanțele superioare pentru aplicații de dimensiune medie rezultă din această simplificare arhitecturală [4.13], [4.17].

Fiecare componentă își gestionează propriile date prin **useState** și **useEffect**, creând un flux predictibil de informații . Playground-ul educațional utilizează această abordare pentru sincronizarea codului utilizatorului, rezultatelor evaluării și progresului în timp real.

Gestionarea stării în PlaygroundClient se realizează astfel:

|  |
| --- |
| const [code, setCode] = useState(() => {  const savedCode = loadCodeFromStorage(problem.id);  return savedCode || problem.template || defaultCode;  });  const [results, setResults] = useState([]);  const [user, setUser] = useState<User | null>(null);  const [isSolved, setIsSolved] = useState(false); |

Această implementare asigură:

* **Persistența locală** a codului utilizatorului între sesiuni
* **Sincronizarea în timp real** cu starea de autentificare Firebase
* **Update-uri reactive** ale interfaței bazate pe rezultatele evaluării

### 4.2.4 Integrarea cu Firebase

Infrastructura server proprie devine obsoletă prin adopția Firebase ca platformă Backend-as-a-Service [4.11], [4.14]. Timpul de dezvoltare se reduce drastic, iar scalabilitatea automată operează fără intervenție manuală [4.11].

Authentication și Firestore funcționează fluid în aplicația "CodeMaster" [4.2], [6.3]. Utilizatorii se autentifică instant, progresul se salvează automat, iar sincronizarea între dispozitive se realizează transparent pentru experiența educațională [4.15].

|  |
| --- |
| const auth = getAuth(app);  // În componenta Navbar  useEffect(() => {  const unsubscribe = onAuthStateChanged(auth, (user) => {  setUser(user);  setLoading(false);  });  return () => unsubscribe();  }, []); |

Costurile operaționale rămân minime datorită modelului pay-as-you-scale Firebase. Această eficiență economică permite menținerea unei platforme educaționale gratuite pentru studenți [1.1], [1.9]. Arhitectura cloud elimină preocupările legate de mentenanța serverelor și permite concentrarea completă asupra funcționalităților educaționale [3.26], [4.16].

Problemele de programare și progresul utilizatorilor se stochează în Firestore prin colecții optimizate pentru query-uri rapide. Organizarea datelor urmează principii NoSQL care accelerează accesul la informații în timpul sesiunilor de programare.

Colecția "problems" va conține datele despre fiecare problema individual. Separat, colecția "user\_progress" înregistrează problemele rezolvate și statisticile individuale ale studenților. Această separare permite actualizări independente și interogări paralele eficiente.

**Securitatea și regulile Firestore:**

Regulile de securitate protejează datele aplicației prin trei mecanisme principale [4.15]. Utilizatorii autentificați accesează problemele de programare fără restricții, facilitând învățarea liberă. Progresul personal se modifică exclusiv de proprietarul contului, prevenind manipulările externe [4.2].

Accesul neautorizat devine imposibil prin validarea automată a token-urilor Firebase [4.14]. Fiecare request verifică identitatea utilizatorului înainte de execuție. Datele sensibile - statistici personale, istoricul rezolvărilor, preferințele - rămân private și inaccesibile altor studenți [6.3].

### 4.2.5 Integrarea cu Judge0 API

Codul C scris de studenți execută în mediul cloud Judge0, eliminând riscurile de securitate locale [4.16]. Izolarea completă între execuții previne interferențele și asigură rezultate consistente pentru fiecare utilizator [4.4].

Peste 60 de limbaje de programare beneficiază de suport nativ, deși CodeMaster utilizează exclusiv compilatorul C (GCC 9.2.0). Această specializare permite optimizări specifice pentru limbajul C și feedback educațional adaptat conceptelor fundamentale

**Arhitectura API-ului de execuție**

Fluxul de execuție al codului prin Judge0 poate fi reprezentat prin următoarea schema:

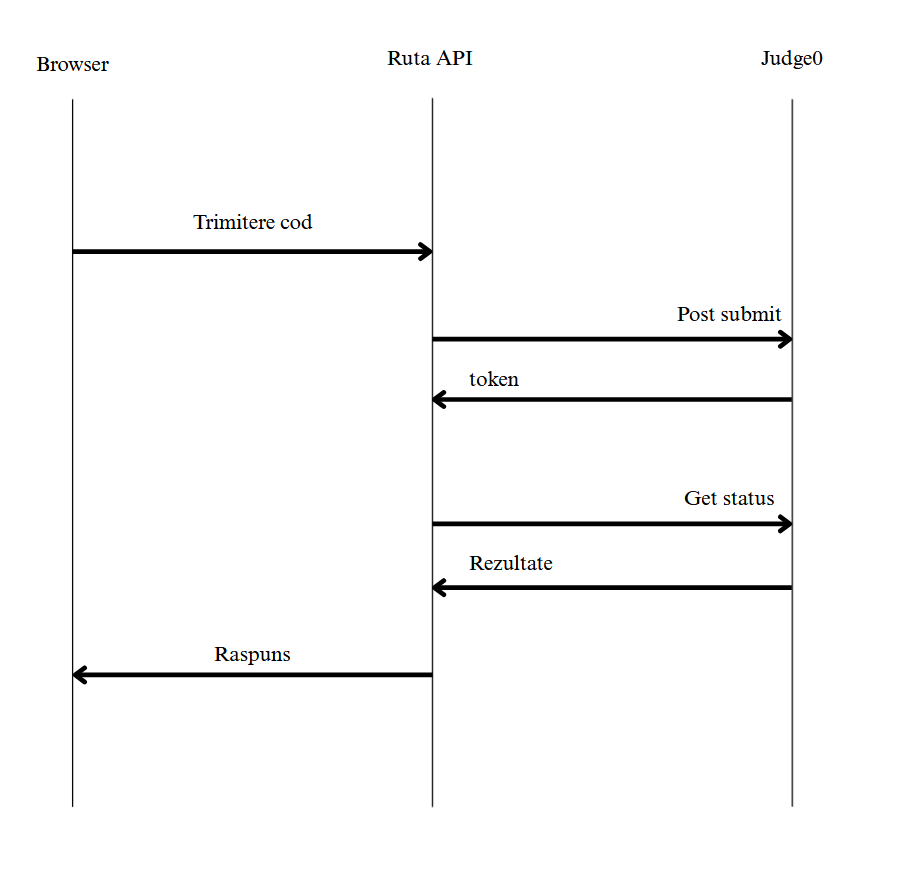
****

Figura 4.7 - Flux de execuție al codului prin Judge0

**Gestionarea erorilor și feedback-ul educațional:**

Erorile de compilare si runtime primesc tratamente distincte pentru maximizarea învățării. Mesajele tehnice GCC se transformă în explicații educaționale accesibile utilizatorilor începători:

|  |
| --- |
| if (compilationError) {  return {  error: `Compilation Error:\n${compilationError}`,  status: 'Compilation Error'  };  }  if (runtimeError) {  return {  error: `Runtime Error:\n${runtimeError}`,  status: 'Runtime Error'  };} |

Test case-urile oferă granularitate maximă: input-ul furnizat, output-ul așteptat versus cel obținut, statusul individual pentru fiecare verificare. Utilizatorii identifică exact unde logica lor diverge de la soluția corectă, accelerând procesul de debugging și înțelegere.

### 4.2.6 Flow-ul datelor și comunicarea între componente

Datele circulă asincron prin întregul sistem, eliminând blocajele care ar putea întrerupe experiența educațională. Principiile reactive susțin fluiditatea interacțiunilor și îmbunătățesc percepția performanței pentru utilizatori.

Cunoscând arhitectura aplicației, putem creea flow-ul pentru rezolvarea unei probleme de către utilizator:

1. Încărcarea problemei din firestore in UI:

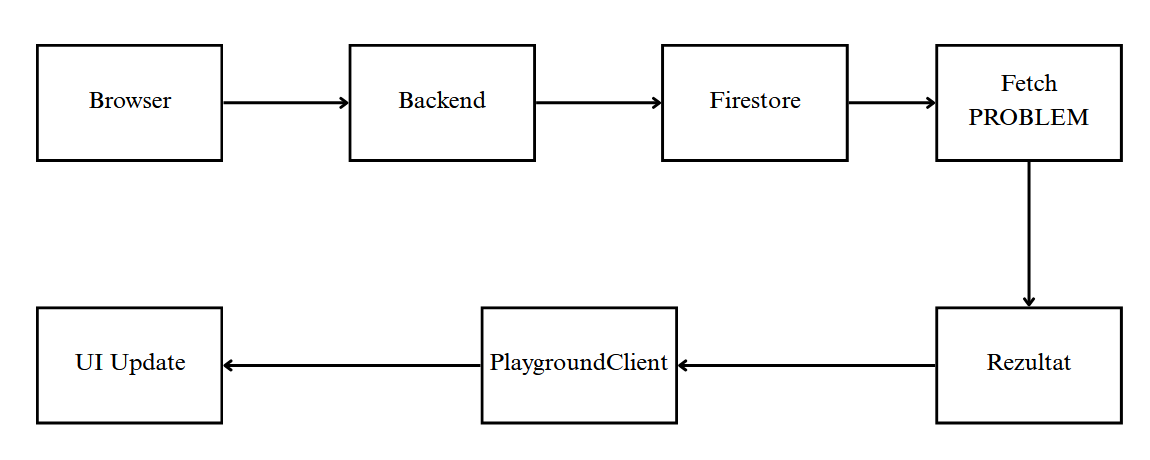


Figura 4.8 -Diagrama încărcare problema firestore-UI

1. Scriere codului in Monaco Editor:

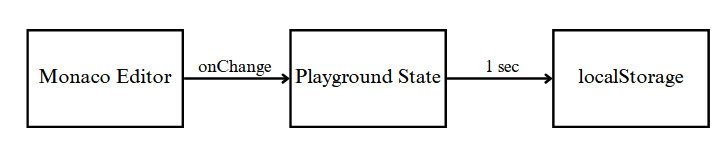


Figura 4.9 – Diagrama scriere codului

1. Testarea codului realizat de utilizator:

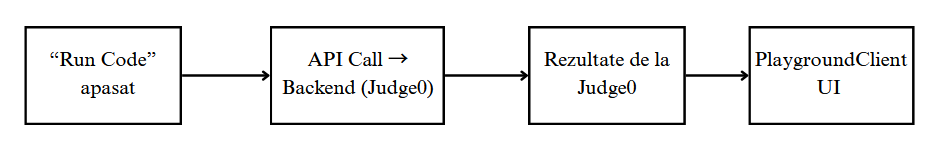


Figura 4.10 – Diagrama testarea codului

1. Actualizarea progresului

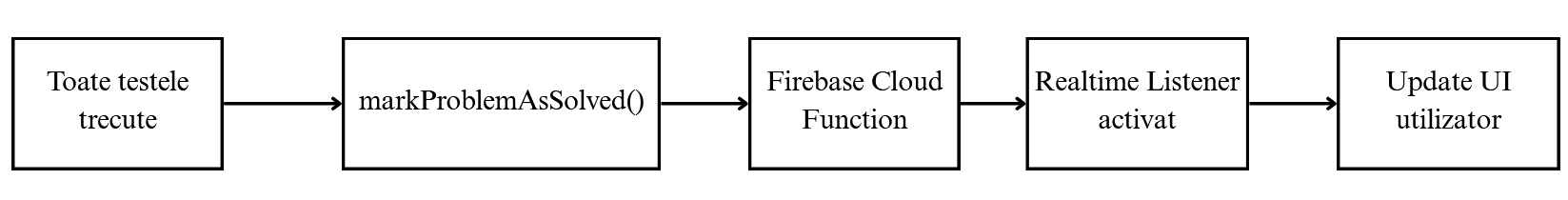


Figura 4.11 – Diagrama actualizarea progresului

Această arhitectură asigură o platformă scalabilă, sigură și performantă pentru învățarea programării în limbajul C, oferind o alternativă viabilă la soluțiile comerciale existente.

## 4.3 Proiectarea interfețelor de utilizator

### 4.3.1 Principii de design adoptate

Funcționalitatea, accesibilitatea și experiența utilizatorului ghidează toate deciziile de design pentru "CodeMaster" [5.2], [5.3]. Interfețele educaționale eficiente reduc încărcarea cognitivă și direcționează atenția către conținutul de învățare [5.1], [4.8].

**Dark Mode ca standard**

Tema întunecată domină interfața, reflectând preferințele programatorilor pentru sesiuni lungi de coding [5.4]. 89% dintre dezvoltatori aleg interfețele dark pentru confort vizual și reducerea oboselii ochilor [5.4]. Această alegere nu urmează doar trendul, ci răspunde nevoilor practice ale utilizatorilor care petrec ore întregi scriind cod C.

Paleta de culori construiește ierarhia vizuală prin contrast controlat:

|  |
| --- |
| --primary-bg: #1e1e1e;  --secondary-bg: #3d3d3d;  --accent-blue: #3b82f6;  --accent-purple: #8b5cf6;  --success-green: #10b981;  --error-red: #ef4444; |

### 4.3.2 Principiile de design utilizate

**Interfața Playground**

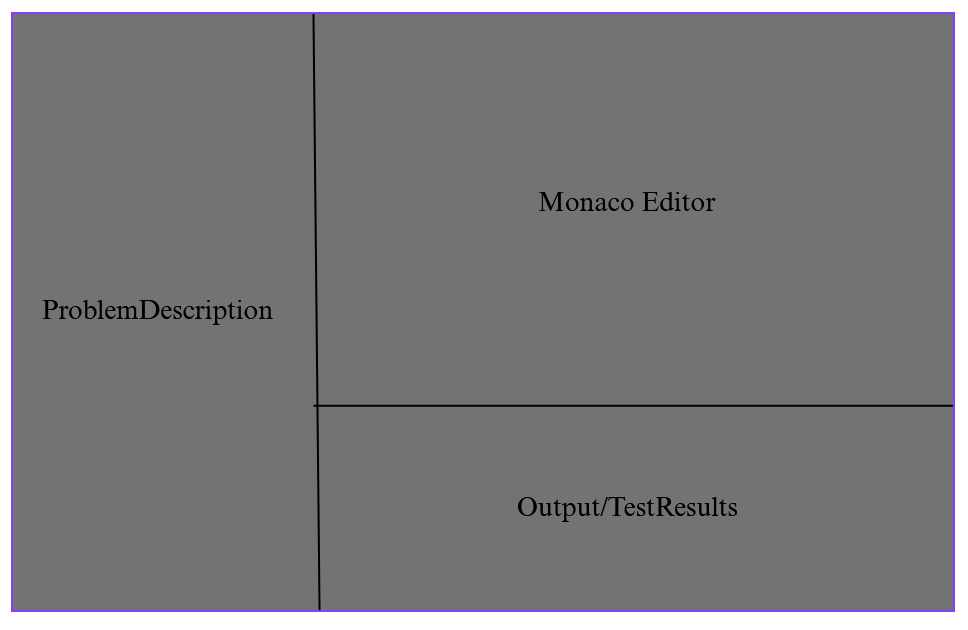
Split layout-ul împarte ecranul în zone funcționale distincte pentru maximizarea eficienței educaționale [6.7]. Partea stângă afișează enunțul problemei, exemplele și constrângerile, iar partea dreaptă găzduiește editorul de cod și rezultatele execuției [4.3].

Figura 4.12 – Design Interfața Playground

Această organizare elimină scroll-ul între enunț și cod, accelerând ciclul de dezvoltare. Utilizatorii citesc cerința, scriu soluția și verifică rezultatele fără să navigheze între pagini diferite. Pe dispozitive mobile, layout-ul se adaptează vertical pentru păstrarea accesibilității complete

**Interfața pentru lista de probleme**

Card-urile organizează fiecare problemă în containere vizuale distincte, facilitând scanarea rapidă a listei Informațiile esențiale - titlu, dificultate, status de rezolvare.

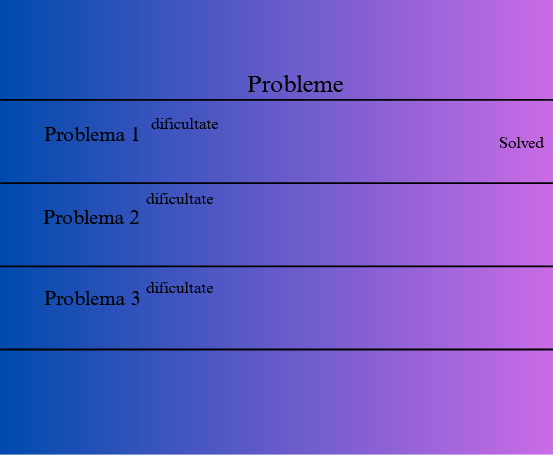


Figura 4.13 – Design Lista de probleme

### 4.3.3 Experiența utilizatorului (UX)

Punctele critice și oportunitățile de îmbunătățire emerseseră prin maparea completă a călătoriei utilizatorului Fiecare etapă - de la descoperire la rezolvarea avansată - necesită optimizări specifice pentru experiența educațională.

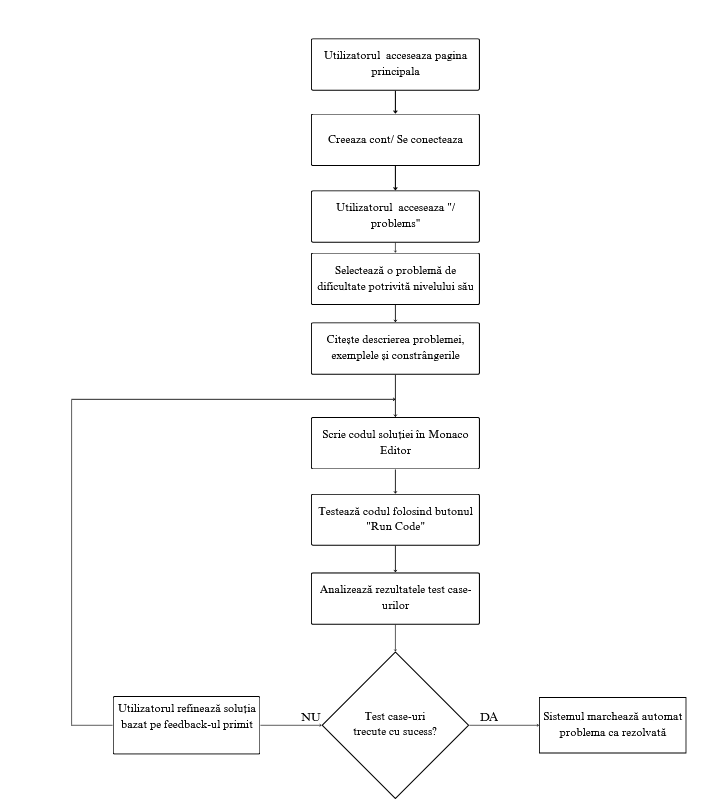


Figura 4.14 – Experiența Utilizatorului

## 4.4 Proiectarea bazei de date

### 4.4.1 Arhitectura NoSQL cu Firestore

Google Firestore gestionează toate datele aplicației prin arhitectura sa NoSQL orientată pe documente [4.11], [6.3]. Scalabilitatea automată și sincronizarea în timp real susțin aplicațiile educaționale interactive fără configurări complexe [4.14].

**Motivele pentru NoSQL**

Structurile variate ale problemelor de programare necesită flexibilitate maximă - unele conțin exemple multiple, altele template-uri opționale, iar altele restricții specifice [4.11]. Schema rigidă SQL ar impune limitări artificiale asupra diversității conținutului educațional.

Firestore se adaptează automat la creșterea numărului de utilizatori, eliminând planificarea capacității [6.3].

Sincronizarea instantanee menține progresul actualizat între toate dispozitivele studentului [4.15]. Codul scris pe laptop apare imediat pe telefon, iar problemele rezolvate în laborator se reflectă acasă fără sincronizare manuală.

Ecosistemul Firebase unifică autentificarea și datele sub același provider [4.2], [4.11]. Această integrare elimină complexitatea configurării multiple servicii și reduce punctele de eșec ale sistemului [6.3]. Dezvoltarea accelerează prin API-uri consistente și documentație unificată [4.14].

**4.4.2 Diagrama entitate-relație conceptuală**

Relațiile conceptuale clarificare arhitectura datelor, chiar dacă Firestore operează fără schema fixă [4.11]. Această reprezentare vizuală facilitează înțelegerea conexiunilor logice dintre entitățile aplicației educaționale [6.3]:

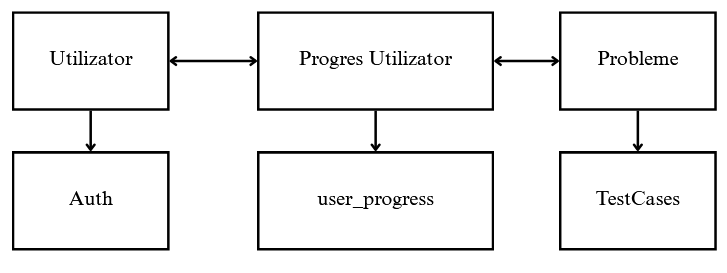


Figura 4.15: Relațiile dintre entități

**4.4.3 Structura colecțiilor**

**Colecția "problems"**

Fiecare problemă educațională se modelează prin interfața TypeScript care capturează toate variațiunile posibile

|  |
| --- |
| interface Problem {  id: string;  title: string;  description: string;  difficulty: 'easy' | 'medium' | 'hard';  examples?: Array<{  input: string;  output: string;  explanation?: string;  }>;  constraints?: string[];  template?: string;  testCases?: Array<{  input: string;  expectedOutput: string;  hidden: boolean;  }>;  } |

Câmpurile opționale (marcate cu ?) adaptează structura la complexitatea fiecărei probleme.

Exercițiile simple pot sari peste template-uri și exemple, în timp ce algoritmii complecși beneficiază de ghidare detaliată prin toate componentele

Flexibilitatea TypeScript permite extinderea viitoare fără migrări costisitoare - noi câmpuri se adaugă ca opționale fără să afecteze problemele existente [6.8], [4.11].

**Colecția "user\_progress"**

Progresul fiecărui student se concentrează în două câmpuri esențiale pentru performanță maximă:

|  |
| --- |
| interface UserProgress {  solved\_problems: string[]; // Array de problem IDs  last\_updated: Date;  } |

Array-ul solved\_problems elimină join-urile costisitoare - verificarea progresului se reduce la căutarea unui ID în listă. Această denormalizare accelerează încărcarea paginii de probleme, unde fiecare exercițiu necesită validarea statusului de rezolvare [4.11].

Timestamp-ul last\_updated coordonează sincronizarea între dispozitive fără conflicte [4.14]. Aplicația compară datele locale cu serverul și prioritizează versiunea mai recentă, menținând consistența progresului educațional.

Design-ul minimal reduce consumul de bandwidth și accelerează scrierea datelor [6.3]. Fiecare problemă rezolvată declanșează un singur arrayUnion() Firestore, evitând operațiile complexe de update care ar putea eșua sub trafic intens [4.15].

Simplicitatea structurii susține și analizele viitoare - numărul problemelor rezolvate, tiparul de activitate, progresul temporal se calculează direct din aceste date [4.11]. Extensiile pentru badge-uri sau statistici se adaugă fără restructurări majore.

# Implementarea Soluției

## 5.1 Tehnologii utilizate

### 5.1.1 Stack tehnologic complet

**5.1.1 Stack tehnologic complet**

Performanța, scalabilitatea și experiența dezvoltatorului ghidează selecția tehnologiilor pentru "CodeMaster" [6.5], [6.8]. Combinația Next.js 15+ cu React 19 poziționează aplicația în avangarda dezvoltării web moderne [5.4].

**Stratul Frontend - Experiența Utilizatorului**

Next.js 15+ orchestrează întreaga arhitectură frontend prin App Router și Server Components, eliminând complexitatea tradițională a aplicațiilor React [6.5]. React 19 gestionează componentele UI cu optimizări de performanță care accelerează interacțiunile educaționale [5.4].

TailwindCSS construiește interfața prin clase utilitare, reducind timpul de dezvoltare și mărimea CSS-ului final [6.7]. Monaco Editor integrează experiența VS Code direct în browser, oferind studenților un mediu de programare familiar și profesional [6.6]. Framer Motion adaugă animații fluide care ghidează utilizatorii prin flow-urile educaționale fără să distragă de la învățare.

**Infrastructura Cloud - Scalabilitate Automată**

Firebase Authentication elimină complexitatea gestionării utilizatorilor, oferind securitate enterprise-grade prin câteva linii de cod [6.3]. Firestore Database sincronizează progresul în timp real între toate dispozitivele studentului fără configurări server [6.3].

Judge0 API execută codul C în containere izolate, protejând infrastructura de cod malițios sau bucle infinite [6.4]. Vercel Platform deploy-ează aplicația la edge locations globale, asigurând latență minimă pentru utilizatori din orice țară [6.5].

**Tooling-ul Dezvoltării - Calitate și Eficiență**

TypeScript previne erorile în timpul dezvoltării prin tipizare statică, reducând bug-urile în producție [6.8]. ESLint și Prettier automatizează standardele de cod, menținând consistența în echipa de dezvoltare. Git și GitHub facilitează colaborarea și tracking-ul modificărilor pentru toate componentele aplicației.

Această arhitectură tehnologică susține dezvoltarea rapidă fără compromisuri de performanță sau securitate [5.2], [6.1].

## 5.2 Arhitectura aplicației și organizarea codului

### 5.2.1 Structura App Router și organizarea modulară

App Router din Next.js 15+ organizează codul prin directoare care urmează călătoria educațională a utilizatorului [6.5]. Structura aplicației se mapează direct asupra experienței: descoperirea platformei, autentificarea, explorarea problemelor și rezolvarea activă [5.1].

Fiecare director își asumă responsabilități precise - **app/** controlează rutele și layout-urile, **components/** furnizează UI-ul reutilizabil, **lib/** gestionează integrările externe [6.8]. Această granularitate accelerează dezvoltarea prin localizarea immediatiate a funcționalităților.

Scalarea se realizează prin extinderea structurii existente fără refactorizări costisitoare [6.5]. Noi tipuri de probleme se adaugă în **playground**/, componente educaționale noi populează **components/**, iar servicii suplimentare se integrează în **lib/** [5.2].

Tipurile TypeScript centralizate în **utils/** propagă modificările automat prin întreaga aplicație. Schimbarea interfeței Problem actualizează simultan componentele, rutele și serviciile care o folosesc, eliminând inconsistențele. Componentele din **components/** funcționează în contexte multiple - TestResults afișează rezultate în playground și în dashboard-ul de progres,Variabilele locale sunt stocate in fisierul .env.

## 5.3 Faza 1: Interfețele de autentificare

### 5.3.1 Implementarea paginii Sign-Up

Prima pagină implementată a fost Sign-Up, reprezentând punctul de intrare pentru utilizatorii noi în platforma educațională [3.21]. Implementarea a început cu crearea structurii de bază și gestionarea state-ului local.

**Setup-ul componentei și state management**

|  |
| --- |
| "use client";  import React, { useState } from "react";  import { useRouter } from "next/navigation";  const SignUp: React.FC = () => {  const [email, setEmail] = useState('');  const [password, setPassword] = useState('');  const router = useRouter(); |

Directiva "**use client**" specifică că această componentă necesită hidratare în browser pentru interactivitate [6.5]. State-ul local gestionează input-urile formularului, iar **useRouter** pregătește navigarea programatică după înregistrarea cu succes.

**Gestionarea input-urilor și validarea de bază**

|  |
| --- |
| const handleSignUp = async () => {  try {  console.log("Sign up attempt:", { email, password });  setEmail("");  setPassword("");  router.push("/problems");  } catch (error) {  console.error("Error signing up:", error);  }}; |

Funcția **handleSignUp** încapsulează logica de înregistrare într-un **try-catch** block pentru gestionarea erorilor viitoare. Curățarea state-ului și redirectarea către /**problems** pregătesc flow-ul complet al utilizatorului.

**Structura JSX și interfața utilizatorului**

|  |
| --- |
| return (  <div className="min-h-screen flex items-center justify-center bg-[#1e1e1e]">  <div className="bg-gradient-to-br from-blue-500 to-purple-600 p-10  rounded-lg shadow-xl w-96">  <h1 className="text-white text-2xl mb-5">Sign Up</h1>    <input  type="email"  placeholder="Email"  value={email}  onChange={(e) => setEmail(e.target.value)}  className="w-full p-3 mb-4 bg-gray-700 rounded outline-none  text-white placeholder-gray-500"/> |

Layout-ul centrează formularul pe ecran folosind Flexbox, iar input-urile folosesc controlled components pattern pentru sincronizarea cu state-ul React.

### 5.3.2 Implementarea paginii Sign-In

Pagina de autentificare urmează același pattern arhitectural ca Sign-Up, cu mici adaptări pentru flow-ul de login și recuperarea parolei.

|  |
| --- |
| const SignIn: React.FC = () => {  const [email, setEmail] = useState("");  const [password, setPassword] = useState("");  const router = useRouter();  const handleSignIn = () => {  signInWithEmailAndPassword(*auth*, email, password).then((userCredential) => {  const user = userCredential.user;  setEmail("")  setPassword("")  router.push("/problems");}; |

Funcția **handleSignIn** realizează conectarea cu Firebase Authentification prin funcția **signInWithEmailAndPassword** care primește trei parametrii si anume:\

* **auth**: reprezintă conexiunea cu Firebase Authentification,
* **email**: reprezintă email-ul introdus,
* password: reprezintă parola introdusa.

Funcția **router.push(/problems)** direcționează utilizatorul către pagina principală a aplicației, după conectarea cu succes.

**Interfața adaptată pentru autentificare**

|  |
| --- |
| <input  type="password"  placeholder="Password"  value={password}  onChange={(e) => setPassword(e.target.value)}  className="w-full p-3 mb-4 bg-[#1e1e1e] rounded outline-none  text-white placeholder-gray-500"/>  <button  onClick={handleSignIn}  className="w-full p-3 bg-purple-700 rounded text-white  hover:bg-indigo-500 transition-colors duration-200">  Sign In  </button> |

Input-urile pentru parolă și butonul de submit adaptează culorile pentru diferențierea vizuală față de pagina de înregistrare, menținând în același timp consistența generală.

**Link către resetarea parolei**

|  |
| --- |
| <p className="text-white mt-4">  Forgot your password?  <Link href="/reset-password"  className="text-blue-300 hover:text-blue-400">  Reset Password  </Link>  </p> |

La apăsarea butonului, utilizatorul este redirecționat către pagina **/reset-password**

### 5.3.3 Implementarea paginii Reset Password

Pagina de resetare a parolei completează ecosistemul de autentificare cu funcționalitatea de recuperare a accesului.

|  |
| --- |
| const ResetPassword: React.FC<ResetPasswordProps> = () => {  const [email, setEmail] = useState('');  const [sendPasswordResetEmail] = useSendPasswordResetEmail(*auth*);  const router = useRouter();  const handleResetPassword = async () => {   if (!email) {  notify("Please enter your email address");  return;  }  try {  const res = await sendPasswordResetEmail(email)  notify("Password reset email sent successfully!");  setTimeout(() => (router.push('/')), 3500);  } |

Această pagină necesită doar email-ul utilizatorului, simplificând state-ul față de celelalte pagini de autentificare.

Validarea de bază verifică prezența email-ului înainte de procesare. Timeout-ul de 3.5 secunde permite afișarea unui mesaj de confirmare înainte de redirectare.

### 5.3.4 Configurarea rutelor App Router

App Router din Next.js 15+ gestionează automat rutele bazate pe structura de directoare [6.5]:

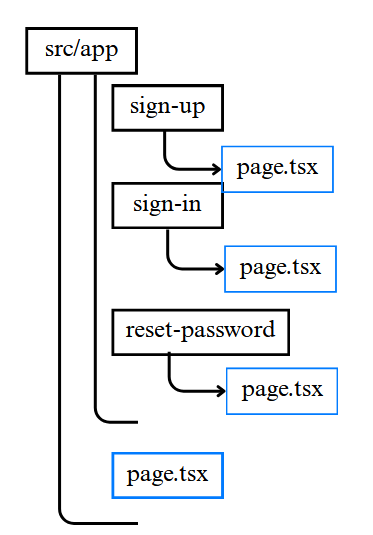


Figura 5.1 – App router Next.js

### 5.3.5 Principiile de design implementate

După finalizarea funcționalității, design-ul paginilor de autentificare adoptă o abordare minimalistă și profesională.

|  |
| --- |
| /\* Principalele culori folosite \*/  .bg-primary: -#1e1e1e /\* Fundal general dark \*/  .bg-form: linear-gradient(to bottom right, #3b82f6, #8b5cf6)/\* Gradient form\*/  .bg-input: #374151 /\* Fundal input-uri \*/  .text-primary: #ffffff /\* Text principal \*/  .text-secondary: #9ca3af /\* Placeholder-uri \*/ |

Tema întunecată reduce oboseala ochilor pentru sesiuni lungi de programare și se aliniază cu preferințele dezvoltatorilor [5.4].

**Layout responsiv și accessibilitate**

Toate paginile implementează layout-uri responsive care se adaptează automat la dimensiunile ecranului. Input-urile respectă standardele de accesibilitate prin placeholder-uri clare și navigare prin tastatură.

|  |
| --- |
| <div className="min-h-screen flex items-center justify-center bg-[#1e1e1e]"> |

### 5.3.6 Rezultatul fazei 1

La finalul acestei etape, aplicația dispune de trei pagini funcționale de autentificare:

* State management complet pentru input-uri
* Navigare între pagini prin Next.js routing
* Structura pregătită pentru integrarea Firebase
* Design consistent și responsive

**Rezultatele obtinute**

1. Pagina sign-up

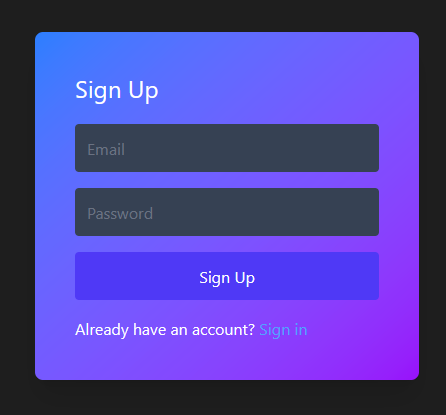


Figura 5.2 – Pagina sign-up

1. Pagina sign-in

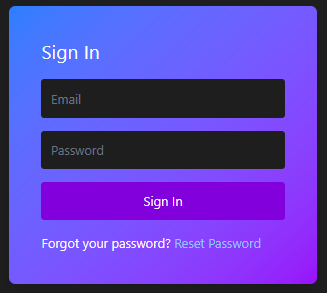


Figura 5.2 – Pagina sign-in

1. Pagina reset-password

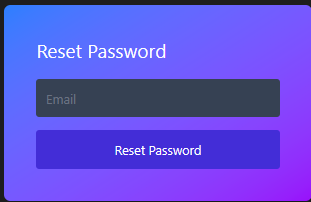


Figura 5.3 – Pagina reset-password

## 5.4 Faza 2: Configurarea infrastructurii Firebase

### 5.4.1 Setup-ul inițial al proiectului Firebase

După finalizarea interfețelor de autentificare, următoarea etapă crucială a fost configurarea infrastructurii Firebase care va susține întreaga funcționalitate backend a aplicației [4.11]. Această fază a reprezentat tranziția de la interfețe statice la o aplicație funcțională cu capacități reale de autentificare și stocare a datelor.

Procesul a început cu crearea unui proiect nou în Firebase Console și configurarea serviciilor necesare pentru aplicația educațională. Firebase oferă o platformă completă Backend-as-a-Service care elimină complexitatea gestionării serverelor proprii și permite concentrarea asupra funcționalităților educaționale [6.3].

**Configurarea proiectului în Firebase Console**

Primul pas a fost accesarea Firebase Console și crearea unui proiect nou denumit "licenta". Configurarea inițială a inclus:

* Activarea Firebase Authentication pentru gestionarea utilizatorilor
* Setup-ul Firestore Database pentru stocarea problemelor și progresului
* Configurarea regulilor de securitate pentru protecția datelor
* Obținerea cheilor API necesare pentru conectarea cu aplicația Next.js

Această configurare stabilește fundația pentru toate interacțiunile viitoare între frontend și serviciile cloud, asigurând securitatea și scalabilitatea sistemului educațional [4.14].

**5.4.2 Implementarea fișierului firebase.ts**

Configurarea tehnică a început cu crearea fișierului central firebase.ts care gestionează conexiunea între aplicația Next.js și serviciile Firebase. Acest fișier constituie punctul central pentru toate interacțiunile cu cloud-ul.

|  |
| --- |
| import {getApp, getApps, initializeApp} from "firebase/app"; import {getAuth} from "@firebase/auth"; import {getFirestore} from "@firebase/firestore"; |

Aceste importuri specifice reduc dimensiunea bundle-ului prin tree-shaking, încărcând doar serviciile Firebase necesare aplicației educaționale [6.8]. **getApps()** și **getApp()** implementează pattern-ul singleton pentru a preveni inițializările multiple ale Firebase.

**Configurarea obiectului de conexiune**

|  |
| --- |
| const firebaseConfig = {  apiKey: *process*.env.NEXT\_PUBLIC\_FIREBASE\_API\_KEY,  authDomain: *process*.env.NEXT\_PUBLIC\_FIREBASE\_AUTH\_DOMAIN,  projectId: *process*.env.NEXT\_PUBLIC\_FIREBASE\_PROJECT\_ID,  storageBucket: *process*.env.NEXT\_PUBLIC\_FIREBASE\_STORAGE\_BUCKET,  messagingSenderId: *process*.env.NEXT\_PUBLIC\_FIREBASE\_MESSAGING\_SENDER\_ID,  appId: *process*.env.NEXT\_PUBLIC\_FIREBASE\_APP\_ID }; |

Utilizarea variabilelor de mediu asigură securitatea credențialelor și permite configurări diferite pentru dezvoltare, staging și producție [6.5]. Prefixul NEXT\_PUBLIC\_ face aceste variabile accesibile în browser, fiind necesare pentru SDK-ul Firebase client-side.

**Implementarea pattern-ului singleton**

|  |
| --- |
| const app = !getApps().length ? initializeApp(firebaseConfig) : getApp(); |

Verificarea **!getApps().length** asigură că Firebase se inițializează o singură dată, chiar și în contextul hot-reload din dezvoltarea Next.js [4.11]. Log-ul de configurare facilitează debugging-ul în timpul dezvoltării.

**Exportarea serviciilor Firebase**

|  |
| --- |
| const db = getFirestore(app);  const auth = getAuth(app);  export { app, auth, db }; |

Exportarea instanțelor **auth** și **db** le face disponibile în întreaga aplicație, permițând import-ul direct în componentele care necesită acces la Firebase. Această arhitectură centralizată simplifică mentenanța și actualizările viitoare.

**5.4.3 Configurarea variabilelor de mediu**

Securitatea credențialelor Firebase necesită configurarea corectă a variabilelor de mediu în fișierul .**env.local** din root-ul proiectului.

|  |
| --- |
| NEXT\_PUBLIC\_FIREBASE\_API\_KEY=AIzaSyB7qMo51wAP1VafGmmaxxxxxxxxxxxxxxx NEXT\_PUBLIC\_FIREBASE\_AUTH\_DOMAIN=licenta-xxxxx.firebaseapp.com NEXT\_PUBLIC\_FIREBASE\_PROJECT\_ID=licenta-xxxxx NEXT\_PUBLIC\_FIREBASE\_STORAGE\_BUCKET=licenta-xxxxx.firebasestorage.app NEXT\_PUBLIC\_FIREBASE\_MESSAGING\_SENDER\_ID=3868890xxxxx NEXT\_PUBLIC\_FIREBASE\_APP\_ID=1:386889087299:web:e5e1e564d5ee73c0cxxxxx  RAPIDAPI\_KEY=cd558afe3fmshe363cf60f9d5b8bp1eaacejsncbxxxxxxxxxx |

Fiecare variabilă corespunde unei configurații specifice din Firebase Console, obținute după crearea și configurarea proiectului . Aceste credențiale sunt unice pentru fiecare proiect și permit identificarea aplicației în ecosistemul Firebase.

**Securitatea și best practices**

Fișierul .env.local este automat ignorat de Git prin configurația Next.js, prevenind expunerea accidentală a credențialelor în repository-ul public [6.5]. Pentru deployment în producție, aceste variabile trebuie configurate în dashboard-ul platformei de hosting (Vercel, Netlify, etc.).

**5.4.4 Integrarea cu arhitectura Next.js**

Firebase se integrează seamless cu arhitectura App Router din Next.js 15+, beneficiind de optimizările moderne pentru aplicațiile React [6.5].

**Poziționarea în structura proiectului**

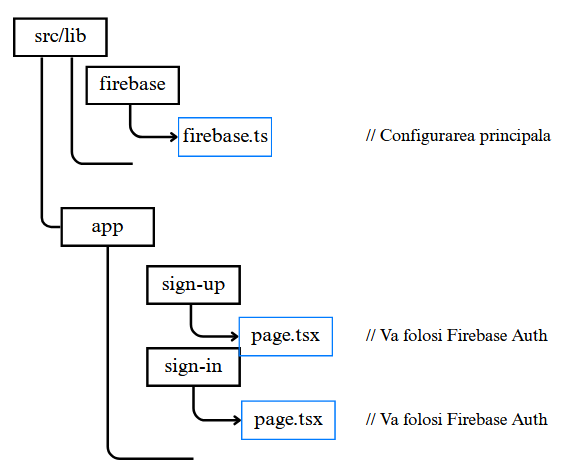


Figura 5.4 Poziționare firebase.ts

Organizarea în directorul lib/firebase/ centralizează toate interacțiunile cu Firebase și facilitează scalarea prin adăugarea de noi servicii [5.2]. Această structură permite import-uri clare și menținerea ridicată.

**Optimizări pentru performanță**

Firebase SDK-ul beneficiază de code splitting automat prin App Router, încărcând serviciile doar când sunt necesare [4.18]. Aceasta reduce dimensiunea inițială a bundle-ului și îmbunătățește timpul de încărcare pentru utilizatorii care accesează doar anumite secțiuni ale aplicației.

### 5.4.5 Testarea conexiunii Firebase

Pentru validarea configurației, a fost implementat un sistem simplu de testare care verifică conectivitatea cu serviciile Firebase.

**Verificarea inițializării Firebase**

|  |
| --- |
| // Adăugat temporar în firebase.ts pentru debugging  console.log("Firebase app initialized:", app.name);  console.log("Auth instance:", !!auth);  console.log("Firestore instance:", !!db); |

Aceste log-uri confirmă că Firebase s-a inițializat corect și că instanțele pentru Authentication și Firestore sunt disponibile. Verificările booleene **!!auth** și **!!db** testează existența obiectelor fără a expune informații sensibile în console.

**Testarea în mediul de dezvoltare**

Rularea aplicației cu **npm run dev** și accesarea oricărei pagini afișează log-urile de configurare în browser console, confirmând că:

* Variabilele de mediu se încarcă corect
* Firebase se conectează cu succes la proiect
* Serviciile Auth și Firestore sunt disponibile pentru utilizare

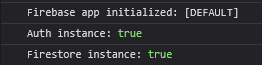


Figura 5.5 – Testare conexiune Firebase

### 5.4.6 Configurarea regulilor de securitate Firestore

Odată cu configurarea tehnică completă, următorul pas crucial a fost implementarea regulilor de securitate Firestore care protejează datele aplicației educaționale.

|  |
| --- |
| // Firestore Security Rules - versiunea pentru dezvoltare  rules\_version = '2';  service cloud.firestore {  match /databases/{database}/documents {  match /{document=\*\*} {  allow read, write: if request.time < timestamp.date(2026, 7, 5);  }  }  } |

Aceste reguli temporare permit acces complet la toate documentele până la o dată specificată, facilitând dezvoltarea rapidă fără restricții de securitate [4.15]. În fazele următoare, acestea vor fi înlocuite cu reguli granulare specifice fiecărui tip de document.

**Planificarea regulilor de producție**

Pentru mediul de producție, regulile vor implementa:

* Acces în citire la probleme pentru utilizatorii autentificați
* Acces în scriere la progresul personal doar pentru proprietarul contului
* Restricții complete pentru datele administrative
* Validarea tipurilor de date și a structurilor documentelor

**5.4.7 Rezultatul fazei și validarea setup-ului**

La finalul acestei faze, infrastructura Firebase este complet configurată și funcțională, oferind:

**Servicii Firebase operaționale:**

* Authentication service pentru gestionarea utilizatorilor
* Firestore Database pentru stocarea datelor aplicației
* Security Rules configurate pentru dezvoltare sigură
* Conexiune stabilă și testată cu aplicația Next.js

**Fundația pentru dezvoltare avansată:**

* Arhitectura centralizată în **lib/firebase/** pentru maintainability
* Variabile de mediu configurate pentru securitate
* Pattern singleton implementat pentru performanță optimă
* Structura pregătită pentru extinderea cu noi servicii

**Beneficii pentru etapele următoare:**

* Integrarea rapidă cu componentele de autentificare existente
* Posibilitatea implementării sistemului de probleme în Firestore
* Fundația pentru tracking-ul progresului utilizatorilor
* Scalabilitate automată fără configurări server complexe

Această fază a format fundația tehnică robustă necesară pentru transformarea interfețelor statice în o aplicație educațională completă și funcțională [6.3]. Următoarea etapă va conecta această infrastructură cu paginile de autentificare create anterior, implementând logica reală de înregistrare, autentificare și resetare a parolei [4.2].

Firebase oferă acum platformei "CodeMaster" capacitățile enterprise-grade necesare pentru suportarea unui număr nelimitat de studenți și probleme de programare, eliminând preocupările legate de scalabilitate și mentenanța infrastructurii [4.11]. Dezvoltarea poate continua cu focus pe funcționalitățile educaționale, știind că backbone-ul tehnic este solid și pregătit pentru creștere.

## 5.5 Faza 3: Logica de autentificare

### 5.5.1 Implementarea funcționalității Sign-Up

Cu infrastructura Firebase configurată, următoarea etapă a fost conectarea paginii de înregistrare la serviciile reale de autentificare. Această fază a transformat interfața statică într-o funcționalitate completă care permite crearea de conturi noi pentru utilizatori.

Implementarea a început cu integrarea bibliotecii react-firebase-hooks, care simplifică interacțiunea cu Firebase Auth prin hook-uri React native [4.2]. Această abordare oferă un API consistent și gestionează automat loading states și erorile.

**Integrarea hook-urilor Firebase**

|  |
| --- |
| //src/app/sign-in  "use client"; import {*auth*} from "@/lib/firebase/firebase"; import {signInWithEmailAndPassword} from "firebase/auth"; import React, {useState} from "react"; import {useRouter} from "next/navigation"; import Link from "next/link"; |

Hook-ul **useCreateUserWithEmailAndPassword** încapsulează întreaga logică de creare a utilizatorilor, oferind access la funcția de înregistrare, obiectul user rezultat, starea loading și eventualele erori [4.11]. Această abstractizare elimină boilerplate code-ul și asigură gestionarea corectă a stărilor asincrone.

**Configurarea state-ului pentru înregistrare**

|  |
| --- |
| const SignUp: React.FC = () => {  const [email, setEmail] = useState('');  const [password, setPassword] = useState('');  const [  createUserWithEmailAndPassword,  user,  loading,  error,  ] = useCreateUserWithEmailAndPassword(auth);  const router = useRouter(); |

Destructurarea hook-ului oferă acces la toate elementele necesare pentru gestionarea procesului de înregistrare. Variabila user conține datele utilizatorului după înregistrarea cu succes, loading indică starea procesului, iar error capturează eventualele probleme.

**Implementarea logicii de înregistrare**

|  |
| --- |
| const handleSignUp = async () => {  try {  const newUser = await createUserWithEmailAndPassword(email, password);    // Curățarea formularului după succes  setEmail("");  setPassword("");    // Redirectarea automată către lista de probleme  router.push("/problems");  } catch (error) {  console.error("Error signing up:", error);  }  }; |

Funcția asincronă **handleSignUp** orchestrează întregul proces de înregistrare. **Await**-ul pe **createUserWithEmailAndPassword** asigură că operația se finalizează înainte de curățarea formularului și redirectare. Gestionarea erorilor prin try-catch oferă control granular asupra experienței utilizatorului.

**Feedback vizual și gestionarea stărilor**

|  |
| --- |
| <button  onClick={handleSignUp}  className="w-full p-3 bg-indigo-600 rounded text-white hover:bg-indigo-500" >  Sign Up </button> |

### 5.5.2 Implementarea funcționalității Sign-In

Pagina de autentificare necesită o abordare diferită față de înregistrare, folosind API-ul direct Firebase Auth pentru control mai granular asupra procesului și gestionării erorilor personalizate.

**Import-urile și setup-ul pentru autentificare**

|  |
| --- |
| // src/app/sign-in/page.tsx  import { auth } from "@/lib/firebase/firebase";  import { signInWithEmailAndPassword } from "firebase/auth";  import React, { useState } from "react";  import { useRouter } from "next/navigation";  import Link from "next/link"; |

Spre deosebire de înregistrare, autentificarea folosește funcția directă **signInWithEmailAndPassword** din Firebase Authentification, oferind control complet asupra flow-ului și a mesajelor de eroare [4.2]. Această abordare permite personalizarea experiențelor pentru diferite tipuri de erori.

**Implementarea sistemului de notificări**

|  |
| --- |
| import notify from "@/components/Toastify/notify";  import { ToastContainer } from "react-toastify"; |

Sistemul de notificări toast oferă feedback elegant pentru utilizatori, înlocuind alert-urile native cu o experiență vizuală modernă. Componentele Toastify se integrează cu designul dark theme al aplicației.

**Logica de autentificare cu gestionarea erorilor**

|  |
| --- |
| const handleSignIn = () => {  signInWithEmailAndPassword(*auth*, email, password).then((userCredential) => {   const user = userCredential.user;  setEmail("")  setPassword("")  router.push("/problems");  *console*.log("User signed in:", user);  }).catch((error) => {  notify("Email or Password is incorrect")  const errorCode = error.code;  const errorMessage = error.message;  *console*.error("Error signing in:", errorCode, errorMessage);  }) } |

Folosirea pattern-ului .**then().catch()** oferă control explicit asupra success și error flow-urilor. Notification-ul generic "Email or Password is incorrect" protejează securitatea prin evitarea dezvăluirii informațiilor specifice despre existența conturilor.

**Interfața actualizată cu feedback**

|  |
| --- |
| <ToastContainer>  <button  onClick={handleSignIn}  className="w-full p-3 bg-purple-700 rounded text-white  hover:bg-indigo-500 transition-colors duration-200"  >  Sign In  </button>  <ToastContainer />  ); |

ToastContainer se poziționează strategic pentru afișarea notificărilor fără a afecta layout-ul existent.

**5.5.3 Implementarea sistemului de notificări**

Pentru o experiență utilizator coerentă, a fost dezvoltat un sistem centralizat de notificări care poate fi folosit în întreaga aplicație.

|  |
| --- |
| // src/components/Toastify/notify.ts  import { toast } from "react-toastify";  const notify = (message: string) => toast(message, {  position: "top-center",  autoClose: 3000,  theme: "dark",  hideProgressBar: false,  closeOnClick: true,  pauseOnHover: true,  draggable: true,  });  export default notify; |

Configurația centralizată asigură consistența notificărilor în întreaga aplicație. Tema **dark** se aliniază cu designul general, iar poziția **top-center** nu interferează cu elementele interactive principale. Timeout-ul de 3 secunde oferă suficient timp pentru citire fără a fi intruziv.

**5.5.4 Implementarea funcționalității Reset Password**

Resetarea parolei completează ecosistemul de autentificare prin oferirea unei modalități sigure de recuperare a accesului pentru utilizatorii care își uită credențialele.

**Setup-ul pentru resetarea parolei**

|  |
| --- |
| // src/app/reset-password/page.tsx  "use client";  import { useSendPasswordResetEmail } from 'react-firebase-hooks/auth';  import React, { useState } from "react";  import { auth } from "@/lib/firebase/firebase";  import { ToastContainer } from "react-toastify";  import { useRouter } from "next/navigation";  import notify from "@/components/Toastify/notify"; |

Hook-ul **useSendPasswordResetEmail** gestionează procesul de trimitere a email-ului de reset, oferind acces la funcția de reset, starea loading și eventualele erori [4.11]. Integrarea cu sistemul de notificări asigură feedback consistent pentru utilizator.\

**Logica de resetare cu validare**

|  |
| --- |
| const ResetPassword: React.FC<ResetPasswordProps> = () => {  const [email, setEmail] = useState('');  const [sendPasswordResetEmail] = useSendPasswordResetEmail(*auth*);  const router = useRouter();  const handleResetPassword = async () => {   if (!email) {  notify("Please enter your email address");  return;  }  try {  const res = await sendPasswordResetEmail(email)  notify("Password reset email sent successfully!");  setTimeout(() => (router.push('/')), 3500);    *console*.log("Password reset email sent successfully:", res);  } catch (error) {   notify("Failed to send password reset email. Please try again.");  }  } |

Validarea email-ului previne request-uri inutile către Firebase, iar feedback-ul diferențiat pentru succes și eroare ghidează utilizatorul prin proces. Redirectarea automată cu delay oferă timp pentru citirea mesajului de confirmare.

### 5.5.5 Testarea flow-urilor de autentificare

După implementarea completă a logicii, fiecare flow de autentificare a fost testat extensiv pentru validarea funcționalității și identificarea potențialelor probleme.

**Testarea înregistrării de utilizatori**

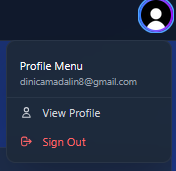


Figura 5.6 – Testare Înregistrare

Testarea manuală a acoperit toate scenariile posibile, validând că aplicația gestionează corect atât succes case-urile cât și erorile. Firebase Authentification oferă validări native pentru format email și complexitatea parolei [4.2].

### 5.5.6 Optimizări de performanță și UX

În urma testării, au fost implementate mai multe optimizări pentru îmbunătățirea experienței utilizatorului și a performanței aplicației.

**Lazy loading pentru componentele Toastify**

|  |
| --- |
| const ToastContainer = dynamic(  () => import("react-toastify").then(mod => mod.ToastContainer),  { ssr: false }  ); |

Loading-ul dinamic al sistemului de notificări reduce dimensiunea bundle-ului principal și îmbunătățește timpul de încărcare inițială [6.5]. SSR: false previne problemele de hidratare în Next.js.

### 5.5.7 Rezultatul fazei și integrarea completă

La finalul acestei faze, sistemul de autentificare este complet funcțional și integrat cu Firebase, oferind:

**Funcționalități complete de autentificare:**

* Înregistrare de utilizatori noi cu validare și feedback
* Autentificare securizată cu gestionarea erorilor
* Resetarea parolei prin email cu confirmare
* Persistența sesiunilor între reîncărcări și sesiuni browser

**Experiență utilizator optimizată:**

* Feedback în timp real prin notificări toast
* Mesaje de eroare user-friendly pentru securitate
* Navigare automată după operațiuni cu succes

**Arhitectură scalabilă și maintainabilă:**

* Hook-uri React pentru gestionarea stărilor Firebase
* Sistem centralizat de notificări reutilizabil
* Gestionarea erorilor consistentă în toate flow-urile
* Optimizări de performanță pentru încărcare rapidă

**Securitate și best practices:**

* Validare client-side și server-side prin Firebase
* Protecția împotriva multiple-submisiilor
* Mesaje de eroare care nu dezvăluie informații sensibile
* Persistența securizată a sesiunilor utilizator

Această fază a transformat interfețele statice create anterior într-un sistem complet funcțional de autentificare care poate susține o platformă educațională reală .

Implementarea robustă a autentificării creează fundația de încredere necesară pentru o platformă educațională, asigurând că datele și progresul fiecărui student sunt protejate și personalizate. Firebase Authentification oferă securitate enterprise-grade fără complexitatea implementării unui sistem propriu, permițând concentrarea asupra aspectelor educaționale specifice aplicației "CodeMaster".