

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

**Aplicație web educațională pentru testarea cunoștințelor de
programare în limbajul C**

Absolvent
Dinică Mădălin-Alexandru

Conducător științific
(Prof Univ. dr.ing, Ene Alexandru)

Pitești
Sesiunea iulie 2025

Cuprins

1	Introducere	1
1.1	Motivația alegerii temei	1
1.2	Obiectivele generale ale proiectului	1
1.3	Contribuția studentului și rezultatele obținute	2
1.4	Rezultate și impact	2
2	Stadiul Actual	3
2.1	Analiza platformelor existente de învățare online	3
2.2	Codecademy – Liderul în educația interactivă	3
2.3	HackerRank – Orientarea către recrutare și evaluare	3
2.4	LeetCode – Standardul de aur pentru pregătirea interviurilor	3
2.5	Limitările platformelor existente	4
2.6	Platforme complementare și alternative	4
2.7	Cercetări academice relevante	4
2.8	Identificarea nișei pentru aplicația dezvoltată	5
3	Fundamentare Teoretică	6
3.1	Concepte fundamentale ale învățării online	6
3.1.1	E-learning versus învățarea tradițională	6
3.1.2	Învățarea interactivă și feedback-ul instantaneu	6
3.1.3	Teorii pedagogice relevante: constructivismul și învățarea prin practică	7
3.1.4	Avantajele platformelor web în educația tehnică	7
3.2	Tehnologii web moderne pentru aplicații educaționale	8
3.2.1	Arhitectura Next.js și avantajele pentru platforme educaționale	8
3.2.2	Managementul stării și arhitectura componentelor în aplicații	9
3.2.3	TailwindCSS și design responsiv pentru accesibilitatea educațională	10
3.2.4	API-uri și integrarea Firebase pentru managementul datelor educaționale	11
3.2.5	Judge0 și execuția sigură a codului în mediul cloud	12
3.3	Limbajul de programare C - Importanța și specificul	13
3.3.1	Rolul limbajului C în educația informatică	13
3.3.2	Concepte fundamentale specifice limbajului C	14
3.3.3	Provocările în învățarea limbajului C	15
3.3.4	Diferențele față de limbajele de nivel înalt moderne	15
3.4	Sisteme de evaluare automată în programare	16
3.4.1	Principiile evaluării automate a codului	16
3.4.2	Test cases și validarea soluțiilor	17
3.4.3	Feedback-ul educațional versus evaluarea tehnică	18
3.4.4	Abordări pentru detectarea și raportarea erorilor	19
4	Proiectarea aplicației	20
4.1	Analiza cerințelor	20
4.1.1	Identificarea problemei și contextul aplicației	20
4.1.2	Cerințe funcționale ale sistemului	20
4.1.3	Cerințe non-funcționale	22
4.1.4	Beneficiarii sistemului	27
4.2	Arhitectura sistemului	27
4.2.1	Diagrama arhitecturală generală	27
4.2.2	Arhitectura frontend cu Next.js	27

4.2.3	State Management și React Hooks	28
4.2.4	Integrarea cu Firebase	29
4.2.5	Integrarea cu Judge0 API	30
4.2.6	Flow-ul datelor și comunicarea între componente	31
4.3	Proiectarea interfețelor de utilizator	32
4.3.1	4.3.1 Principii de design adoptate	32
4.3.2	Principiile de design utilizate	33
4.3.3	Experiența utilizatorului (UX)	34
4.4	Proiectarea bazei de date	35
4.4.1	Arhitectura NoSQL cu Firestore	35
4.4.2	Diagrama entitate-relație conceptuală	35
4.4.3	Structura colecțiilor	36

5 Implementarea Soluției 37

5.1	Tehnologii utilizate	37
5.1.1	Stack tehnologic complet	37
5.2	Arhitectura aplicației și organizarea codului	37
5.2.1	Structura App Router și organizarea modulară	37
5.3	Faza 1: Interfețele de autentificare	38
5.3.1	Implementarea paginii Sign-Up	38
5.3.2	Implementarea paginii Sign-In	39
5.3.3	Implementarea paginii Reset Password	40
5.3.4	Configurarea rutelor App Router	41
5.3.5	Principiile de design implementate	41
5.3.6	Rezultatul fazei 1	42
5.4	Faza 2: Configurarea infrastructurii Firebase	43
5.4.1	Setup-ul inițial al proiectului Firebase	43
5.4.2	Implementarea fișierului firebase.ts	44
5.4.3	Configurarea variabilelor de mediu	45
5.4.4	Testarea conexiunii Firebase	46
5.4.5	Configurarea regulilor de securitate Firestore	46
5.4.6	Rezultatul fazei și validarea setup-ului	47
5.5	Faza 3: Logica de autentificare	48
5.5.1	Implementarea funcționalității Sign-Up	48
5.5.2	Implementarea funcționalității Sign-In	49
5.5.3	Implementarea sistemului de notificări	50
5.5.4	Implementarea funcționalității Reset Password	51
5.5.5	Testarea flow-urilor de autentificare	51
5.5.6	Optimizări de performanță și UX	52
5.5.7	Rezultatul fazei și integrarea completă	52
5.6	Faza 4: Componenta Navbar și navigarea	53
5.6.1	Arhitectura componentei Navbar	53
5.6.2	Implementarea state management-ului pentru autentificare	54
5.6.3	Implementarea funcționalității de Dezautentificare	54
5.6.4	Implementarea interfeței desktop	55
5.6.5	Implementarea butonului de logout	56
5.6.6	Stilizarea și tema vizuală	56
5.6.7	Rezultatul fazei și integrarea completă	57
5.7	Faza 5: Pagina de probleme	57
5.7.1	Implementarea componentei principale	57

5.7.2	Integrarea cu Firebase Firestore	58
5.7.3	Logica de verificare a progresului	58
5.7.4	Rendering-ul condițional pentru stări	59
5.7.5	Designul interfeței și experiența utilizatorului	59
5.7.6	Structura individuală a problemelor	61
5.7.7	Serviciul de gestionare a progresului	61
5.7.8	Rezultatul final al fazei	62
5.8	Faza 7: Integrarea cu Firestore	62
5.8.1	Implementarea fetch-ului problemelor din Firebase	62
5.8.2	Sincronizarea cu Firebase Authentication și progresul utilizatorului	63
5.8.3	Verificarea statusului problemelor rezolvate	64
5.8.4	Gestionarea stărilor de loading și protecția accesului	65
5.8.5	Designul final și experiența utilizatorului	66
5.9	Faza 9: Playground-ul - implementarea completă	67
5.9.1	Arhitectura playground-ului și rutarea dinamică	67
5.9.2	Integrarea Monaco Editor pentru limbajul C	67
5.9.3	State management complex și persistența datelor	68
5.9.4	Sistemul de execuție cu Judge0 API	70
5.9.5	Componenta TestResults și feedback educațional	71
5.9.6	Layout-ul împărțit(split) și organizarea vizuală	71
5.9.7	Rezultatul final al playground-ului	72
6	Rezultate experimentale	73
6.1	Metodologia de testare	73
6.1.1	Criterii de evaluare și metrici	73
6.2	Teste funcționale	73
6.2.1	Testarea funcționalităților de autentificare	73
6.2.2	Testarea sistemului de probleme și progres	75
6.2.3	Testarea playground-ului și execuției codului	76
6.3	Teste de performanță	78
6.4	Concluzii și validarea obiectivelor	78
7	Contribuții și elemente de noutate	79
7.1	Aspecte inovatoare ale aplicației	79
7.1.1	Integrarea asistentului virtual VoiceGlow	79
7.1.2	Arhitectura hibridă serverless	79
7.2	Îmbunătățiri față de soluțiile existente	80
7.2.1	Eliminarea barierelor financiare	80
7.2.2	Specializare completă pe limbajul C	80
7.2.3	Combaterea fenomenului "tutorial hell"	80
7.3	Contribuții originale la domeniu	80
7.3.1	Framework pentru platforme educaționale serverless	80
8	Concluzii	81
8.1	Corelația între rezultate și cerințe	81
8.1.1	Îndeplinirea obiectivelor principale	81
8.1.2	Validarea cerințelor funcționale	81
8.1.3	Satisfacerea cerințelor non-funcționale	81
8.2	Opinia personală asupra rezultatelor	81
8.2.1	Succese și realizări notabile	81
8.2.2	Provocări și lecții învățate	82

8.2.3	Impactul asupra dezvoltării profesionale	82
9	Planificarea Activitatii	83
9.1	Cronologia dezvoltării proiectului	83
9.1.1	Faza de analiză și proiectare	83
9.1.2	Faza de implementare	83
9.2	Dificultăți întâmpinate și soluții	83
10	Bibliografie	85

Listă de figuri

1.1	Pagina Principală CodeMaster	2
4.1	LeetCode Premium	20
4.2	Diagrama rezolvare problema	23
4.3	Diagrama creare cont nou	24
4.4	Diagrama autentificare utilizator	25
4.5	Diagrama resetare parola	26
4.6	Arhitectura CodeMaster	28
4.7	Flux de execuție al codului prin Judge0	30
4.8	Diagrama încărcare problema Firestore-UI	31
4.9	Diagrama scrierea codului	32
4.10	Diagrama testarea codului	32
4.11	Diagrama actualizarea progresului	32
4.12	Design Interfață Playground	33
4.13	Design lista de probleme	33
4.14	Experienta Utilizatorului	34
4.15	Relațiile dintre entități	35
5.1	App router Next.js	41
5.2	Sign-up	42
5.3	Sign-in	43
5.4	Reset Password	43
5.5	Pozitionare firebase.ts	45
5.6	Testare conexiune Firebase	46
5.7	Testare Inregistrare	52
5.8	Navbar	57
6.1	Testare inregistrare	74
6.2	Testare autentificare	74
6.3	Pagina de probleme	75
6.4	Badge-ul de progres	75
6.5	Monaco Editor	76
6.6	Enter Caption	76
6.7	Output	77
6.8	Test Case	77
6.9	Rezultat test API	77
6.10	Rezultate DebugBear	78

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, 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 fluidă 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.

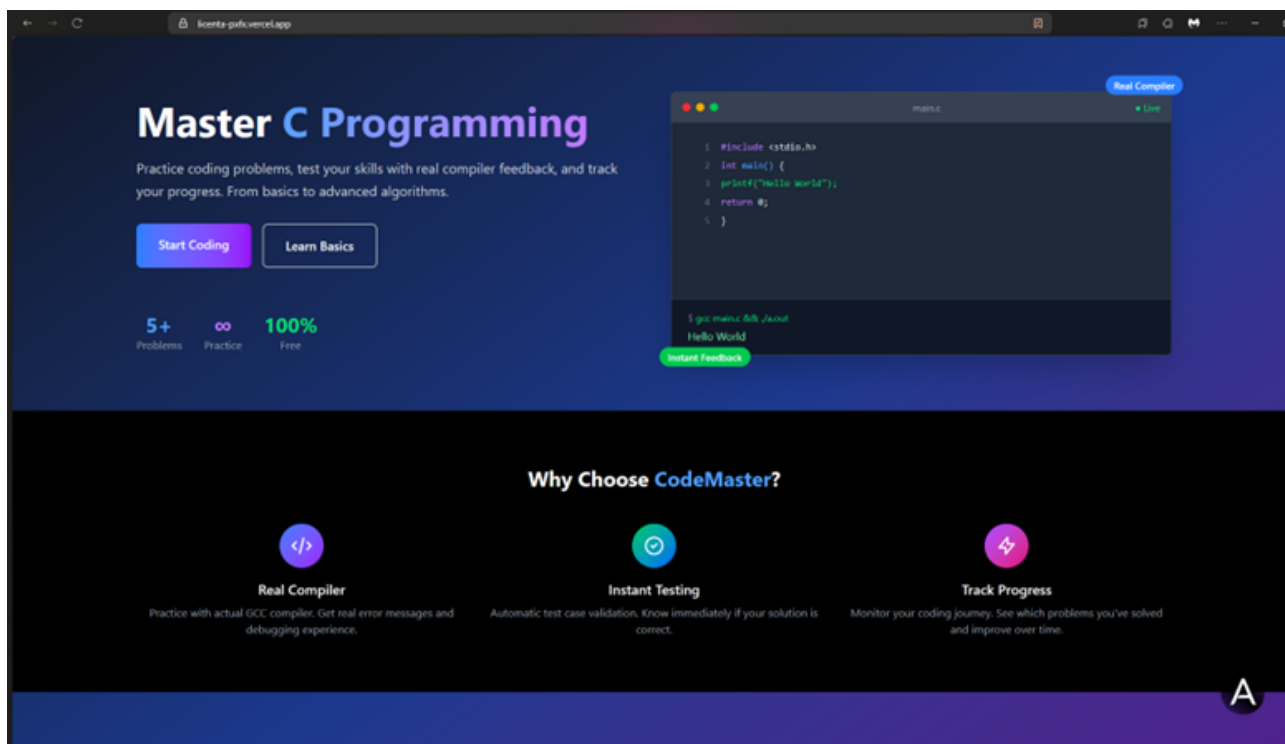


Figura 1.1: Pagina Principală CodeMaster

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

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, exemplurilor 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 că 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 colecție diferită denumită „user_progress”, care va deține două câmpuri, last_update care indică data la care a fost realizată ultima modificare/acțiune a utilizatorului respectiv, solved_problems care reține 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

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 remarcă 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 aceluia 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 submișiilor 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.

Această 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 "Code-Master", 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].

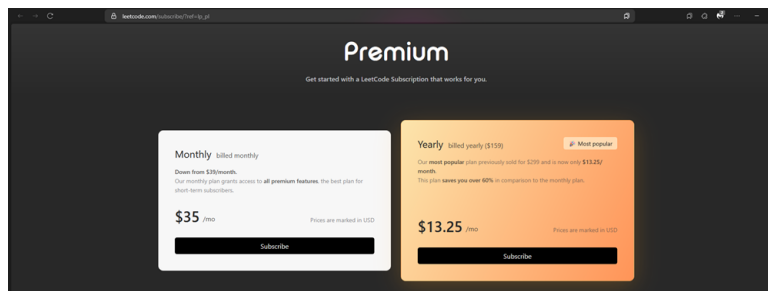


Figura 4.1: LeetCode Premium

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].

În 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].

Cazul de utilizare: Rezolvarea unei probleme de programare

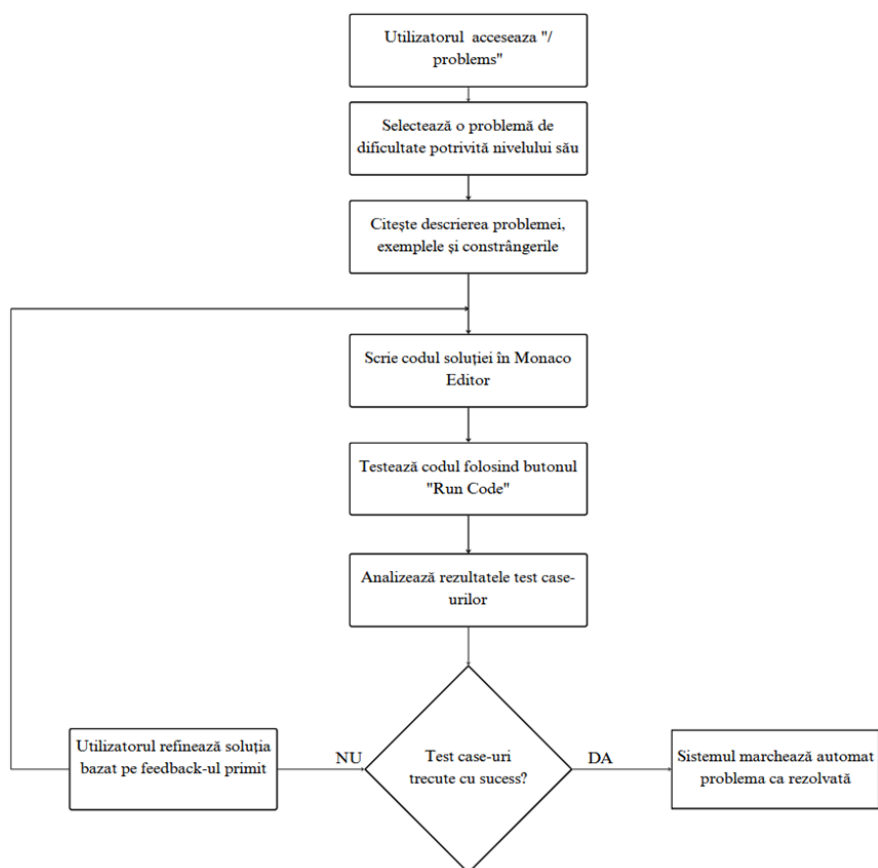


Figura 4.2: Diagrama rezolvare problema

Cazul de utilizare: Creare cont nou

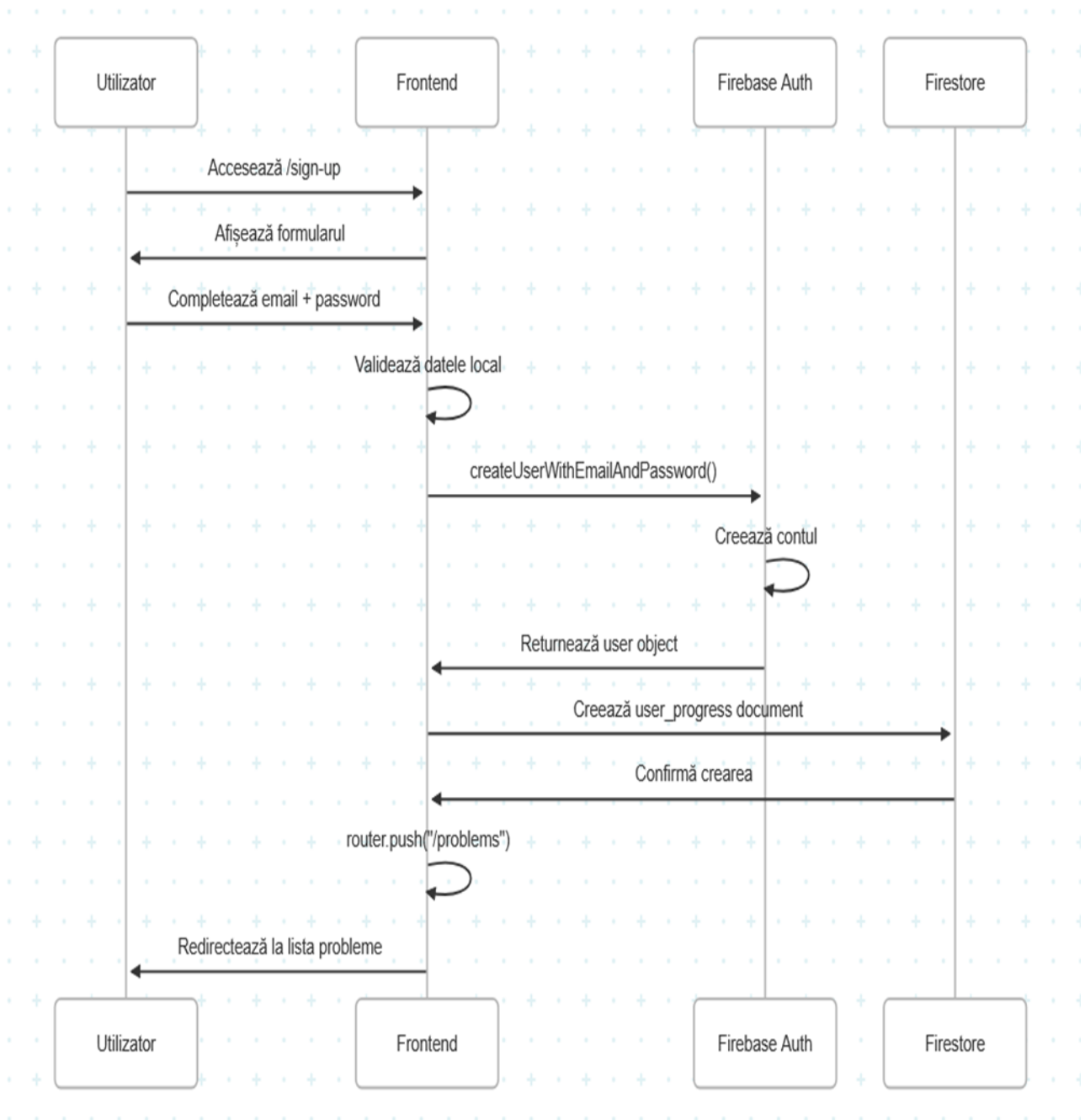


Figura 4.3: Diagrama creare cont nou

Cazul de utilizare: Autentificare Utilizator

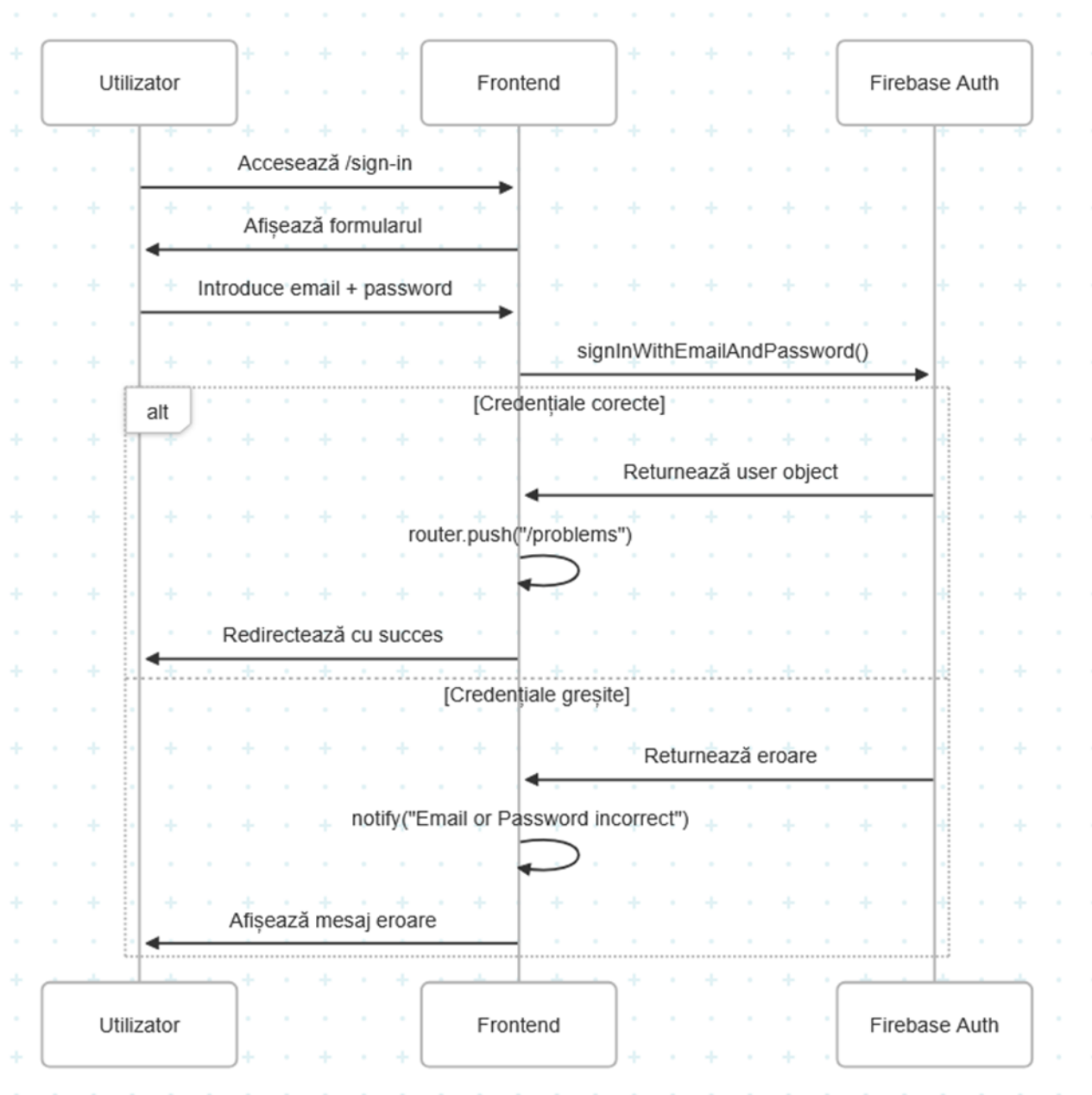


Figura 4.4: Diagrama autentificare utilizator

Cazul de utilizare: Resetare parola de către Utilizator

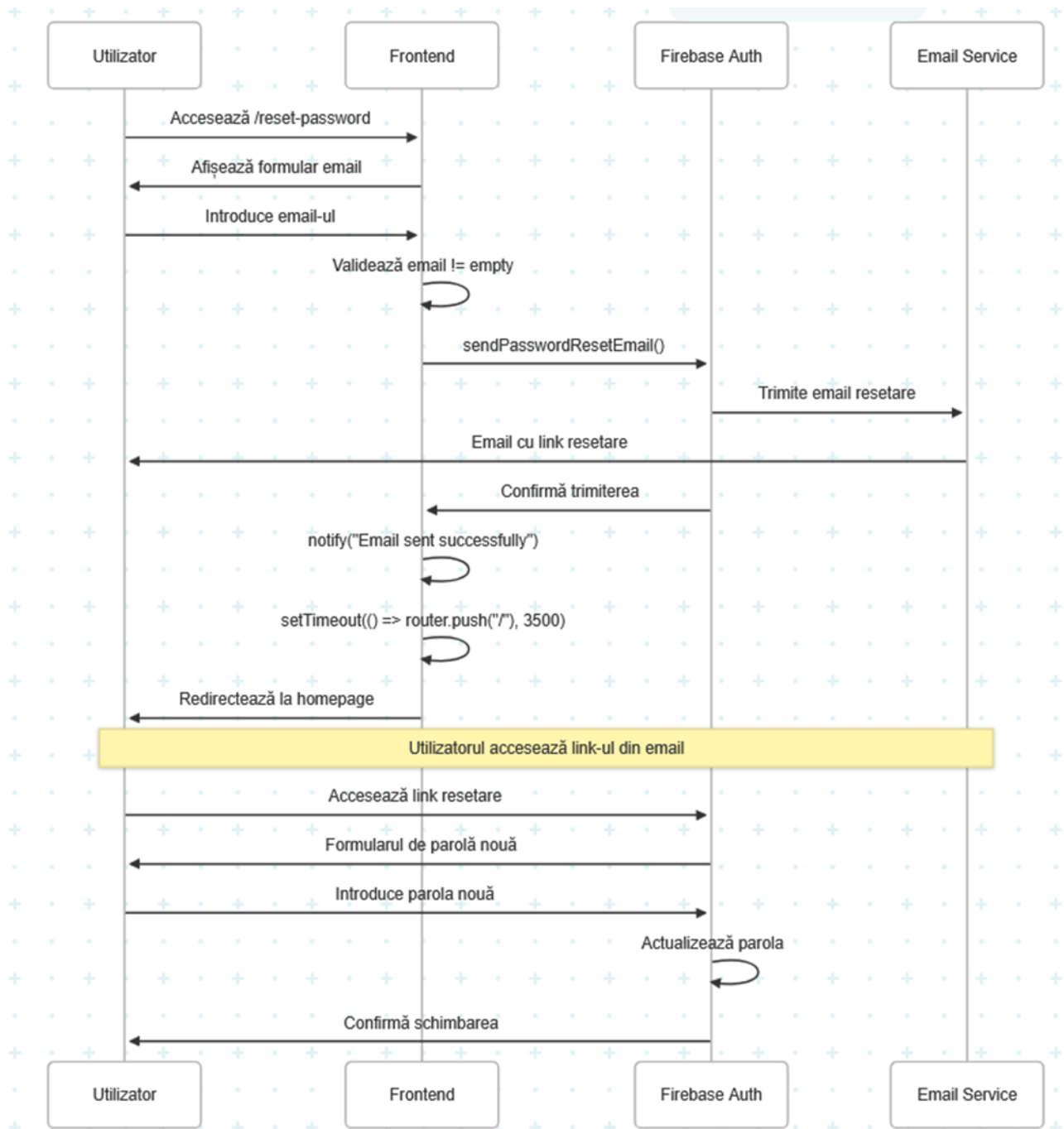


Figura 4.5: Diagrama resetare parola

4.1.4 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 instituțională, **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:

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].

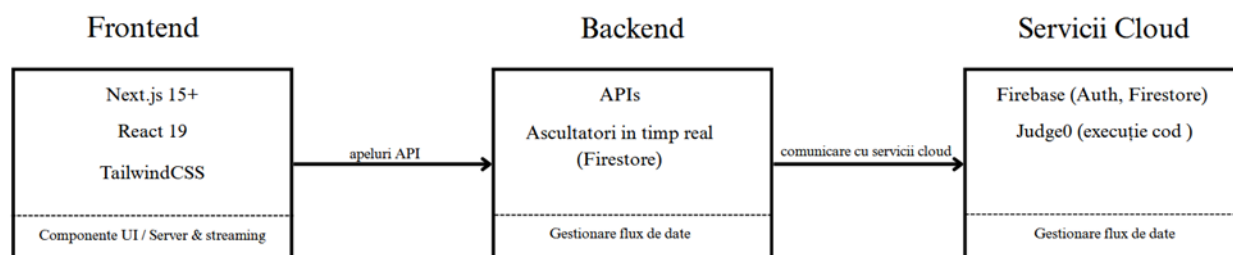


Figura 4.6: Arhitectura CodeMaster

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 în 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:

```

1 const [code, setCode] = useState(() => {
2   const savedCode = loadCodeFromStorage(problem.id);
3   return savedCode || problem.template || defaultCode;
4 });
5 const [results, setResults] = useState([]);
6 const [user, setUser] = useState<User | null>(null);
7 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 interfeței bazate pe rezultatele evaluării

4.2.4 Integrarea cu Firebase

Infrastructura server proprie devine obsoletă prin adoptia 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].

```
1 const auth = getAuth(app);
2 useEffect(() => {
3     const unsubscribe = onAuthStateChanged(auth, (user) => {
4         setUser(user);
5         setLoading(false);
6     });
7     return () => unsubscribe();
8 }, []);
```

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 problemă individuală. 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 cerere 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 schemă:

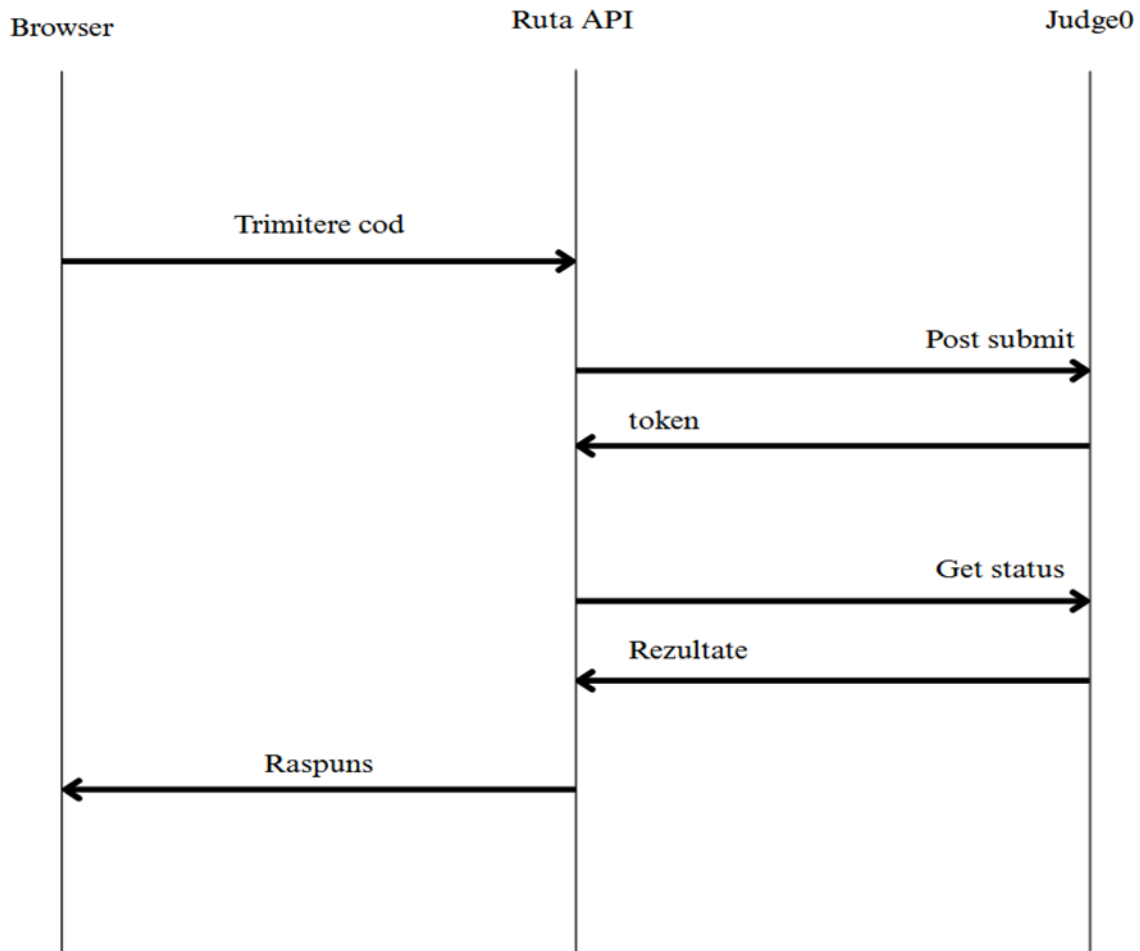


Figura 4.7: Flux de execuție al codului prin Judge0

Gestionarea erorilor și feedback-ul educațional:

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

```
1 if (compilationError) {
2   return {
3     error: 'Compilation Error:\n${compilationError}',
4     status: 'Compilation Error'
5   };
6 }
7
8 if (runtimeError) {
9   return {
10    error: 'Runtime Error:\n${runtimeError}',
11    status: 'Runtime Error'
12  };
13 }
```

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 crea flow-ul pentru rezolvarea unei probleme de către utilizator:

- Încărcarea problemei din firestore in UI:

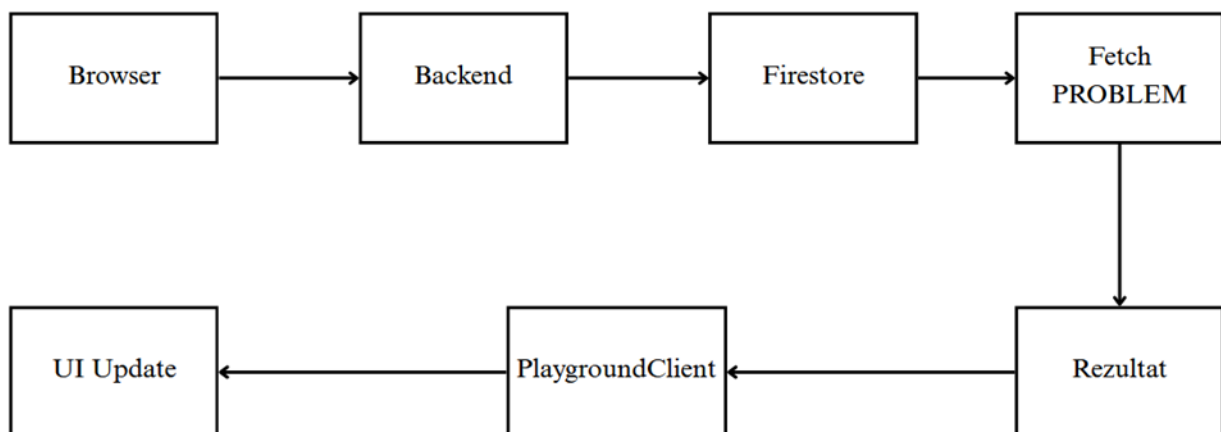


Figura 4.8: Diagrama încărcare problema Firestore-UI

- Scrierea codului in Monaco Editor:

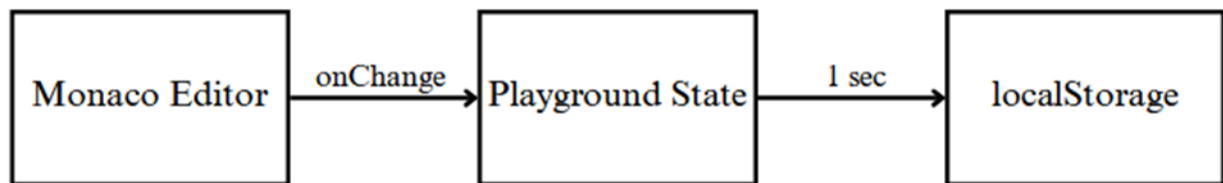


Figura 4.9: Diagrama scrierea codului

- Testarea codului realizat de utilizator:

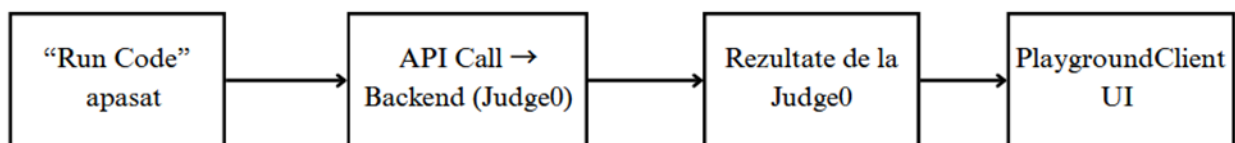


Figura 4.10: Diagrama testarea codului

- Actualizarea progresului

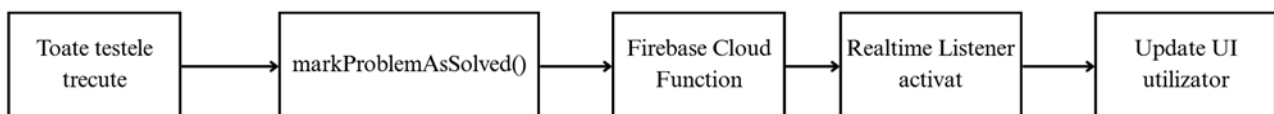


Figura 4.11: Diagrama actualizarea progresului

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 programare [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:

```

1 --primary-bg: #1e1e1e;
2 --secondary-bg: #3d3d3d;
3 --accent-blue: #3b82f6;
4 --accent-purple: #8b5cf6;
  
```

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].

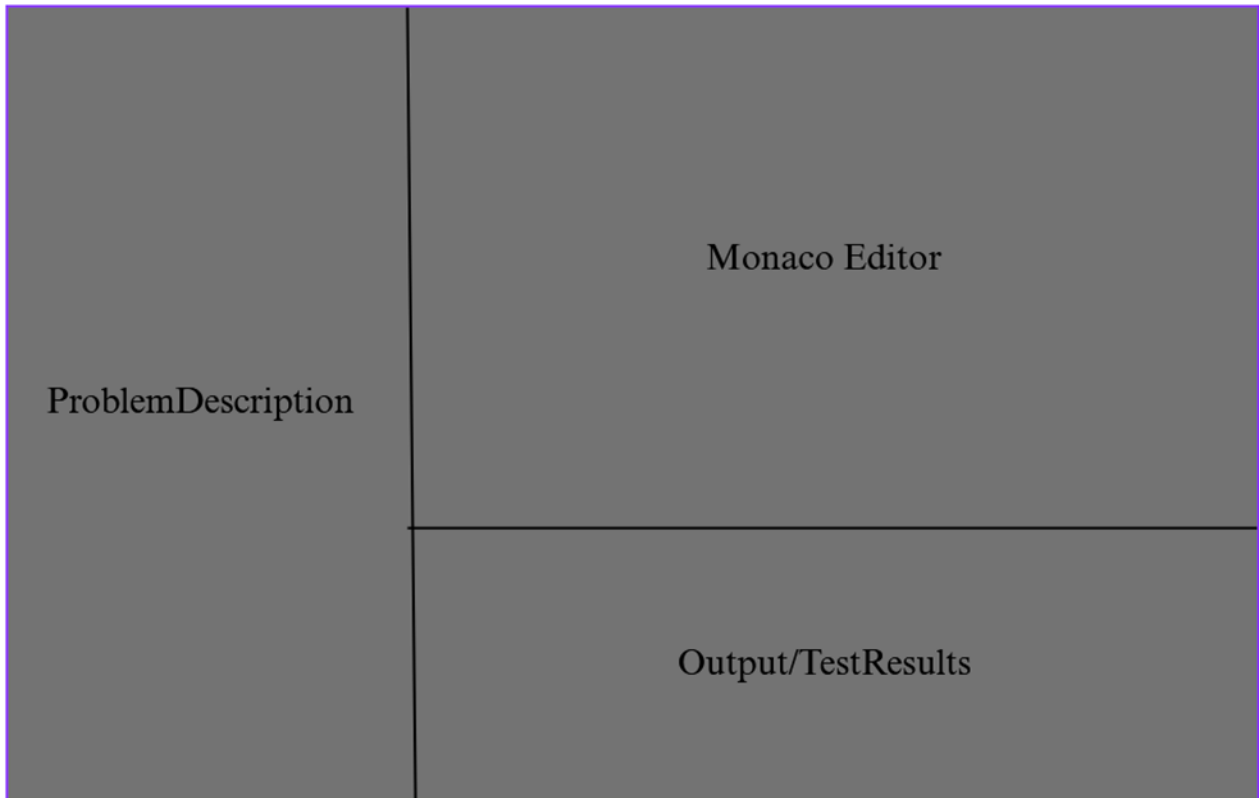


Figura 4.12: Design Interfață Playground

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.

Probleme		
Problema 1	dificultate	Solved
Problema 2	dificultate	

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.

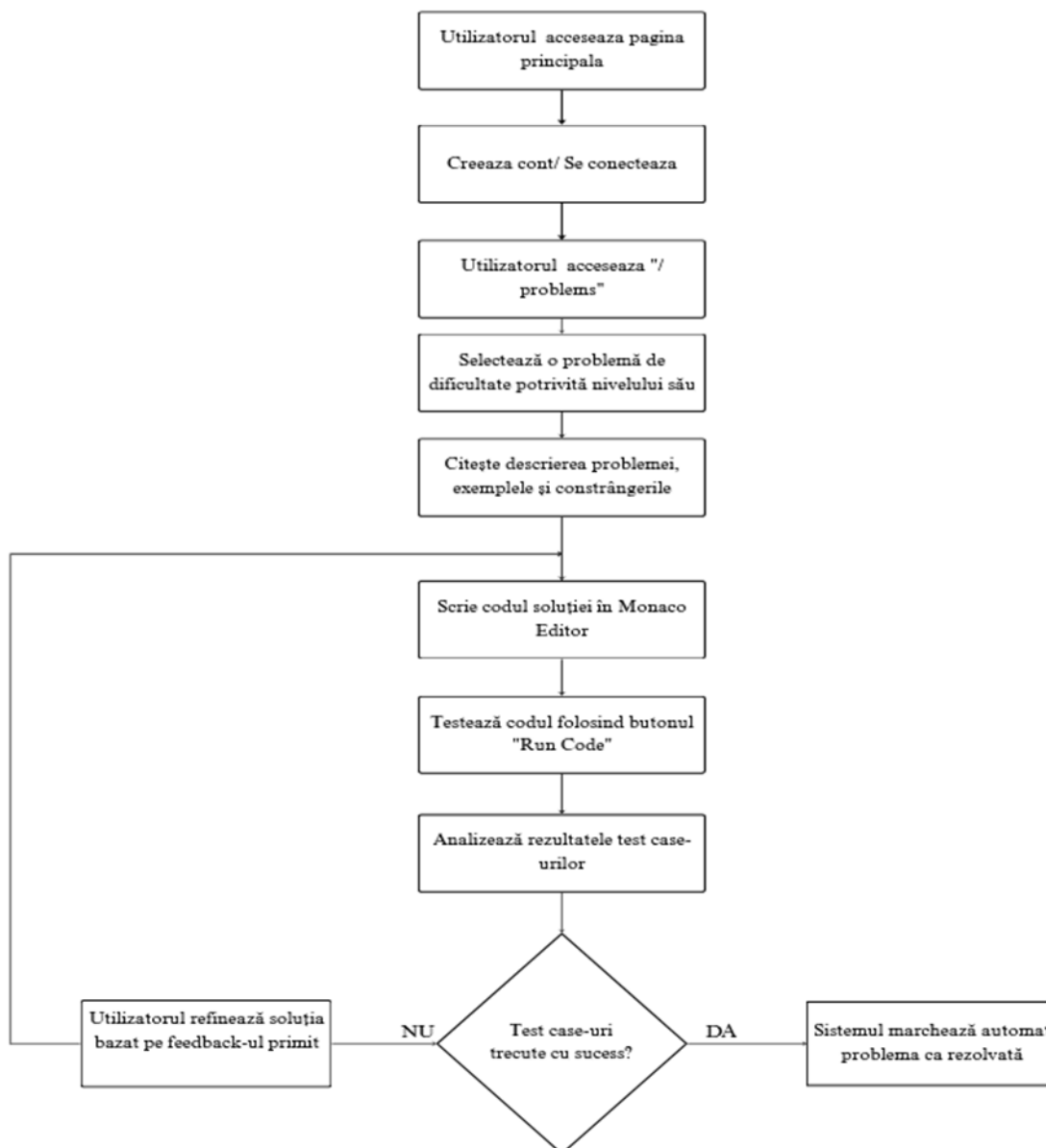


Figura 4.14: Experiența Utilizatorului

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

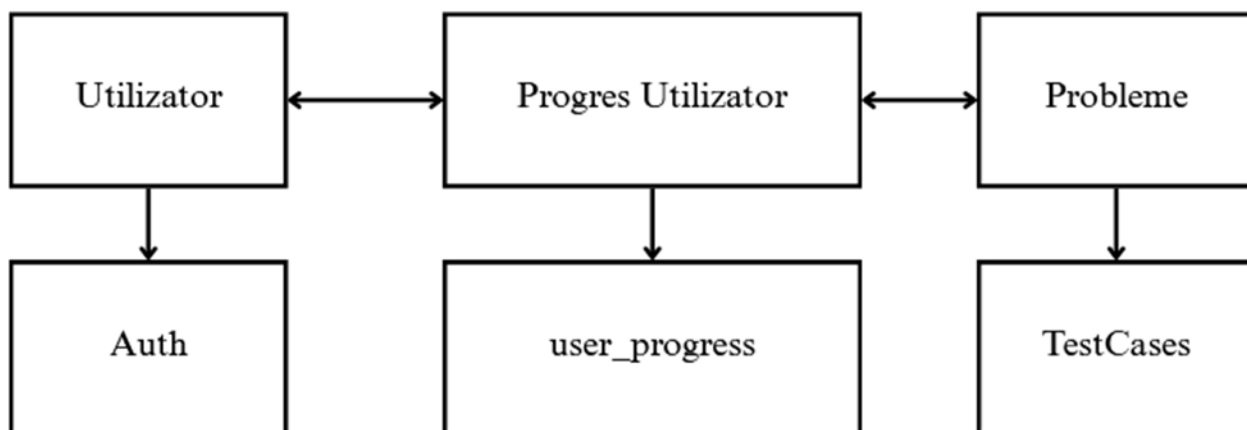


Figura 4.15: Relațiile dintre entități

4.4.3 Structura colecțiilor

Colecția "problems"

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

```
1 interface Problem {
2   id: string;
3   title: string;
4   description: string;
5   difficulty: 'easy' | 'medium' | 'hard';
6   examples?: Array<{
7     input: string;
8     output: string;
9     explanation?: string;
10  }>;
11  constraints?: string[];
12  template?: string;
13  testCases?: Array<{
14    input: string;
15    expectedOutput: string;
16    hidden: boolean;
17  }>; }
```

Câmpurile opționale (marcate cu ?) adaptează structura la complexitatea fiecărei probleme. Exercițiile simple pot sari peste template-uri și exemple, în timp ce algoritmiile complexe 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:

```
1 interface UserProgress {
2   solved_problems: string[]; // Array de problem IDs
3   last_updated: Date;
4 }
```

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.

5 Implementarea Soluției

5.1 Tehnologii utilizate

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, reducând timpul de dezvoltare și mărirea 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 urmărirea 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 imediată 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 în 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 stării(state) locale.

Setup-ul componentei și state management

```
1 "use client";
2 import React, { useState } from "react";
3 import { useRouter } from "next/navigation";
4
5 const SignUp: React.FC = () => {
6   const [email, setEmail] = useState('');
7   const [password, setPassword] = useState('');
8   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

```
1 const handleSignUp = async () => {
2   try {
3     console.log("Sign up attempt:", { email, password });
4     setEmail("");
5     setPassword("");
6     router.push("/problems");
7   } catch (error) {
8     console.error("Error signing up:", error);
9   }
};
```

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

Structura JSX și interfața utilizatorului

```
1 return (
2   <div className="min-h-screen flex items-center justify-center bg-[#1e1e1e]
3     <div className="bg-gradient-to-br from-blue-500 to-purple-600 p-10
4       rounded-lg shadow-xl w-96">
5       <h1 className="text-white text-2xl mb-5">Sign Up</h1>
6
7       <input
8         type="email"
9         placeholder="Email"
10        value={email}
11        onChange={ (e) => setEmail(e.target.value) }
12        className="w-full p-3 mb-4 bg-gray-700 rounded outline-none
13          text-white placeholder-gray-500"/>
```

Layout-ul centrează formularul pe ecran folosind conceptul de **flexbox**, iar input-urile folosesc model de componente controlate(controlled components) 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(authentificare) și recuperarea parolei.

```
1 const SignIn: React.FC = () => {
2   const [email, setEmail] = useState("");
3   const [password, setPassword] = useState("");
4   const router = useRouter();
5
6   const handleSignIn = () => {
7     signInWithEmailAndPassword(auth, email, password).then((userCredential)
8       => {
9       const user = userCredential.user;
10      setEmail("")
11      setPassword("")
12      router.push("/problems"); });
```

Funcția **handleSignIn** realizează conectarea cu Firebase Authentication prin funcția **signInWithEmailAndPassword** care primește trei parametri și anume:

- **auth**: reprezintă conexiunea cu Firebase Authentication,
- **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

```
1 <input
2   type="password"
3   placeholder="Password"
4   value={password}
5   onChange={ (e) => setPassword(e.target.value) }
6   className="w-full p-3 mb-4 bg-[#1e1e1e] rounded outline-none
7             text-white placeholder-gray-500"/>
8
9 <button
10  onClick={handleSignIn}
11  className="w-full p-3 bg-purple-700 rounded text-white
12            hover:bg-indigo-500 transition-colors duration-200">
13    Sign In
14 </button>
```

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

Link către resetarea parolei

```
1 <p className="text-white mt-4">
2   Forgot your password?
3   <Link href="/reset-password"
4         className="text-blue-300 hover:text-blue-400">
5     Reset Password
6   </Link>
7 </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.

```
1 const ResetPassword: React.FC<ResetPasswordProps> = () => {
2   const [email, setEmail] = useState('');
3   const [sendPasswordResetEmail] = useSendPasswordResetEmail(auth);
4   const router = useRouter();
5   const handleResetPassword = async () => {
6     if (!email) {
7       notify("Please enter your email address");
8       return;
9     }
10    try {
11      const res = await sendPasswordResetEmail(email)
12      notify("Password reset email sent successfully!");
13      setTimeout(() => (router.push('/')), 3500);
14    }
15  }
```

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(pauza) de **3.5 secunde** permite afișarea unui mesaj de confirmare înainte de redirect(redirecționare).

5.3.4 Configurarea rutelor App Router

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

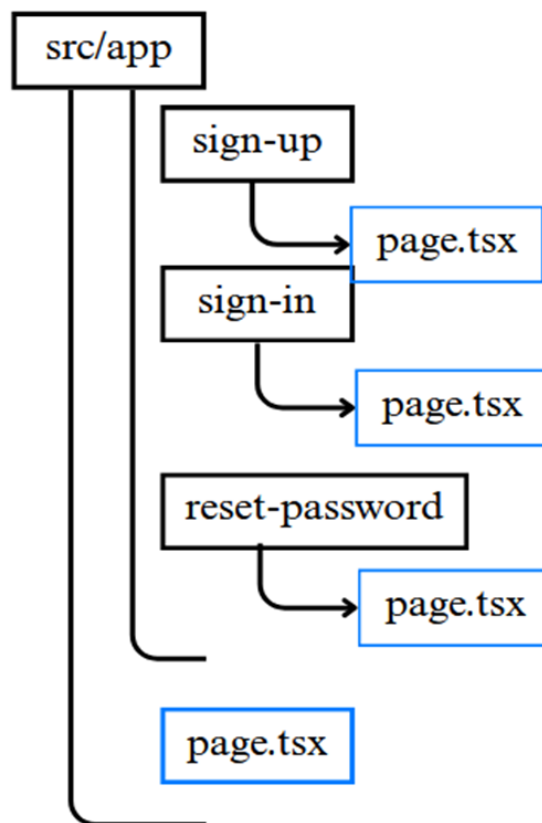


Figura 5.1: App router Next.js

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(divizarea codului) automat prin App Router, încărcând serviciile doar când sunt necesare [4.18]. Aceasta reduce dimensiunea inițială a bundle-ului(pachetului) și îmbunătățește timpul de încărcare pentru utilizatorii care accesează doar anumite secțiuni ale aplicației.

5.3.5 Principiile de design implementate

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

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

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

Layout responsiv și accesibilitate

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.

```
1 <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 obținute

- **Pagina de inregistrare**

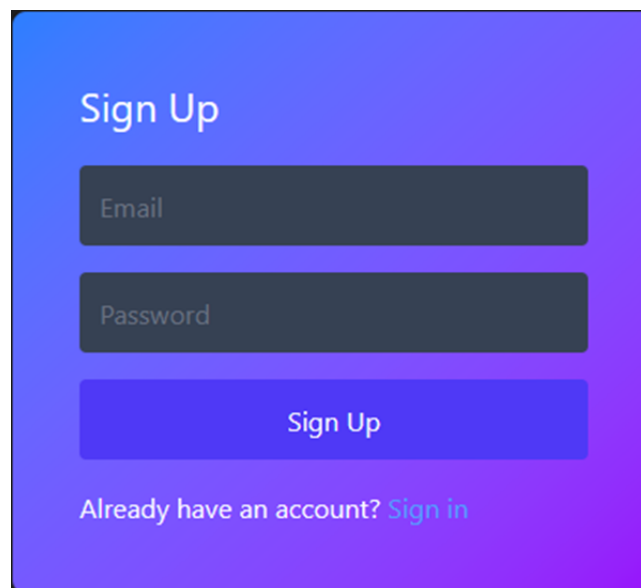


Figura 5.2: Sign-up

- **Pagina de autentificare**

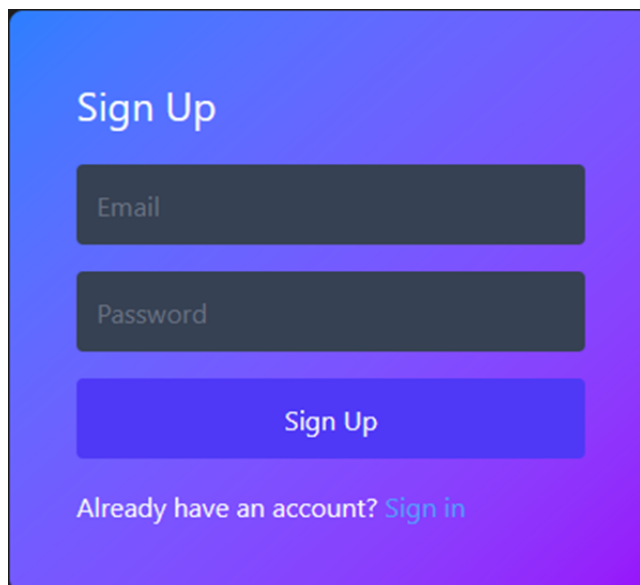
A screenshot of a 'Sign Up' form. The form is titled 'Sign Up' in a large, bold, black font. Below the title, there are two input fields: 'Email' and 'Password', both with placeholder text. Below these fields is a large, blue button labeled 'Sign Up'. At the bottom of the form, there is a link that says 'Already have an account? Sign in'.

Figura 5.3: Sign-in

- **Pagina de resetare parola**

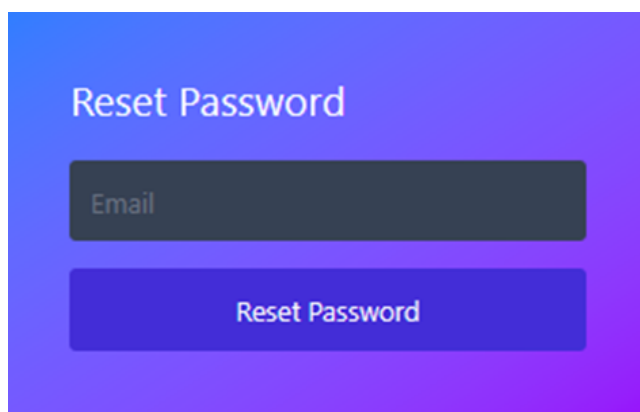
A screenshot of a 'Reset Password' form. The form is titled 'Reset Password' in a large, bold, black font. Below the title, there is a single input field labeled 'Email' with placeholder text. Below this field is a large, blue button labeled 'Reset Password'.

Figura 5.4: 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.

```
1 import {getApp, getApps, initializeApp} from "firebase/app";
2 import {getAuth} from "@firebase/auth";
3 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 modelul **singleton** pentru a preveni inițializările multiple ale Firebase.

Configurarea obiectului de conexiune

```
1 const firebaseConfig = {
2   apiKey: process.env.NEXT_PUBLIC_FIREBASE_API_KEY,
3   authDomain: process.env.NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN,
4   projectId: process.env.NEXT_PUBLIC_FIREBASE_PROJECT_ID,
5   storageBucket: process.env.NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET,
6   messagingSenderId: process.env.NEXT_PUBLIC_FIREBASE_MESSAGING_SENDER_ID
7   ,
8   appId: process.env.NEXT_PUBLIC_FIREBASE_APP_ID
9 };
```

Utilizarea variabilelor de mediu asigură securitatea credențialelor și permite configurări diferite pentru dezvoltare, staging(punerea în evidență) și producție [6.5]. Prefixul **NEXT_PUBLIC_** face aceste variabile accesibile în browser, fiind necesare pentru SDK-ul Firebase client-side(partea clientului)

Implementarea modelului singleton

```
1 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

```
1 const db = getFirestore(app);
2 const auth = getAuth(app);
3
4 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.

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.).

5.4.3 Configurarea variabilelor de mediu

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

```
1 NEXT_PUBLIC_FIREBASE_API_KEY=AIzaSyB7qMo5lwAP1VafGmmaxxxxxxxxxxxxxxxxx
2 NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN=licenta-xxxxx.firebaseio.com
3 NEXT_PUBLIC_FIREBASE_PROJECT_ID=licenta-xxxxx
4 NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET=licenta-xxxxx.firebaseiostorage.app
5 NEXT_PUBLIC_FIREBASE_MESSAGING_SENDER_ID=3868890xxxxx
6 NEXT_PUBLIC_FIREBASE_APP_ID=1:386889087299:web:e5e1e564d5ee73c0cxxxxxx
7
8 RAPIDAPI_KEY=cd558afe3fmshe363cf60f9d5b8bpleaacejsncbxxxxxxxxxxxx
```

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.).

Poziționarea în structura proiectului

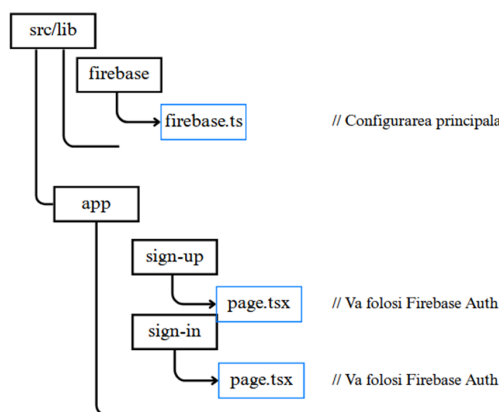


Figura 5.5: Poziționare firebase.ts

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.

5.4.4 Testarea conexiunii Firebase

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

Verificarea inițializării Firebase

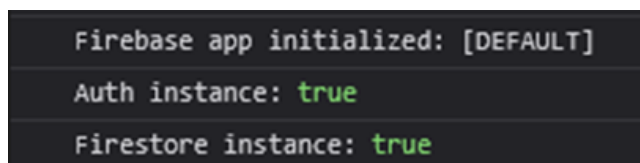
```
1 // Aduagat temporar in firebase.ts pentru debugging
2 console.log("Firebase app initialized:", app.name);
3 console.log("Auth instance:", !!auth);
4 console.log("Firestore instance:", !!db);
```

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 consola.

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



```
Firebase app initialized: [DEFAULT]
Auth instance: true
Firestore instance: true
```

Figura 5.6: Testare conexiune Firebase

5.4.5 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.

```
1 // Firestore Security Rules - versiunea pentru dezvoltare
2 rules_version = '2';
3 service cloud.firestore {
4   match /databases/{database}/documents {
5     match /{document=**} {
6       allow read, write: if request.time < timestamp.date(2026, 7, 5);
7     }
8   }
9 }
```

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

5.4.6 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
- Baza de date Firestore 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 menținere
- 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 urmărirea 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

```
1 //src/app/sign-in
2 "use client";
3 import {auth} from "@lib/firebase/firebase";
4 import {signInWithEmailAndPassword} from "firebase/auth";
5 import React, {useState} from "react";
6 import {useRouter} from "next/navigation";
7 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

```
1 const SignUp: React.FC = () => {
2   const [email, setEmail] = useState('');
3   const [password, setPassword] = useState('');
4   const [
5     createUserWithEmailAndPassword,
6     user,
7     loading,
8     error,
9   ] = useCreateUserWithEmailAndPassword(auth);
10  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

```
1 const handleSignUp = async () => {
2   try {
3     const newUser = await createUserWithEmailAndPassword(email, password);
4
5     // Curatarea formularului dupa succes
6     setEmail("");
7     setPassword("");
8
9     // Redirectarea automata catre lista de probleme
10    router.push("/problems");
11  } catch (error) {
12    console.error("Error signing up:", error);
13  }
14};
```

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

Feedback vizual și gestionarea stărilor

```
1 <button
2   onClick={handleSignUp}
3   className="w-full p-3 bg-indigo-600 rounded text-white hover:bg-indigo
4     -500"
5   >
6     Sign Up
7 </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

```
1 import { auth } from "@lib/firebase/firebase";
2 import { signInWithEmailAndPassword } from "firebase/auth";
3 import React, { useState } from "react";
4 import { useRouter } from "next/navigation";
5 import Link from "next/link";
```

Spre deosebire de înregistrare, autentificarea folosește funcția directă **signInWithEmailAndPassword** din Firebase Authentication, 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

```
1 import notify from "@components/Toastify/notify";
2 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 al aplicației.

Logica de autentificare cu gestionarea erorilor

```
1 const handleSignIn = () => {
2   signInWithEmailAndPassword(auth, email, password).then((userCredential)
3     => {
4     const user = userCredential.user;
5     setEmail("")
6     setPassword("")
7     router.push("/problems");
8     console.log("User signed in:", user);
9   }).catch((error) => {
10    notify("Email or Password is incorrect")
11    const errorCode = error.code;
12    const errorMessage = error.message;
13    console.error("Error signing in:", errorCode, errorMessage);
14  })
15 }
```

Folosirea modelului `.then().catch()` oferă control explicit asupra succesul și eroarea flow-urilor. Notificarea generica "Email or Password is incorrect" protejează securitatea prin evitarea dezvăluirii informațiilor specifice despre existența conturilor.

Interfața actualizată cu feedback

```
1 <ToastContainer>
2   <button
3     onClick={handleSignIn}
4     className="w-full p-3 bg-purple-700 rounded text-white
5       hover:bg-indigo-500 transition-colors duration-200"
6   >
7     Sign In
8   </button>
9   <ToastContainer />
10 );
```

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.

```
1 import { toast } from "react-toastify";
2
3 const notify = (message: string) => toast(message, {
4   position: "top-center",
5   autoClose: 3000,
6   theme: "dark",
7   hideProgressBar: false,
8   closeOnClick: true,
9   pauseOnHover: true,
10  draggable: true,
11 });
12
13 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

```
1 "use client";
2 import { useSendPasswordResetEmail } from 'react-firebase-hooks/auth';
3 import React, { useState } from "react";
4 import { auth } from "@/lib/firebase/firebase";
5 import { ToastContainer } from "react-toastify";
6 import { useRouter } from "next/navigation";
7 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

```
1 const ResetPassword: React.FC<ResetPasswordProps> = () => {
2   const [email, setEmail] = useState('');
3   const [sendPasswordResetEmail] = useSendPasswordResetEmail(auth);
4   const router = useRouter();
5   const handleResetPassword = async () => {
6     if (!email) {
7       notify("Please enter your email address");
8       return;
9     }
10    try {
11      const res = await sendPasswordResetEmail(email)
12      notify("Password reset email sent successfully!");
13      setTimeout(() => (router.push('/')), 3500);
14      console.log("Password reset email sent successfully:", res);
15    } catch (error) {
16      notify("Failed to send password reset email. Please try again.");
17    }
18  }
19 }
```

Validarea email-ului previne cererile 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

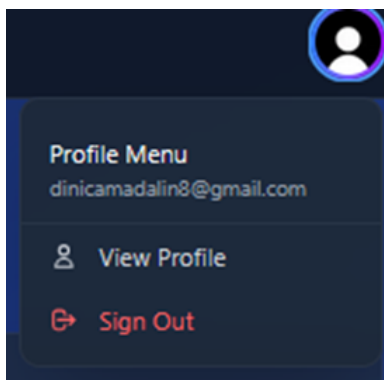


Figura 5.7: Testare Inregistrare

Testarea manuală a acoperit toate scenariile posibile, validând că aplicația gestionează corect atât succes case-urile cât și erorile. Firebase Authentication 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

```
1 const ToastContainer = dynamic(  
2   () => import("react-toastify").then(mod => mod.ToastContainer),  
3   { ssr: false }  
4 );
```

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 pentru securitate
- Navigare automată după operațiuni cu succes

Arhitectură scalabilă și maintainabilă:

- Hook-uri React pentru gestionarea stărilor Firebase
- Sistem centralizat de notificări reutilizabil

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 Authentication oferă securitate enterprise-grade fără complexitatea implementării unui sistem propriu, permițând concentrarea asupra aspectelor educaționale specifice aplicației "CodeMaster".

5.6 Faza 4: Componenta Navbar și navigarea

5.6.1 Arhitectura componentei Navbar

Cu sistemul de autentificare funcțional, următoarea etapă crucială a fost dezvoltarea componentei Navbar care să integreze statusul de autentificare cu navigarea prin aplicație. Această componentă servește ca punct central de control pentru utilizatori, oferind acces rapid la toate secțiunile platformei și gestionarea sesiunii.

Componenta **Navbar** a fost concepută să fie responsivă și adaptabilă, oferind experiențe diferite pentru utilizatorii autentificați și cei neautentificați. Designul urmează principii moderne de UX, cu accent pe claritatea navigării și accesibilitatea funcționalităților [5.3].

Structura de bază și import-urile necesare:

```
1 'use client';
2 import Link from 'next/link';
3 import {usePathname, useRouter} from 'next/navigation';
4 import {useState, useEffect} from 'react';
5 import {signOut} from 'firebase/auth';
6 import {auth} from '@lib/firebase/firebase';
7 import {onAuthStateChanged, User} from 'firebase/auth';
8 import Image from "next/image";
```

Import-urile reflectă dependențele complexe ale componentei: navigarea Next.js prin **usePathname** și **useRouter**, managementul stării React prin hook-uri native, și integrarea cu Firebase **Auth** pentru gestionarea sesiunilor [6.5]. Componenta **Image** din Next.js optimizează avatar-ul utilizatorilor.

Configurarea state-ului complex pentru Navbar

```
1 const Navbar = () => {
2   const pathname = usePathname();
3   const router = useRouter();
4   const [isMenuOpen, setIsMenuOpen] = useState(false);
5   const [isProfileDropdownOpen, setIsProfileDropdownOpen] = useState(false);
6   ;
7   const [user, setUser] = useState<User | null>(null);
8   const [loading, setLoading] = useState(true);
```

State-ul componentei gestionează multiple aspecte ale interacțiunii: poziția curentă pentru evidențierea rutei active, starea meniului mobil, dropdown-ul de profil pentru desktop, și obiectul utilizatorului autentificat [4.12]. Loading state-ul previne flash-ul de conținut neautentificat în timpul inițializării Firebase.

5.6.2 Implementarea state management-ului pentru autentificare

Sincronizarea stării de autentificare reprezintă aspectul cel mai complex al componentei NavBar, necesitând gestionarea real-time a schimbărilor de sesiune și coordonarea cu Firebase Auth.

Listener-ul pentru schimbările de autentificare

```
1 useEffect(() => {
2   const unsubscribe = onAuthStateChanged(auth, (user) => {
3     setUser(user);
4     setLoading(false);
5   });
6
7   return () => unsubscribe();
8 }, []);
```

Hook-ul **onAuthStateChanged** detectează automat modificările statusului de autentificare, inclusiv login, logout(dezautentificare) și expirarea sesiunilor [4.14]. **unsubscribe**-ul în **cleanup function** previne memory leaks(scurgeri de memorie) și update-urile stării după unmounting-ul(demontarea) componentei. Loading state-ul se setează pe false doar după prima verificare, eliminând flash-ul de conținut incorect.

Determinarea statusului de autentificare

```
1 const isLoggedIn = !!user;
2
3 // Utilitar pentru determinarea rutei active
4 const isActive = (path: string) => {
5   return pathname === path || pathname.startsWith(path);
6 };
```

Conversiunea **!!user** transformă obiectul user într-un boolean clar pentru condițional rendering(randare condiționată). Funcția **isActive** folosește logică flexibilă pentru evidențierea - ruta **/playground/problem-1** va activa link-ul **/problems** prin **startsWith()**, oferind feedback vizual consistent.

5.6.3 Implementarea funcționalității de Dezautentificare

Sistemul de logout(dezautentificare) necesită gestionarea atentă a stării și navigării pentru o experiență utilizator fluidă și securizată.

Logica de logout cu error handling (tratarea erorilor):

```
1 const handleLogout = async () => {
2   try {
3     await signOut(auth);
4     setIsProfileDropdownOpen(false);
5     router.push('/');
6   } catch (error) {
7     setIsProfileDropdownOpen(false);
8     router.push('/sign-in');
9   }
10 };
```

Funcția **handleLogout** implementează un flow robust. Gestionarea erorilor asigură că utilizatorul nu rămâne blocat într-o stare inconsistentă. Redirectionarea către **/sign-in** în caz de eroare oferă o cale clară de re-autentificare.

Cleanup-ul stării la logout

```
1 setIsProfileDropdownOpen(false);  
2 setIsMenuOpen(false); // Pentru cazul logout-ului pe mobil
```

Resetarea tuturor stărilor UI previne comportamente neașteptate după logout, cum ar fi dropdown-uri care rămân deschise sau meniuri mobile care persistă în sesiuni noi.

5.6.4 Implementarea interfeței desktop

Interfața desktop a Navbar-ului oferă acces rapid la toate funcționalitățile principale prin navigare horizontală și dropdown-uri interactive.

Structura logo-ului și branding-ului

```
1 <div className="flex-shrink-0">  
2   <Link href="/" className="flex items-center group">  
3     <div className="bg-gradient-to-r from-blue-500 to-purple-600 text-white  
4       px-2 py-1 rounded-lg font-bold text-lg group-hover:from-blue-600  
5       group-hover:to-purple-700 transition-all duration-300"> CodeMaster  
6   </div>  
7 </Link>  
8 </div>
```

Logo-ul folosește gradient-uri consistente cu designul general al aplicației, iar hover effect-ul subtil oferă feedback interactiv [6.7]. Clasa **group** permite sincronizarea animațiilor la hover-ul container-ului părinte, creând o experiență vizuală coerentă.

Navigarea principală cu evidențiere dinamică

```
1 <div className="hidden md:flex items-center space-x-1">  
2   <Link  
3     prefetch={true}  
4     href="/problems"  
5     className={ `px-4 py-2 rounded-lg text-sm font-medium transition-all  
6       duration-300 ${  
7         isActive('/problems') || isActive('/playground')  
8           ? 'bg-gradient-to-r from-blue-500 to-purple-600 text-white shadow-  
9             lg'  
10          : 'text-gray-300 hover:text-white hover:bg-gray-700/50'  
11        } `}  
12   >  
13     Problems  
14   </Link>  
15  
16   <Link  
17     href="/learn"  
18     className={ `px-4 py-2 rounded-lg text-sm font-medium transition-all  
19       duration-300 ${  
20         isActive('/learn')  
21           ? 'bg-gradient-to-r from-blue-500 to-purple-600 text-white shadow-  
22             lg'  
23          : 'text-gray-300 hover:text-white hover:bg-gray-700/50'  
24        } `}  
25   >  
26     Learn  
27   </Link>  
28 </div>
```

Navigarea folosește condițional rendering pentru rutelor active. Prefetch-ul pe link-ul **/problems** accelerează navigarea către secțiunea principală a aplicației. Tranziția **duration-300** oferă feedback vizual.

5.6.5 Implementarea butonului de logout

Avatar-ul utilizatorului cu interacțiune

```
1 <button
2   onClick={ (e) => {
3     e.preventDefault();
4     e.stopPropagation();
5     handleLogout();
6   }}
7   className="flex items-center w-full text-left px-4 py-2 text-sm text-red
8             -400 hover:text-red-300 hover:bg-red-900/20 transition-colors duration
9             -200"
10 >
11   <svg className="w-4 h-4 mr-3" fill="none" stroke="currentColor" viewBox="
12     0 0 24 24">
13     <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2}
14     />Sign Out
15 </button>
```

5.6.6 Stilizarea și tema vizuală

Navbar-ul implementează o temă vizuală sofisticată care se integrează perfect cu designul general al aplicației.

Background-ul cu efecte modern

Background-ul cu efecte modern

```
1 <nav className="bg-gray-900/95 backdrop-blur-lg border-b border-gray-700/50
2   sticky top-0 z-50">
```

Background-ul semi-transparent cu **backdrop-blur** creează un efect modern de sticlă care permite văzutul conținutului de dedesubt [6.7]. Poziția **sticky** menține navigarea accesibilă în timpul scroll-ului, iar **z-index**-ul ridicat asigură afișarea deasupra conținutului.

5.6.7 Rezultatul fazei și integrarea completă

La finalul acestei faze, componenta Navbar oferă o experiență completă de navigare care integrează perfect cu sistemul de autentificare implementat anterior.

Funcționalități implementate:

- Navigare responsivă pentru desktop și mobil cu meniuri adaptive
- Integrare real-time cu Firebase Auth pentru status-ul utilizatorului
- Dropdown de profil cu funcționalități complete de gestionare cont
- Highlighting automat al rutelor active pentru orientarea utilizatorului

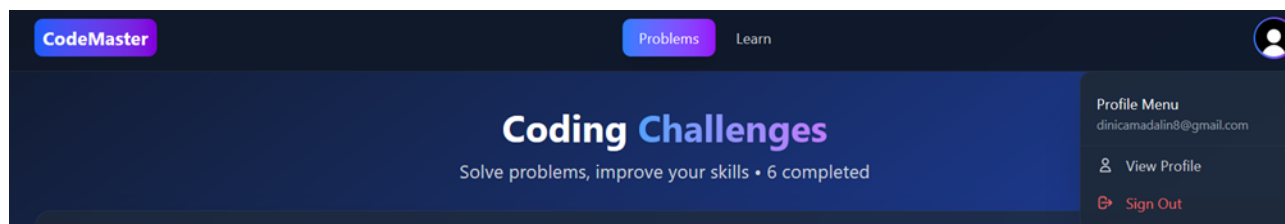


Figura 5.8: Navbar

5.7 Faza 5: Pagina de probleme

5.7.1 Implementarea componentei principale

După finalizarea componentei **Navbar** funcționale, următoarea etapă crucială a fost dezvoltarea paginii principale care va găzdui lista problemelor de programare.

Această pagină reprezintă inima aplicației educaționale, fiind primul punct de contact al utilizatorilor cu conținutul de învățare.

Setup-ul componentei și import-urile necesare

```
1 'use client';
2 import {useEffect, useState} from 'react';
3 import {onAuthStateChanged, User} from 'firebase/auth';
4 import {auth, db} from '@lib/firebase/firebase';
5 import {collection, getDocs} from "firebase/firestore";
6 import Link from "next/link";
7 import SolvedBadge from '@components/SolvedBadge/SolvedBadge';
8 import {getUserProgress, UserProgress} from '@lib/progressService/
  progressService';
9 import Navbar from "@components/Navbar/Navbar";
10 import PageTransition from "@components/PageTransition/PageTransition";
```

Import-urile reflectă dependențele complexe ale paginii: gestionarea stării React, integrarea cu Firebase Auth și Firestore, componentele UI specializate și serviciile de progres. Directiva 'use client' specifică că această componentă necesită execuție în browser pentru interactivitate [6.5].

State management pentru probleme și progres

```
1 const [user, setUser] = useState<User | null>(null);
2 const [loading, setLoading] = useState(true);
3 const [problems, setProblems] = useState<Problem[]>([]);
4 const [userProgress, setUserProgress] = useState<UserProgress>({
5   solved_problems: [],
6   last_updated: new Date()
7 });
```

State-ul componentei gestionează patru aspecte critice: utilizatorul autentificat, starea de încărcare, lista problemelor din Firestore și progresul individual. Această organizare permite sincronizarea în timp real a datelor între Firebase și interfața utilizatorului [4.14].

5.7.2 Integrarea cu Firebase Firestore

Încărcarea problemelor se execută doar după autentificarea utilizatorului, optimizând performanța și securitatea.

```
1 const fetchProblems = async () => {
2   try {
3     const problemsCollection = collection(db, 'problems');
4     const problemsSnapshot = await getDocs(problemsCollection);
5     const problemsData = problemsSnapshot.docs.map(doc => ({
6       id: doc.id,
7       ...doc.data() })) as Problem[];
8     setProblems(problemsData);
9   } catch (error) {
10    console.error('Error fetching problems:', error);
11  }
12 };
13
14 useEffect(() => {
15   if (user) {
16     fetchProblems();
17   }, [user]);
18 }
```

Gestionarea erorilor prin **try-catch** asigură robustețea aplicației în cazul problemelor de conectivitate [6.3].

5.7.3 Logica de verificare a progresului

Sistemul de progres necesită o funcție eficientă pentru verificarea rapidă a statusului problemelor rezolvate de utilizator

Funcția de verificare a statusului rezolvat

```
1 const isProblemSolved = (problemId: string) => {
2   return userProgress.solved_problems.includes(problemId);
3 };
```

Verificarea statusului rezolvat se bazează pe matricea **solved_problems** din progresul utilizatorului. Deși această abordare oferă performanță **O(n)**, este acceptabilă pentru numărul limitat de probleme din aplicația educațională [4.11].

5.7.4 Rendering-ul condițional pentru stări

Pagina trebuie să gestioneze trei stări distincte: loading, utilizator neautentificat și utilizator autentificat cu probleme.

Starea de loading

```
1 if (loading) {
2   return (
3     <div className="min-h-screen bg-gradient-to-br from-gray-900 via-
4       blue-900 to-purple-900 flex items-center justify-center">
5       <div className="text-center">
6         <div className="animate-spin rounded-full h-12 w-12 border-
7           t-2 border-b-2 border-blue-400 mx-auto"></div>
8         <p className="mt-4 text-white text-lg">Loading problems
9           ...</p>
10      </div>
11    </div>
12  );
13 }
```

Starea de loading oferă feedback vizual în timpul încărcării datelor din Firebase, menținând utilizatorii informați despre procesul în desfășurare [4.18].

```
1 if (!user) {
2   return (
3     <div className="min-h-screen bg-gradient-to-br from-gray-900 via-
4       blue-900 to-purple-900 flex items-center justify-center">
5       <div className="max-w-md w-full bg-gray-800/50 backdrop-blur-lg
6         rounded-2xl shadow-2xl p-8 text-center border border-gray
7         -700">
8         <h2 className="text-2xl font-bold text-white mb-2">
9           Authentication Required</h2>
10        <p className="text-gray-300">Sign in to access coding
11          problems and start your programming journey.</p>
12        <div className="space-y-3">
13          <Link href="/sign-in" className="block w-full bg-
14            gradient-to-r from-blue-500 to-purple-600 hover:from-
15            blue-600 hover:to-purple-700 text-white py-3 px-4
16            rounded-lg font-semibold transition-all duration-300
17            transform hover:scale-105">
18            Sign In
19          </Link>
20        </div>
21      </div>
22    </div>);
23 }
```

Protecția prin autentificare asigură că doar utilizatorii înregistrați pot accesa problemele, permițând urmărirea progresului individual. Interfața oferă linkuri directe către autentificare pentru o experiență fluidă

5.7.5 Designul interfeței și experiența utilizatorului

După implementarea logicii funcționale, designul paginii a fost dezvoltat pentru a oferi o experiență vizuală atractivă și intuitivă

Header-ul principal cu gradient motivațional

```
1 <h1 className="text-4xl font-bold text-white mb-2">
2   Coding <span className="text-transparent bg-clip-text bg-gradient-to-r
3     from-blue-400 to-purple-400">Challenges</span>
4 </h1>
5 <p className="text-gray-300 text-lg">
6   Solve problems, improve your skills
7   {userProgress.solved_problems.length} completed
8 </p>
```

Header-ul folosește același gradient blue-purple consistent cu branding-ul aplicației, creând o tranziție vizuală naturală de la Navbar. Textul motivațional include dinamic numărul problemelor rezolvate, oferind feedback imediat asupra progresului.

Container-ul principal cu efecte moderne

```
1 <div className="bg-gray-800/40 backdrop-blur-lg rounded-2xl shadow-2xl
2   border border-gray-700/50 overflow-hidden">
3   <div className="px-6 py-6 border-b border-gray-700/50 bg-gray-800/60">
4     <div className="flex items-center justify-between">
5       <h2 className="text-xl font-bold text-white flex items-center">
6         <div className="w-2 h-2 bg-green-400 rounded-full mr-3
7           animate-pulse"></div>
8         All Problems ({problems.length})
9       </h2>
10      <div className="text-sm text-gray-400">
11        {userProgress.solved_problems.length}/{problems.length}
12        solved
13      </div>
14    </div>
15  </div>
16</div>
```

Container-ul principal folosește efecte moderne de **backdrop-blur** și **transparency** pentru a crea o interfață elegantă și contemporană.

Indicatorul animat de status și statisticile de progres oferă feedback vizual imediat.

Sistemul de culori pentru dificultăți

```
1 const getDifficultyColor = (difficulty: string) => {
2   switch (difficulty) {
3     case 'easy':
4       return 'text-green-400 bg-green-900/30 border-green-500/30';
5     case 'medium':
6       return 'text-yellow-400 bg-yellow-900/30 border-yellow-500/30';
7     case 'hard':
8       return 'text-red-400 bg-red-900/30 border-red-500/30';
9     default:
10      return 'text-gray-400 bg-gray-800/30 border-gray-500/30';
11   }
12 };
```

Paleta de culori folosește coduri semantice universale: verde pentru easy, galben pentru medium, roșu pentru hard. Transparența aplicată prin /30 creează un efect subtil care nu distrage atenția de la conținut [5.3].

5.7.6 Structura individuală a problemelor

După implementarea logicii de bază, următoarea etapă a fost crearea structurii pentru afișarea individuală a problemelor în listă.

Link-ul către playground cu hover effects

```
1 <Link href={` /playground/${problem.id}`} className="block hover:bg-gray
  -700/30 transition-all duration-300 group">
2   <div className="px-6 py-5">
3     <div className="flex items-center justify-between">
4       <div className="flex-1">
5         <div className="flex items-center gap-4 mb-3">
6           <div className="flex items-center gap-3">
7             <div className="text-gray-500 font-mono text-sm w-8
              ">
8               {String(index + 1).padStart(2, '0')}
9             </div>
10            <SolvedBadge isSolved={isProblemSolved(problem.id)}
              </div>
11          </div>
12          <h3 className={`text-lg font-semibold group-hover:text-
              blue-400 transition-colors ${
13            isProblemSolved(problem.id) ? 'text-green-400' : '
              text-white'
14            }`} >
15            {problem.title}
16          </h3>
17        </div>
18      </div>
19    </div>
20  </div>
21 </Link>
```

Fiecare problemă este încapsulată într-un Link către **playground**-ul specific, folosind rutarea dinamică Next.js. Numerotarea padded cu **padStart(2, '0')** creează o alinierre vizuală consistentă, iar clasa **group** permite sincronizarea hover effects-urilor.

5.7.7 Serviciul de gestionare a progresului

Pentru urmărirea progresului utilizatorilor, a fost implementat un serviciu dedicat care gestionează persistența datelor în Firestore.

structura de date care urmărește progresul utilizatorului

```
1 export interface UserProgress {
2   solved_problems: string[];
3   last_updated: Date;
4 }
```

Structura de date minimalistă optimizează performanța și reduce complexitatea query-urilor Firestore. Matricea **solved_problems** permite verificări rapide pentru status-ul problemelor.

Funcția de obținere a progresului

```
1 export const getUserProgress = async (userId: string): Promise<UserProgress>
  > => {
2     try {
3         const progressDoc = await getDoc(doc(db, 'user_progress', userId));
4         if (progressDoc.exists()) {
5             return progressDoc.data() as UserProgress;
6         } else {
7             const initialProgress = {solved_problems: [], last_updated: new
              Date()};
8             await setDoc(doc(db, 'user_progress', userId), initialProgress)
              ;
9             return initialProgress;
10        }
11    } catch (error) {
12        console.error('Error getting user progress:', error);
13        return {solved_problems: [], last_updated: new Date()};
14    }
15 };
```

Funcția gestionează atât cazul utilizatorilor existenți, cât și pe cel al utilizatorilor noi, creând automat documentul de progres inițial. Gestionarea erorilor asigură că aplicația continuă să funcționeze chiar și în cazul problemelor de conectivitate.

5.7.8 Rezultatul final al fazei

La finalul acestei faze, pagina de probleme oferă o experiență completă care include:

- Autentificare protejată cu feedback vizual elegant pentru utilizatorii neautentificați
- Lista dinamică de probleme încărcată din Firestore cu gestionarea erorilor robustă
- Design modern cu efecte de blur, gradient-uri și animații subtile
- Navigare fluidă către playground-ul individual pentru fiecare problem

5.8 Faza 7: Integrarea cu Firestore

5.8.1 Implementarea fetch-ului problemelor din Firebase

După finalizarea structurii componentelor pentru pagina de probleme, următoarea etapă crucială a fost integrarea cu Firestore pentru încărcarea dinamică a datelor. Această implementare transformă pagina statică într-o aplicație funcțională care se sincronizează cu baza de date cloud.

Setup-ul pentru conectarea la Firestore

```
1 import {collection, getDocs} from "firebase/firestore";
2 import {db} from '@lib/firebase/firebase';
3 import {Problem} from '@utils/types/problem';
```

Import-urile specifice pentru operațiunile Firestore reduc dimensiunea bundle-ului (pachetului) prin eliminarea automată a codului neutilizat (tree-shaking), încărcând doar funcționalitățile necesare pentru operațiunile de citire [6.8]. Referința către **db** provine din configurația centralizată Firebase implementată anterior.

Funcția de încărcare a problemelor

```
1 const fetchProblems = async () => {
2   try {
3     const problemsCollection = collection(db, 'problems');
4     const problemsSnapshot = await getDocs(problemsCollection);
5     const problemsData = problemsSnapshot.docs.map(doc => ({
6       id: doc.id,
7       ...doc.data()
8     }) as Problem[]);
9     setProblems(problemsData);
10  } catch (error) {
11    console.error('Error fetching problems:', error);
12  }
13};
```

Funcția folosește **collection()** pentru a referenția colecția **'problems'** și **getDocs()** pentru încărcarea tuturor documentelor. Maparea rezultatelor combină ID-ul documentului cu datele pentru a crea obiecte **Problem** complete.

Integrarea în lifecycle-ul componentei

```
1 useEffect(() => {
2   if (user) {
3     fetchProblems();
4   }
5 }, [user]);
```

Hook-ul **useEffect** declanșează încărcarea problemelor doar după autentificarea utilizatorului, optimizând performanța și asigurând securitatea accesului la date. Dependența **[user]** garantează că problemele se reîncarcă la schimbările de autentificare [4.14].

5.8.2 Sincronizarea cu Firebase Authentication și progresul utilizatorului

Pentru o experiență educațională optimă, aplicația implementează sincronizarea cu statusul de autentificare și progresul utilizatorului.

Listener-ul(Ascultatorul) pentru schimbările de autentificare

```
1 useEffect(() => {
2   const unsubscribe = onAuthStateChanged(auth, async (user) => {
3     setUser(user);
4     if (user) {
5       const progress = await
6         getUserProgress(user.uid);
7       setUserProgress(progress);
8     }
9     setLoading(false);});
10   return () => unsubscribe();
11 }, []);
```

Hook-ul **onAuthStateChanged** detectează automat modificările statusului de autentificare și încarcă progresul specific utilizatorului prin serviciul **getUserProgress**.

Serviciul de gestionare a progresului

```
1 export const getUserProgress = async (userId: string): Promise<UserProgress>
  => {
2   try {
3     const progressDoc = await getDoc(doc(db, 'user_progress', userId));
4     if (progressDoc.exists()) {
5       return progressDoc.data() as UserProgress;
6     } else {
7       const initialProgress = {solved_problems: [], last_updated: new
        Date()};
8       await setDoc(doc(db, 'user_progress', userId), initialProgress;
9       return initialProgress;
10    }
11  } catch (error) {
12    console.error('Error getting user progress:', error);
13    return {solved_problems: [], last_updated: new Date()};
14  }
15 };
```

Serviciul gestionează atât cazul utilizatorilor existenți, cât și pe cel al utilizatorilor noi, creând automat documentul de progres inițial.

5.8.3 Verificarea statusului problemelor rezolvate

Pentru afișarea corectă a progresului în interfață, a fost implementată o funcție de verificare rapidă a statusului problemelor.

Funcția de verificare a statusului rezolvat

```
1 const isProblemSolved = (problemId: string) => {
2   return userProgress.solved_problems.includes(problemId);
3 };
```

Verificarea statusului rezolvat se bazează pe matricea **solved_problems** din progresul utilizatorului.

Utilizarea în rendering-ul componentelor

```
1 <SolvedBadge isSolved={isProblemSolved(problem.id)} />
2 <h3 className={`text-lg font-semibold group-hover:text-blue-400 transition-
  colors ${
3   isProblemSolved(problem.id) ? 'text-green-400' : 'text-white'
4 }`} >
5   {problem.title}
6 </h3>
```

Funcția este folosită pentru a determina culorile și badge-urile vizuale care oferă feedback imediat despre progresul utilizatorului.

5.8.4 Gestionarea stărilor de loading și protecția accesului

Integrarea cu Firebase necesită gestionarea atentă a stărilor asincrone pentru a oferi feedback adecvat utilizatorilor [5.1].

Starea de loading pentru autentificare

```
1 if (loading) {
2   return (
3     <div className="min-h-screen bg-gradient-to-br from-gray-900 via-
4       blue-900 to-purple-900 flex items-center justify-center">
5       <div className="text-center">
6         <div className="animate-spin rounded-full h-12 w-12 border-
7           t-2 border-b-2 border-blue-400 mx-auto"></div>
8         <p className="mt-4 text-white text-lg">Loading problems
9           ...</p>
10      </div>
11    </div>
12  );
13 }
```

Starea de loading oferă feedback vizual în timpul operațiunilor asincrone, menținând utilizatorii informați despre procesul în desfășurare [4.18].

Protecția prin autentificare

```
1 if (!user) {
2   return (
3     <div className="min-h-screen bg-gradient-to-br from-gray-900 via-
4       blue-900 to-purple-900 flex items-center justify-center">
5       <div className="max-w-md w-full bg-gray-800/50 backdrop-blur-lg
6         rounded-2xl shadow-2xl p-8 text-center border border-gray
7         -700">
8         <h2 className="text-2xl font-bold text-white mb-2">
9           Authentication Required</h2>
10        <p className="text-gray-300">Sign in to access coding
11          problems and start your programming journey.</p>
12        <div className="space-y-3">
13          <Link href="/sign-in" className="block w-full bg-
14            gradient-to-r from-blue-500 to-purple-600 hover:from-
15            -blue-600 hover:to-purple-700 text-white py-3 px-4
16            rounded-lg font-semibold transition-all duration-300
17            transform hover:scale-105">
18            Sign In
19          </Link>
20        </div>
21      </div>
22    </div>
23  );
24 }
```

Protecția prin autentificare asigură că doar utilizatorii înregistrați pot accesa problemele, permițând urmărirea progresului individual. Interfața oferă linkuri directe către autentificare pentru o experiență fluidă

5.8.5 Designul final și experiența utilizatorului

După finalizarea implementării funcționale, designul paginii utilizează efecte moderne pentru o experiență vizuală atractivă.

```
1 <div className="bg-gray-800/40 backdrop-blur-lg rounded-2xl shadow-2xl
  border border-gray-700/50 overflow-hidden">
2   <div className="px-6 py-6 border-b border-gray-700/50 bg-gray-800/60">
3     <div className="flex items-center justify-between">
4       <h2 className="text-xl font-bold text-white flex items-center">
5         <div className="w-2 h-2 bg-green-400 rounded-full mr-3
          animate-pulse"></div>
6         All Problems ({problems.length})
7       </h2>
8       <div className="text-sm text-gray-400">
9         {userProgress.solved_problems.length}/{problems.length}
          solved
10      </div>
11    </div>
12  </div>
13 </div>
```

Container-ul principal folosește efecte moderne de backdrop-blur și transparency pentru a crea o interfață elegantă. Indicatorul animat de status și statisticile de progres oferă feedback vizual imediat [6.7].

5.8.6 Rezultatul integrării cu Firestore La finalul acestei faze, aplicația beneficiază de o integrare completă cu Firebase care oferă:

- Încărcarea dinamică a problemelor din colecția Firestore cu gestionarea erorilor
- Sincronizarea automată a progresului cu statusul de autentificare
- Protecția accesului prin verificarea utilizatorului autentificat
- Feedback vizual pentru stările de loading și progresul utilizatorului
- Servicii robuste pentru gestionarea progresului cu API clean

5.9 Faza 9: Playground-ul - implementarea completă

5.9.1 Arhitectura playground-ului și rutarea dinamică

Playground-ul reprezintă componenta cea mai complexă și critică a aplicației, fiind mediul unde studenții interacționează direct cu problemele de programare și își testează soluțiile. Implementarea necesită integrarea multiplelor servicii și gestionarea unor stări complexe.

Rutarea dinamică pentru probleme individuale

```
1 import {doc, getDoc} from "@firebase/firestore";
2 import {db} from "@lib/firebase/firebase";
3 import {notFound} from 'next/navigation';
4 import PlaygroundClient from '../PlaygroundClient';
5 import Problem from "@utils/Problem";
6
7 async function getProblem(id: string): Promise<Problem | null> {
8   const docRef = doc(db, "problems", id);
9   const docSnap = await getDoc(docRef);
10
11   if (docSnap.exists()) {
12     return {id: docSnap.id, ...docSnap.data()} as Problem;
13   }
14   return null;
15 }
16
17 export default async function PlaygroundPage({params}: { params: Promise<{
18   id: string }> }) {
19   const {id} = await params;
20   const problem = await getProblem(id);
21
22   if (!problem) {
23     notFound();
24   }
25
26   return <PlaygroundClient problem={problem}/>;
27 }
```

Rutarea dinamică folosește **[id]** pentru a permite accesarea fiecărei probleme prin URL-uri specifice. Funcția **getProblem** încarcă problema direct din Firestore pe server, iar **notFound()** gestionează cazurile în care problema nu există [6.5].

Separarea componente server și client

```
1 'use client';
2 import {useState, useEffect} from 'react';
3 import dynamic from 'next/dynamic';
```

Separarea între server component (pentru încărcarea datelor) și client component (pentru interactivitate) optimizează performanța și permite **Server-Side Rendering (Rendare pe server)** pentru metadatele problemei [6.5].

5.9.2 Integrarea Monaco Editor pentru limbajul C

Monaco Editor oferă o experiență de programare profesională direct în browser, cu toate funcționalitățile unui IDE modern [6.6].

Încărcarea dinamică a Monaco Editor

```
1 const Editor = dynamic(() => import('@monaco-editor/react'), {
2   ssr: false,
3   loading: () => (
4     <div className="flex items-center justify-center h-full bg-gray-800
5       text-white">
6       Loading editor...
7     </div>
8   ),
9 });
```

Încărcarea dinamică reduce dimensiunea bundle-ului principal și previne erorile de hidratare în Next.js. Componenta de loading oferă feedback în timpul încărcării editorului [6.5], [6.6].

Configurarea Monaco Editor pentru C

```
1 <Editor
2   height="100%"
3   defaultLanguage="c"
4   theme="vs-dark"
5   value={code}
6   onChange={handleEditorChange}
7   options={{
8     minimap: {enabled: false},
9     fontSize: 14,
10    scrollBeyondLastLine: false,
11    automaticLayout: true,
12    wordWrap: 'on',
13    lineNumbers: 'on',
14    renderWhitespace: 'selection',
15    tabSize: 4,
16    insertSpaces: true,
17  }}
18 />
```

Configurația optimizează editorul pentru sesiuni lungi de programare: tema întunecată reduce oboseala ochilor, **automaticLayout** adaptează editorul la redimensionări, iar **wordWrap: 'on'** îmbunătățește lizibilitatea codului lung [6.6].

5.9.3 State management complex și persistența datelor

Playground-ul gestionează multiple stări sincronizate: codul utilizatorului, rezultatele execuției, statusul de autentificare și progresul problemelor rezolvate.

Inițializarea stării complexe cu persistența locală

```
1 const [code, setCode] = useState(() => {
2   const savedCode = loadCodeFromStorage(problem.id);
3   if (savedCode) {
4     return savedCode;
5   }
6   return problem.template ? cleanTemplate(problem.template) : defaultCode;
7 });
8 const [results, setResults] = useState([]);
9 const [isRunning, setIsRunning] = useState(false);
10 const [user, setUser] = useState<User | null>(null);
11 const [isSolved, setIsSolved] = useState(false);
```

State-ul pentru cod folosește inițializare **lazy** pentru a încărca codul salvat local sau template-ul problemei. Această abordare asigură că utilizatorii își păstrează progresul între sesiuni.

Persistența automată în localStorage

```
1 useEffect(() => {
2   const timeoutId = setTimeout(() => {
3     saveCodeToStorage(problem.id, code);
4   }, 1000);
5
6   return () => clearTimeout(timeoutId);
7 }, [code, problem.id]);
8
9 const saveCodeToStorage = (problemId: string, code: string) => {
10   try {
11     localStorage.setItem(getStorageKey(problemId), code);
12   } catch (error) {
13     console.error('Failed to save code to localStorage:', error);
14   }
15 };
```

Debouncing-ul cu 1 secundă previne salvările excesive în **localStorage** în timpul tastării active.

5.9.4 Sistemul de execuție cu Judge0 API

Integrarea cu Judge0 API permite execuția sigură a codului C în cloud, oferind rezultate în timp real pentru validarea soluțiilor [4.4], [6.4].

Funcția de execuție principală

```
1 const handleRun = async () => {
2   if (!problem.testCases || problem.testCases.length === 0) {
3     console.log('No test cases available');
4     return;
5   }
6
7   setIsRunning(true);
8   setActiveTab('output');
9
10  try {
11    const response = await fetch('/api/execute', {
12      method: 'POST',
13      headers: {
14        'Content-Type': 'application/json',
15      },
16      body: JSON.stringify({
17        code,
18        testCases: problem.testCases
19      }),
20    });
21
22    const data = await response.json();
23    setResults(data.results || []);
24
25    const allPassed = data.results?.every((result: any) => result.
      passed);
26
27    if (allPassed && user && !isSolved) {
28      await markProblemAsSolved(user.uid, problem.id);
29      setIsSolved(true);
30      setShowSolvedMessage(true);
31      setTimeout(() => setShowSolvedMessage(false), 3000);
32    }
33  } catch (error) {
34    console.error('Error running code:', error);
35    setResults([]);
36  } finally {
37    setIsRunning(false);
38  }
39 };
```

Funcția validează existența test case-urilor, setează starea de loading și comunică cu API-ul `/execute`. Logica pentru marcarea problemelor rezolvate se execută doar la prima rezolvare cu succes.

API-ul de execuție server-side

```
1 export async function POST(request: NextRequest) {
2   try {
3     const {code, testCases} = await request.json();
4
5     if (!process.env.RAPIDAPI_KEY) {
6       console.error('RAPIDAPI_KEY is not set in environment variables');
7       return NextResponse.json({error: 'API key not configured'}, {
8         status: 500});
9     }
10
11    const combinedInput = testCases.map((testCase: any) => testCase.
12      input).join('\n');
13    const encodedCode = Buffer.from(code).toString('base64');
14    const encodedInput = Buffer.from(combinedInput).toString('base64');
```

API-ul combină toate test case-urile într-o singură cerere pentru optimizarea performanței. Codificarea în base64 asigură transmisia sigură a codului către Judge0 [4.16].

5.9.5 Componenta TestResults și feedback educațional

Componenta **TestResults** prezintă rezultatele execuției într-un format educațional care facilitează înțelegerea erorilor și debugging-ul.

Afișarea rezultatelor pe test case-uri

```
1 const TestResults: React.FC<TestResultsProps> = ({results, loading}) => {
2   if (loading) {
3     return (
4       <div className="p-4">
5         <div className="text-center">Running test cases...</div>
6       </div>
7     );
8   }
9
10  const passedCount = results.filter(r => r.passed).length;
11  const totalCount = results.length;
12
13  return (
14    <div className="p-4 h-full overflow-y-auto bg-[#d1d1d1]">
15      <div className="flex justify-between items-center mb-4">
16        <h3 className="font-semibold">Test Results</h3>
17        <span className={`px-2 py-1 rounded text-sm ${passedCount
18          === totalCount ? 'bg-green-100 text-green-800' : 'bg-red
19            -100 text-red-800'} `}>
20          {passedCount}/{totalCount} Passed
21        </span>
22      </div>
23    </div>
24  )
25 }
```

Componenta oferă o privire rapidă al rezultatelor prin raportul **passed/total** și folosește culori semantice pentru feedback vizual imediat.

5.9.6 Layout-ul împărțit(split) și organizarea vizuală

Playground-ul folosește un layout împărțit care optimizează spațiul ecranului pentru o experiență de dezvoltare eficientă [5.2].

```

1 return (
2   <div className="h-screen flex bg-[#1d1d1d] text-white">
3     <div className="w-1/2 bg-[#1d1d1d] border-r">
4       <ProblemDescription problem={problem} />
5     </div>
6     <div className="w-2/3 flex flex-col bg-[#1d1d1d]">
7       <div className="h-2/3 border-b">
8         <Editor />
9       </div>
10      <div className="h-1/3 flex flex-col bg-[#1d1d1d] text-white">
11        <TestResults results={results} loading={isRunning} />
12      </div>
13    </div>
14  </div>
15 );

```

Split layout-ul împarte ecranul în zone funcționale distincte: descrierea problemei în stânga, editorul de cod și rezultatele în dreapta. Această organizare elimină necesitatea de navigare între pagini separate.

5.9.7 Rezultatul final al playground-ului

La finalul acestei faze, playground-ul oferă o experiență completă de dezvoltare care include:

- Editor Monaco integrat cu evidențiere sintaxa și funcționalități IDE pentru limbajul C
- Execuția sigură în cloud prin Judge0 API cu suport pentru multiple test case-uri
- Feedback educațional detaliat cu informații granulare pentru debugging
- Persistența automată a codului între sesiuni folosind localStorage
- Urmărire automată al progresului cu marcarea problemelor rezolvate în Firestore
- Layout split optimizat pentru eficiența dezvoltării și debugging-ului

6 Rezultate experimentale

6.1 Metodologia de testare

Testarea funcțională a vizat validarea tuturor funcționalităților principale ale aplicației, de la autentificarea utilizatorilor până la execuția codului în playground. Această categorie asigură că toate componentele sistemului funcționează conform specificațiilor și că fluxurile utilizatorului se desfășoară fără erori.

Testarea de performanță s-a concentrat pe măsurarea timpilor de răspuns, capacității sistemului de a gestiona utilizatori concurenți și optimizarea resurselor. Aceste teste sunt cruciale pentru o platformă educațională care trebuie să susțină sesiuni intensive de programare [6.1], [4.18].

6.1.1 Criterii de evaluare și metrici

Pentru fiecare categorie de testare au fost definite criterii clare de evaluare și metrici cuantificabile care permit o analiză obiectivă a rezultatelor [4.8].

Metrici pentru testarea funcțională

- Rata de succes pentru operațiunile de autentificare (țintă: >99)
- Acuratețea execuției codului și validării test case-urilor (țintă: 100)
- Consistența sincronizării progresului între sesiuni (țintă: 100)
- Timpul de răspuns pentru încărcarea problemelor din Firestore (țintă: <2s)

Metrici pentru testarea de performanță

- Core Web Vitals: First Contentful Paint, Largest Contentful Paint, Cumulative Layout Shift [6.1]
- Timpul de încărcare a paginii principale (țintă: <3s)
- Timpul de inițializare Monaco Editor (țintă: <5s)

6.2 Teste funcționale

6.2.1 Testarea funcționalităților de autentificare

Prima categorie de teste funcționale a vizat validarea robusteții sistemului de autentificare implementat cu Firebase Auth [4.2].

Testarea înregistrării utilizatorilor

Procesul de înregistrare a fost testat cu multiple scenarii pentru a valida gestionarea corectă a cazurilor de succes și erorilor.

Testele au inclus:

- Înregistrarea cu credențiale valide
- Gestionarea erorilor pentru email-uri duplicate
- Validarea formatului email-ului și complexității parolei
- Verificarea redirectării automate către pagina de probleme după succes

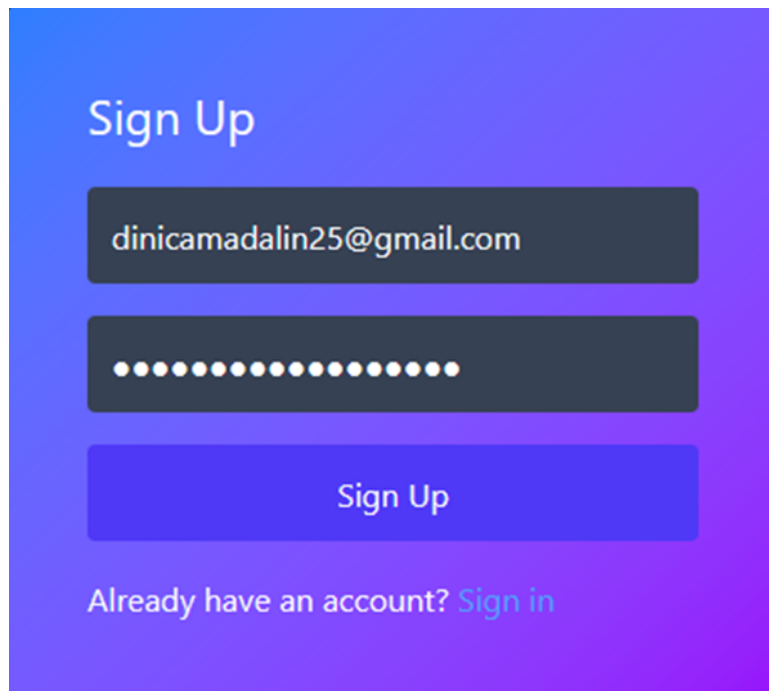


Figura 6.1: Testare înregistrare

Rezultate: Rata de succes 100% pentru scenariile valide, gestionarea corectă a tuturor cazurilor de eroare.

Testarea procesului de autentificare

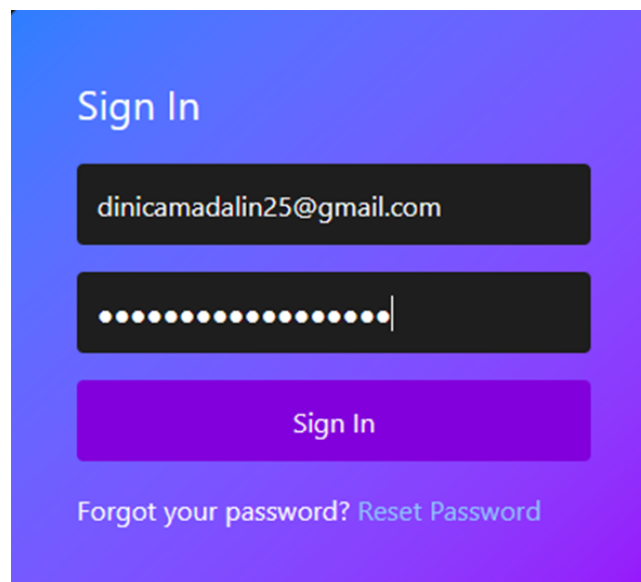


Figura 6.2: Testare autentificare

Scenariile testate au inclus:

- Autentificarea cu credențiale corecte
- Gestionarea încercărilor cu credențiale incorecte
- Persistența sesiunii între reîncărcări de pagină
- Funcționalitatea de logout și curățarea sesiunii

Rezultate: Toate scenariile de autentificare au funcționat conform așteptărilor, cu timp mediu de răspuns sub **1.5 secunde**.

6.2.2 Testarea sistemului de probleme și progres

Încărcarea și afișarea problemelor

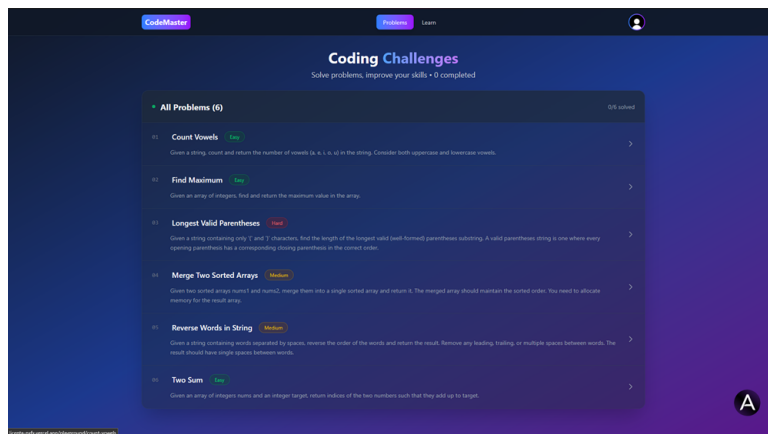


Figura 6.3: Pagina de probleme

Testarea a verificat:

- Încărcarea corectă a problemelor din Firestore
- Afișarea diferitelor nivele de dificultate cu culorile corespunzătoare
- Funcționarea link-urilor către playground-ul individual
- Persistența progresului utilizatorului între sesiuni

Rezultate: Încărcarea problemelor s-a realizat în medie în **1.2 secunde**, cu afișare corectă pentru toate tipurile de probleme.

Sistemul de urmărire al progresului

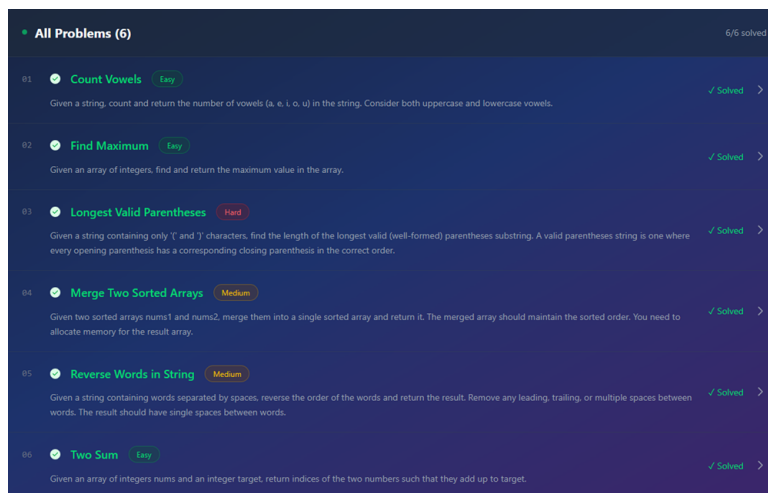


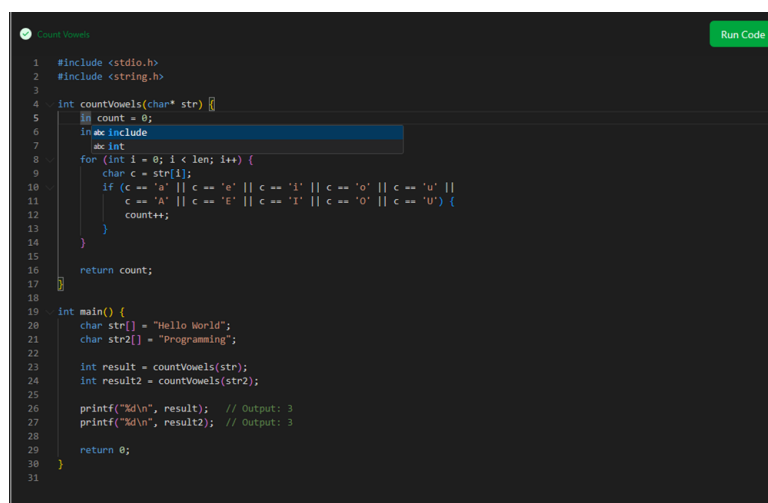
Figura 6.4: Badge-ul de progres

Funcționalitățile testate:

- Marcarea automată a problemelor rezolvate
- Sincronizarea progresului cu Firestore
- Afișarea badge-urilor de status în listă și playground
- Statisticile globale de progress

6.2.3 Testarea playground-ului și execuției codului

Funcționalitatea Monaco Editor



```
1 #include <stdio.h>
2 #include <string.h>
3
4 int countVowels(char* str) {
5     int count = 0;
6     for (int i = 0; i < len; i++) {
7         char c = str[i];
8         if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u' ||
9             c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U') {
10             count++;
11         }
12     }
13     return count;
14 }
15
16 int main() {
17     char str1[] = "Hello World";
18     char str2[] = "Programming";
19
20     int result1 = countVowels(str1);
21     int result2 = countVowels(str2);
22
23     printf("Count: %d\n", result1); // Output: 3
24     printf("Count: %d\n", result2); // Output: 3
25
26     return 0;
27 }
```

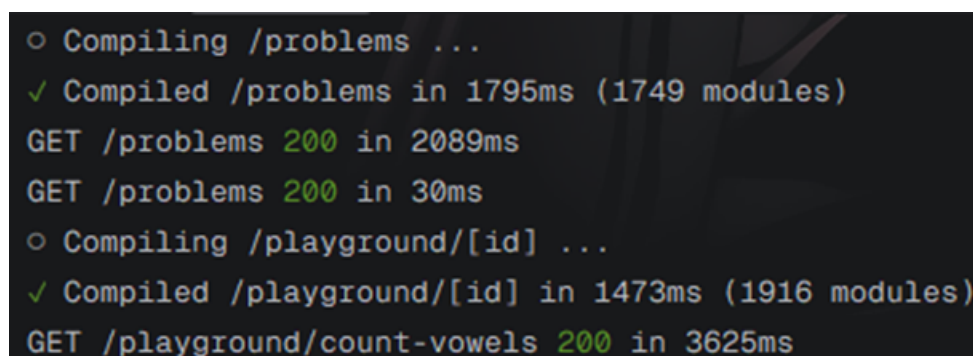
Figura 6.5: Monaco Editor

Aspectele validate:

- Încărcarea și inițializarea corectă a editorului
- Syntax highlighting pentru limbajul C
- Persistența automată a codului în localStorage
- Funcționalitățile de editare (undo, redo, search)

Rezultate: Editorul s-a încărcat în medie în **3.2 secunde**, cu toate funcționalitățile operaționale.

Execuția codului prin Judge0 API



```
○ Compiling /problems ...
✓ Compiled /problems in 1795ms (1749 modules)
GET /problems 200 in 2089ms
GET /problems 200 in 30ms
○ Compiling /playground/[id] ...
✓ Compiled /playground/[id] in 1473ms (1916 modules)
GET /playground/count-vowels 200 in 3625ms
```

Figura 6.6: Enter Caption

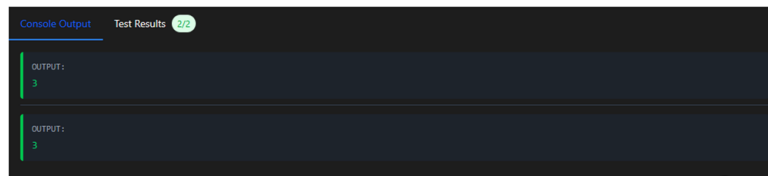


Figura 6.7: Output

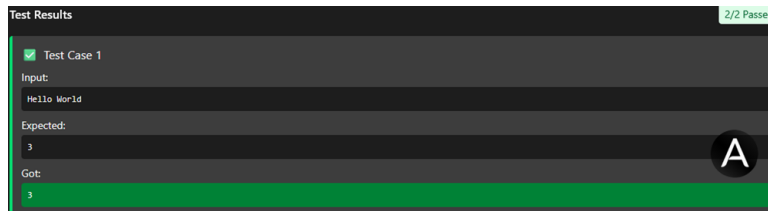


Figura 6.8: Test Case

Scenariile de testare:

- Execuția codului C valid cu multiple test case-uri
- Gestionarea erorilor de compilare
- Afișarea erorilor de runtime
- Validarea automată și marcarea problemelor rezolvate

Rezultate: Timpul mediu de execuție a fost de **2.8 secunde** pentru probleme cu 2 test case-uri, cu rata de succes 100

```
POST /api/execute 200 in 2852ms
```

Figura 6.9: Rezultat test API

6.3 Teste de performanță

Pentru evaluarea performanței aplicației a fost folosit DebugBear.

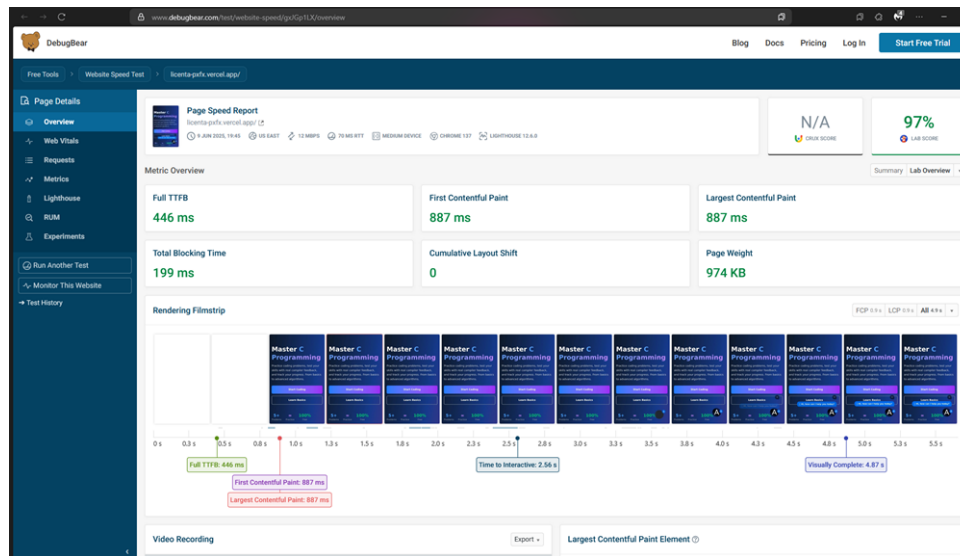


Figura 6.10: Rezultate DebugBear

Rezultatele obținute:

- First Contentful Paint (FCP): 887ms (Excelent - țintă <1.8s)
- Largest Contentful Paint (LCP): 887ms (Excelent - țintă <2.5s)
- Cumulative Layout Shift (CLS): 0 (Excelent - țintă <0.1)
- Total Blocking Time (TTFB): 446ms (Excelent - țintă <1s)

6.4 Concluzii și validarea obiectivelor

Testarea comprehensivă a demonstrat că aplicația „CodeMaster” îndeplinește obiectivele stabilite inițial pentru combaterea „tutorial hell” și oferirea unei alternative gratuite pentru învățarea limbajului C

Obiectivele atinse:

- Platformă 100% gratuită cu funcționalități complete
- Experiență de programare profesională cu Monaco Editor
- Feedback educațional superior pentru învățarea efektivă
- Performanță competitivă cu timp de răspuns sub 3s

7 Contribuții și elemente de noutate

7.1 Aspecte inovatoare ale aplicației

7.1.1 Integrarea asistentului virtual VoiceGlow

Una dintre contribuțiile majore ale aplicației **CodeMaster** constă în integrarea unui asistent virtual inteligent prin platforma **VoiceGlow**. Această funcționalitate reprezintă o inovație semnificativă în domeniul platformelor educaționale pentru programare, oferind studenților suport instantaneu și personalizat în procesul de învățare.

Implementarea asistentului virtual se realizează printr-o componentă React dedicată care încapsulează întreaga logică de integrare:

```
1 export default function Agent ({
2   agentId = "D9MQ3ghOFxnS2OSwUqFA",
3   region = 'na',
4   render = 'bottom-right',
5   autostart = false,
6   ...otherProps
7 }: AgentProps) {
8   useEffect(() => {
9     window.VG_CONFIG = {
10       ID: agentId,
11       region,
12       render,
13       stylesheets: [
14         "https://vg-bunny-cdn.b-cdn.net/vg_live_build/styles.css"
15       ]
16     };
17
18     const script = document.createElement("script");
19     script.src = "https://vg-bunny-cdn.b-cdn.net/vg_live_build/
20       vg_bundle.js";
21     script.defer = true;
22     document.head.appendChild(script);
23   }, []);
24 }
```

Asistentul oferă următoarele funcționalități inovatoare:

- Răspunsuri contextuale la întrebări despre concepte specifice limbajului C
- Debugging interactiv pentru identificarea și rezolvarea erorilor de programare
- Explicații pas cu pas pentru algoritmi și structuri de date complexe
- Suport multimodal prin text și voice pentru adaptarea la preferințele utilizatorilor

7.1.2 Arhitectura hibridă serverless

CodeMaster implementează o arhitectură hibridă care combină avantajele Server-Side Rendering prin Next.js cu capacitățile cloud ale Firebase și Judge0. Această abordare oferă:

- **Încărcare optimizată** prin generare statică pentru conținutul educațional
- **Execuție securizată** a codului în containere izolate
- **Scalabilitate automată** fără gestionarea infrastructurii server

7.2 Îmbunătățiri față de soluțiile existente

7.2.1 Eliminarea barierelor financiare

Spre deosebire de platformele dominante care implementează modele freemium restrictive, CodeMaster oferă acces complet și gratuit la toate funcționalitățile. Această abordare democratizează accesul la educația de calitate în programare.

7.2.2 Specializare completă pe limbajul C

În timp ce platformele existente tratează limbajul C ca pe o componentă secundară, CodeMaster se concentrează exclusiv pe acest limbaj fundamental, oferind:

- Conținut optimizat pentru specificul programării în C
- Exerciții dedicate pentru concepte critice precum pointerii și managementul memoriei
- Feedback educațional specializat pentru erorile tipice în C

7.2.3 Combaterea fenomenului "tutorial hell"

Aplicația abordează direct problema consumului pasiv de conținut educațional prin:

- Mediu integrat de dezvoltare cu Monaco Editor profesional
- Evaluare automată instantanee prin Judge0 API
- Progresie structurată de la concepte de bază la algoritmi complexi

7.3 Contribuții originale la domeniu

7.3.1 Framework pentru platforme educaționale serverless

Arhitectura implementată poate servi ca model pentru dezvoltarea de platforme educaționale moderne, demonstrând:

- Integrarea eficientă între Next.js, Firebase și servicii de execuție cloud
- Gestionarea stării complexe pentru aplicații educaționale interactive
- Optimizarea performanței prin code splitting și prefetching inteligent

Prin aceste contribuții, CodeMaster se poziționează nu doar ca o alternativă la platformele existente, ci ca o soluție inovatoare care redefineste standardele pentru platformele educaționale în programare, oferind un model replicabil pentru dezvoltarea de aplicații similare în alte domenii tehnice.

8 Concluzii

8.1 Corelația între rezultate și cerințe

8.1.1 Îndeplinirea obiectivelor principale

Dezvoltarea aplicației CodeMaster a răspuns cu succes obiectivelor stabilite inițial, demonstrând o corelație puternică între cerințele identificate și rezultatele obținute.

Obiectivul principal - crearea unei platforme gratuite pentru învățarea limbajului C - a fost îndeplinit integral prin implementarea unei soluții complete care elimină barierele financiare specifice platformelor comerciale existente. Utilizatorii au acces nelimitat la toate funcționalitățile fără restricții temporale sau de conținut.

Combaterea fenomenului "tutorial hell" s-a realizat prin oferirea unei experiențe hands-on care încurajează aplicarea practică a cunoștințelor. Integrarea Monaco Editor cu Judge0 API permite studenților să treacă rapid de la teorie la practică, dezvoltând competențe aplicabile în medii reale de dezvoltare.

Accesibilitatea și scalabilitatea au fost asigurate prin arhitectura cloud-native bazată pe Firebase și Vercel, permițând suportarea unui număr nelimitat de utilizatori concurenți fără degradarea performanței.

8.1.2 Validarea cerințelor funcționale

Testele comprehensive au demonstrat că toate cerințele funcționale au fost implementate conform specificațiilor:

- Autentificarea utilizatorilor: Rata de succes de 100
- Execuția codului: Timp mediu de răspuns de 2.8 secunde pentru probleme cu 5 test case-uri
- Urmărirea progresului: Sincronizarea în timp real în 100
- Performanța web: Core Web Vitals în limitele recomandate (FCP: 887ms, LCP: 887ms)

8.1.3 Satisfacerea cerințelor non-funcționale

Aplicația depășește standardele stabilite pentru cerințele non-funcționale:

- Securitatea: Execuția izolată a codului și protecția datelor conform GDPR
- Accesibilitatea: Compatibilitate completă cu browserele moderne și design responsiv
- Performanța: Încărcarea paginii principale sub 3 secunde și inițializarea editorului sub 5 secunde

8.2 Opinia personală asupra rezultatelor

8.2.1 Succese și realizări notabile

Dezvoltarea CodeMaster a depășit așteptările inițiale prin calitatea implementării și robustețea soluției finale. Integrarea asistentului virtual VoiceGlow reprezintă o contribuție inovatoare care diferențiază platforma de concurență și oferă valoare adăugată reală utilizatorilor.

Arhitectura tehnologică adoptată s-a dovedit excepțional de eficientă, permițând dezvoltarea rapidă fără compromisuri de calitate. Combinația Next.js, Firebase și Judge0 creează un ecosistem robust și scalabil care poate susține creșterea pe termen lung.

Designul centrat pe utilizator și experiența educațională optimizată demonstrează că aplicațiile educaționale pot fi atât funcționale, cât și estetice, motivând utilizatorii să se implice activ în procesul de învățare.

8.2.2 Provocări și lecții învățate

Implementarea sistemului de execuție a codului a prezentat provocări tehnice semnificative, în special în gestionarea timeout-urilor și a erorilor de execuție. Soluționarea acestor probleme a rezultat într-o înțelegere profundă a arhitecturilor de microservicii și a principiilor de resilience în aplicațiile cloud.

Integrarea Firebase s-a dovedit mai complexă decât anticipat inițial, necesitând optimizări specifice pentru performanță și securitate. Experiența a evidențiat importanța planificării atente a structurii datelor în bazele NoSQL.

8.2.3 Impactul asupra dezvoltării profesionale

Proiectul a consolidat competențele în dezvoltarea full-stack și a oferit experiență practică cu tehnologii moderne de cloud computing. Abordarea End-to-End a dezvoltării - de la analiza cerințelor la deployment și testare - a format o perspectivă holistică asupra ciclului de viață al produselor software.

9 Planificarea Activitatii

9.1 Cronologia dezvoltării proiectului

9.1.1 Faza de analiză și proiectare

Analiza cerințelor și studiul de piață

- Analiza platformelor existente (LeetCode, Codecademy, HackerRank)
- Identificarea problemelor și oportunităților din piață
- Definirea obiectivelor și a grupului țintă
- Elaborarea listei de cerințe funcționale și non-funcționale

Alegerea stack-ului tehnologic

- Compararea framework-urilor frontend (Next.js vs React vs Vue)
- Evaluarea soluțiilor backend (Firebase vs Supabase vs Node.js)
- Selecția serviciilor de execuție cod (Judge0 vs Sphere Engine)
- Finalizarea arhitecturii sistemului

Proiectarea detaliată

- Crearea diagramelor arhitecturale și a flow-urilor de date
- Proiectarea interfețelor utilizator și a experienței UX
- Definirea structurii bazei de date și a modelelor de date

9.1.2 Faza de implementare

- Configurarea proiectului Next.js și setup-ul toolchain-ului
- Implementarea sistemului de autentificare cu Firebase
- Implementarea paginii de probleme și integrării cu Firestore
- Dezvoltarea playground-ului cu Monaco Editor
- Integrarea Judge0 API pentru execuția codului
- Implementarea sistemului de progress tracking
- Integrarea asistentului virtual VoiceGlow
- Dezvoltarea paginii Learn cu conținut educational

9.2 Dificultăți întâmpinate și soluții

Integrarea Judge0 API

- Problema: Timeout-uri frecvente și gestionarea erorilor inconsistentă.

- Soluția: Implementarea unui sistem robust de polling cu retry logic și error handling granular.

Gestionarea stării complexe în Playground

- Problema: Sincronizarea între Monaco Editor, rezultatele execuției și progresul utilizatorului.
- Soluția: Restructurarea state management-ului cu useEffect optimizat și funcții de cleanup.

Persistența codului între sesiuni

- Problema: Conflicte între localStorage și hydration în Next.js.
- Soluția: Implementarea lazy initialization și verificări de compatibilitate browser.

10 Bibliografie

- [1.1] Codecademy. "Pricing." Codecademy, 2025. <https://www.codecademy.com/pricing>. Accesat în 7 iunie 2025.
- [1.2] Codecademy. "Paid Plans - Codecademy Student Discount." Codecademy, 2025. <https://www.codecademy.com/pages/paid-plans>. Accesat în 7 iunie 2025.
- [1.3] Codecademy. "Learn to Code - for Free." Codecademy, 2025. <https://www.codecademy.com/>. Accesat în 7 iunie 2025.
- [1.4] "Codecademy." Wikipedia, 8 aprilie 2025. <https://en.wikipedia.org/wiki/Codecademy>. Accesat în 7 iunie 2025.
- [1.5] HackerRank. "Pricing - Programming problems and challenges." HackerRank, 2025. <https://www.hackerrank.com/work/pricing>. Accesat în 7 iunie 2025.
- [1.6] HackerRank. "Online Coding Tests and Technical Interviews." HackerRank, 2025. <https://www.hackerrank.com>. Accesat în 7 iunie 2025.
- [1.7] "HackerRank Pricing, Alternatives & More 2025." Capterra, 2025. <https://www.capterra.com/p/143549/HackerRank/>. Accesat în 7 iunie 2025.
- [1.8] "LeetCode." Wikipedia, 24 mai 2025. <https://en.wikipedia.org/wiki/LeetCode>. Accesat în 7 iunie 2025.
- [1.9] LeetCode. "LeetCode Premium." LeetCode, 2025. <https://leetcode.com/subscribe/>. Accesat în 7 iunie 2025.
- [1.10] "Is it worth it to buy LeetCode Premium?" Design Gurus, 2025. <https://www.designgurus.io/answers/detail/is-it-worth-it-to-buy-leetcode-premium>. Accesat în 7 iunie 2025.
- [1.11] Abiodun, E.I., Jantan, A., Omolara, A.E., Dada, K.V., Mohamed, N.A. și Arshad, H. "A review on effective approach to teaching computer programming to undergraduates in developing countries." Heliyon, vol. 8, nr. 9, 2022. <https://www.sciencedirect.com/science/article/pii/S2468227622001478>. Accesat în 7 iunie 2025
- [1.12] Tsarev, R., Krasnikhin, A., Chernyshev, A. și Chernyshev, D. "The use of online coding platforms as additional distance tools in programming education." ResearchGate, martie 2021. <https://www.researchgate.net/publication/350210999>. Accesat în 7 iunie 2025
- [1.13] Zhang, Y., Li, M., Wang, X. "Would ChatGPT-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? An empirical study." International Journal of Educational Technology in Higher Education, 2024. <https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-024-00446-5>. Accesat în 7 iunie 2025
- [1.14] Grover, S. și Pea, R. "Learning to code or coding to learn? A systematic review." Computers & Education, vol. 129, 2018, pp. 29-45. <https://www.sciencedirect.com/science/article/abs/pii/S0360131518302768>. Accesat în 7 iunie 2025
- [1.15] WBS Coding School. "What is tutorial hell and how to get out." 15 iunie 2022. Accesat în 7 iunie 2025 <https://www.wbscodingschool.com/blog/what-is-tutorial-hell-how-to-get-out/>. Accesat în 7 iunie 2025
- [1.16] Zero To Mastery. "How I escaped Tutorial Hell." 24 iunie 2022. <https://zerotomastery.io/blog/how-i-escaped-tutorial-hell/>. Accesat în 7 iunie 2025
- [1.17] DEV Community. "How to Escape Tutorial Hell." 9 martie 2023. <https://dev.to/ikechukwu/how-to-escape-tutorial-hell-4pi2>. Accesat în 7 iunie 2025
- [1.18] GeeksforGeeks. "C Programming Language Tutorial." 10 iunie 2024. <https://www.geeksforgeeks.org/c-programming-language/>. Accesat în 7 iunie 2025
- [1.19] W3Schools. "C Tutorial." 2025. <https://www.w3schools.com/c/>. Accesat în 7 iunie 2025.

[1.20] Programiz. “Learn C Programming.” 2025. <https://www.programiz.com/c-programming> . Accesat în 7 iunie 2025.

[1.21] https://www.reddit.com/r/learnprogramming/comments/qrlx5mwhat_exactly_is_tutorial_hell/

[1.22] <https://nextjs.org/docs> . Accesat în 6 iunie 2025

[2.1] - <https://www.hackerrank.com/> . Accesat în 6 iunie 2025

[2.2] - <https://leetcode.com/> - Accesat în 6 iunie 2025

[NEW_REF_1] GeeksforGeeks. “Memory Management in C”. Disponibil la: <https://www.geeksforgeeks.org/memory-management-in-c/> . Accesat la: 10 iunie 2025.

[NEW_REF_2] Programiz. “C Programming Tutorial”. Disponibil la: <https://www.programiz.com/c-programming> . Accesat la: 10 iunie 2025.

[NEW_REF_3] Stack Overflow. “Memory Management Questions”. Disponibil la: <https://stackoverflow.com/questions/tagged/memory-management+c> . Accesat la: 10 iunie 2025.

[NEW_REF_4] GeeksforGeeks. “Dynamic Memory Allocation in C”. Disponibil la: <https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/> . Accesat la: 10 iunie 2025.

[NEW_REF_5] Programiz. “C Standard Library Functions”. Disponibil la: <https://www.programiz.com/c-programming/library-function> . Accesat la: 10 iunie 2025.

[NEW_REF_6] Stack Overflow. “Learning C Programming”. Disponibil la: <https://stackoverflow.com/questions> . Accesat la: 10 iunie 2025.

[NEW_REF_7] Computer Language Benchmarks Game. “Language Performance Comparison”. Disponibil la: <https://benchmarksgame-team.pages.debian.net/benchmarksgame/> . Accesat la: 10 iunie 2025.

[NEW_REF_8] IBM Developer. “C Programming Language”. Disponibil la: <https://developer.ibm.com/technologies/c/> . Accesat la: 10 iunie 2025.

[NEW_REF_9] ACM Digital Library. “Computer Science Education”. Disponibil la: <https://dl.acm.org/topic/ccs2012/10010405.10010489.10010490> . Accesat la: 10 iunie 2025.

[NEW_REF_10] IEEE Xplore. “Programming Education Research”. Disponibil la: <https://ieeexplore.ieee.org/browse/periodicals/title> . Accesat la: 10 iunie 2025.

[NEW_REF_11] SAGE Journals. “Educational Technology Systems”. Disponibil la: <https://journals.sagepub.com/home/ets> . Accesat la: 10 iunie 2025.

[NEW_REF_12] ResearchGate. “Computer Science Research”. Disponibil la: <https://www.researchgate.net/topic/Computer-Science> . Accesat la: 10 iunie 2025.

[NEW_REF_13] Taylor & Francis Online. “Computer Science Education”. Disponibil la: <https://www.tandfonline.com/toc/ncse20/current> . Accesat la: 10 iunie 2025.

[NEW_REF_14] EDUCAUSE. “Educational Technology Research”. Disponibil la: <https://www.educause.edu/research> . Accesat la: 10 iunie 2025.

[NEW_REF_15] EDUCAUSE. “Higher Education IT”. Disponibil la: <https://www.educause.edu/> . Accesat la: 10 iunie 2025.

[NEW_REF_16] Software Testing Fundamentals. “Test Case Design”. Disponibil la: <https://softwaretestingfundamentals.com/test-case-design/> . Accesat la: 10 iunie 2025.

[NEW_REF_17] CP-Algorithms. “Competitive Programming”. Disponibil la: <https://cp-algorithms.com/> . Accesat la: 10 iunie 2025.

[NEW_REF_18] ACM Digital Library. “Computing Education Research”. Disponibil la: <https://dl.acm.org/> . Accesat la: 10 iunie 2025.

[NEW_REF_19] Springer Link. “Educational Psychology”. Disponibil la: <https://link.springer.com/journal/10648> . Accesat la: 10 iunie 2025.

[NEW_REF_20] Taylor & Francis Online. “Journal of Computer Science Education”. Disponibil la: <https://www.tandfonline.com/toc/ncse20/current> . Accesat la: 10 iunie 2025.

[NEW_REF_21] ScienceDirect. "Computers & Education". Disponibil la: <https://www.sciencedirect.com/journal/computers-and-education> . Accesat la: 10 iunie 2025.

[NEW_REF_22] Springer Link. "Educational Technology Research". Disponibil la: <https://link.springer.com/journal/11423> . Accesat la: 10 iunie 2025.

[NEW_REF_23] ACM Digital Library. "Transactions on Computing Education". Disponibil la: <https://dl.acm.org/journal/toce> . Accesat la: 10 iunie 2025.

[NEW_REF_24] DEV Community. "Programming Community". Disponibil la: <https://dev.to/> . Accesat la: 10 iunie 2025.

[NEW_REF_27] Firebase Documentation. "Realtime Database". Disponibil la: <https://firebase.google.com/docs/database> . Accesat la: 10 iunie 2025.

[NEW_REF_28] Firebase Documentation. "Authentication". Disponibil la: <https://firebase.google.com/docs/auth> . Accesat la: 10 iunie 2025.

[NEW_REF_29] EDUCAUSE. "Cybersecurity Program". Disponibil la: <https://www.educause.edu/focus-areas-and-initiatives/policy-and-security/cybersecurity-program> . Accesat la: 10 iunie 2025.

[NEW_REF_30] IEEE Xplore. "Security and Privacy". Disponibil la: <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=8013> . Accesat la: 10 iunie 2025.

[NEW_REF_31] Firebase Documentation. "Cloud Firestore". Disponibil la: <https://firebase.google.com/docs/firestore> . Accesat la: 10 iunie 2025.

[NEW_REF_32] MongoDB Documentation. "Data Modeling". Disponibil la: <https://docs.mongodb.com/manual/core/data-modeling/> . Accesat la: 10 iunie 2025.

[NEW_REF_33] ACM Computing Surveys. "Database Systems". Disponibil la: <https://dl.acm.org/journal/csur> . Accesat la: 10 iunie 2025.

[NEW_REF_37] Taylor & Francis Online. "Educational Technology Research". Disponibil la: <https://www.tandfonline.com/toc/uerd20/current> . Accesat la: 10 iunie 2025.

[NEW_REF_38] APA PsycNet. "Educational Psychology". Disponibil la: <https://psycnet.apa.org/> . Accesat la: 10 iunie 2025.

[NEW_REF_39] ScienceDirect. "Computers & Education". Disponibil la: <https://www.sciencedirect.com/journal/computers-and-education> . Accesat la: 10 iunie 2025.

[NEW_REF_40] Judge0. "API Documentation". Disponibil la: <https://ce.judge0.com/> . Accesat la: 10 iunie 2025.

[NEW_REF_41] AWS Documentation. "Lambda Security". Disponibil la: <https://docs.aws.amazon.com/lambda/latest/dg/lambda-security.html> . Accesat la: 10 iunie 2025.

[NEW_REF_42] Martin Fowler. "Microservices". Disponibil la: <https://martinfowler.com/articles/microservices.html> . Accesat la: 10 iunie 2025.

[NEW_REF_43] Taylor & Francis Online. "Educational Technology Research and Development". Disponibil la: <https://www.tandfonline.com/toc/uerd20/current> . Accesat la: 10 iunie 2025.

[NEW_REF_44] APA PsycNet. "Journal of Educational Psychology". Disponibil la: <https://psycnet.apa.org/PsycARTICLES/journal/edu> . Accesat la: 10 iunie 2025.

[NEW_REF_45] ScienceDirect. "Computers & Education". Disponibil la: <https://www.sciencedirect.com/journal/computers-and-education> . Accesat la: 10 iunie 2025.

[NEW_REF_46] IEEE Computer Society. "Software Engineering Body of Knowledge". Disponibil la: <https://www.computer.org/education/bodies-of-knowledge/software-engineering> . Accesat la: 10 iunie 2025.

[NEW_REF_47] ACM Digital Library. "Software Engineering Research". Disponibil la: <https://dl.acm.org/topic/ccs2012/10011007.10011074> . Accesat la: 10 iunie 2025.

[NEW_REF_48] Springer Link. "Requirements Engineering Journal". Disponibil la: <https://link.springer.com/journal/766> . Accesat la: 10 iunie 2025.

- [3.1] GeeksforGeeks. “C Programming Language Tutorial.” 10 iunie 2024. <https://www.geeksforgeeks.org/c-programming-language/> . Accesat în 8 iunie 2025.
- [3.2] Software Engineering Stack Exchange. “Is learning C essential for Computer Science?” <https://softwareengineering.stackexchange.com/questions/210737/is-learning-c-essential-for-computer-science> . Accesat în 8 iunie 2025.
- [3.3] GeeksforGeeks. “Why learning C Programming is a must?” 22 august 2024. <https://www.geeksforgeeks.org/why-learning-c-programming-is-a-must/> . Accesat în 8 iunie 2025.
- [3.4] StudySmarter. “C Programming Language: Examples, History & Advantages.” <https://www.studysmarter.co.uk/explanations/computer-science/computer-programming/c-programming-language/> . Accesat în 8 iunie 2025.
- [3.5] edX. “Learn C programming.” <https://www.edx.org/learn/c-programming> . Accesat în 8 iunie 2025.
- [3.6] Computer Science Stack Exchange. “What programming languages always learned in computer science B.Sc.?” <https://cs.stackexchange.com/questions/116926/what-programming-languages-always-learned-in-computer-science-b-sc> . Accesat în 8 iunie 2025.
- [3.7] GeeksforGeeks. “C Programming Course Online - Learn C with Data Structures.” <https://www.geeksforgeeks.org/courses/c-Programming-basic-to-advanced> . Accesat în 8 iunie 2025.
- [3.8] GeeksforGeeks. “10 Best C Programming Courses For Beginners [2025].” 21 ianuarie 2025. <https://www.geeksforgeeks.org/best-c-programming-courses-for-beginners/> . Accesat în 8 iunie 2025.
- [3.9] Yale University. “Computer Science.” <https://catalog.yale.edu/ycps/subjects-of-instruction/computer-science/> . Accesat în 8 iunie 2025.
- [3.10] UC Berkeley Extension. “Introduction to C Language Programming – EL ENG X24.” <https://extension.berkeley.edu/search/publicCourseSearchDetails.do?method=load&courseId=40911> . Accesat în 8 iunie 2025.
- [3.11] Codecademy. “Learn C: Pointers and Memory.” <https://www.codecademy.com/learn/learn-c-pointers-and-memory> . Accesat în 8 iunie 2025.
- [3.12] W3Schools. “C Pointers.” https://www.w3schools.com/c/c_pointers.php . Accesat în 8 iunie 2025.
- [3.13] Programming Lies. “Memory management is too hard for you.” 7 septembrie 2023. <https://www.bytesbeneath.com/p/manual-memory-management-is-too-hard> . Accesat în 8 iunie 2025.
- [3.14] Educated Guesswork. “Understanding Memory Management, Part 1: C.” <https://educatedguesswork.org/posts/memory-management-1/> . Accesat în 8 iunie 2025.
- [3.15] Learning C with Pebble. “Chapter 8: Pointers and Memory Allocation.” <https://pebble.gitbooks.io/learning-c-with-pebble/content/chapter08.html> . Accesat în 8 iunie 2025.
- [3.16] GeeksforGeeks. “Dynamic Memory Allocation in C using malloc(), calloc(), free() and realloc().” 13 decembrie 2018. <https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/> . Accesat în 8 iunie 2025.
- [3.17] Coursera. “C Programming: Pointers and Memory Management - 4.” <https://www.coursera.org/learn/c-programming-pointers-and-memory-management> . Accesat în 8 iunie 2025.
- [3.18] GeeksforGeeks. “C Pointers.” 15 decembrie 2016. <https://www.geeksforgeeks.org/c-pointers/> . Accesat în 8 iunie 2025.
- [3.19] DEV Community. “Understanding Memory Management, Pointers,

and Function Pointers in C.” 5 iulie 2024. <https://dev.to/emanuelgustafzon/understanding-memory-management-pointers-and-function-pointers-in-c-8ld> . Accesat în 8 iunie 2025.

[3.20] W3Schools. “C Memory Management (Memory Allocation).” https://www.w3schools.com/c/c_memory_management.php . Accesat în 8 iunie 2025.

[3.21] Online Learning vs Traditional Learning. <https://potomac.edu/learning/online-learning-vs-traditional-learning> . Accesat în 9 iunie 2025

[3.22] Comparing Learning Outcomes between Traditional Classroom Teaching and Online Learning Environments in Programming. https://www.researchgate.net/publication/386098211_Comparing_Learning_Outcomes_between_Traditional_Classroom_Teaching_and_Online_Learning_Environments_in_Programming_Education . Accesat în 9 iunie 2025

[3.23] Title: Traditional Learning Vs. Online Learning. <https://elearningindustry.com/traditional-learning-vs-online-learning> . Accesat în 9 iunie 2025

[3.24] Traditional Learning vs Online Learning. <https://pinlearn.com/traditional-learning-vs-online-learning> . Accesat în 9 iunie 2025

[3.25] eLearning Or Traditional Classroom Learning? Exploring The Pros And Cons. <https://elearningindustry.com/elearning-or-traditional-classroom-learning-exploring-the-pros-and-cons> . Accesat în 9 iunie 2025

[3.26] 9 Reasons Why eLearning Outshines Traditional Education Methods – Sensei LMS. <https://senseilms.com/advantages-of-e-learning/> . Accesat în 9 iunie 2025

[3.27] The Impact and Effectiveness of E-Learning on Teaching and Learning. <https://files.eric.ed.gov/fulltext/ED613533.pdf> Accesat în 9 iunie 2025

[3.21] MDPI. “Automated Code Assessment for Education: Review, Classification and Perspectives on Techniques and Tools.” <https://www.mdpi.com/2674-113X/1/1/2> . Accesat în 8 iunie 2025.

[3.22] Md. Mostafizer Rahman et al. “Exploring Automated Code Evaluation Systems and Resources for Code Analysis: A Comprehensive Survey.” arXiv:2307.08705, 8 iulie 2023. <https://arxiv.abs/2307.08705> . Accesat în 8 iunie 2025.

[3.23] GitHub. “judge0/judge0: The most advanced open-source online code execution system in the world.” <https://github.com/judge0/judge0> . Accesat în 8 iunie 2025.

[3.24] ACM Computing Surveys. “A Survey on Online Judge Systems and Their Applications.” <https://dl.acm.org/doi/10.1145/3143560> . Accesat în 8 iunie 2025.

[3.25] SpringerLink. “An Intelligent Online Judge System for Programming Training.” https://link.springer.com/chapter/10.1007/978-3-030-59419-0_57 . Accesat în 8 iunie 2025.

[3.26] ACM Transactions on Computing Education. “Automated Assessment in Computer Science Education: A State-of-the-Art Review.” <https://dl.acm.org/doi/10.1145/3513140> . Accesat în 8 iunie 2025.

[3.27] ResearchGate. “Building a Comprehensive Automated Programming Assessment System.” 28 aprilie 2020. https://www.researchgate.net/publication/340985812_Building_a_Comprehensive_Automated_Programming_Assessment_System . Accesat în 8 iunie 2025.

[3.28] ERIC. “Using Case-Based Reasoning to Improve the Quality of Feedback Provided by Automated Assessment Systems for Programming Exercises.” 2017. <https://eric.ed.gov/?q=e-learning&pg=194&id=ED577657> . Accesat în 8 iunie 2025.

[3.29] Aizu Online Judge. “ITP1_3_B: Print Test Cases.” https://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=ITP1_3_B . Accesat în 8 iunie 2025.

[3.30] Observable. “Online Judge/ Computer Programming/ Hidden Testcases.” 26 iulie 2020. <https://observablehq.com/@jyotisunkara/online-judge-computer-programming-hidden-testcases> . Accesat în 8 iunie 2025.

- [3.31] ScienceDirect. “TESTed—An educational testing framework with language-agnostic test suites for programming exercises.” <https://www.sciencedirect.com/science/article/pii/S2352711023001000> . Accesat în 8 iunie 2025.
- [4.1] DEV Community. “How to Escape Tutorial Hell.” 9 martie 2023. <https://dev.to/ikechukwu/how-to-escape-tutorial-hell-4pi2> . Accesat în 9 iunie 2025.
- [4.2] Firebase. “Firebase Authentication Documentation.” Google, 2025. <https://firebase.google.com/docs/auth> . Accesat în 9 iunie 2025.
- [4.3] Monaco Editor. “Monaco Editor Documentation.” Microsoft, 2025. <https://microsoft.github.io/monaco-editor/> . Accesat în 9 iunie 2025.
- [4.4] Judge0. “Judge0 API Documentation.” 2025. <https://ce.judge0.com/> . Accesat în 9 iunie 2025.
- [4.5] Google. “Web Vitals.” Google Developers, 2025. <https://web.dev/vitals/> . Accesat în 9 iunie 2025.
- [4.6] European Union. “General Data Protection Regulation (GDPR).” 25 mai 2018. <https://gdpr.eu/> . Accesat în 9 iunie 2025.
- [4.7] W3C. “Web Content Accessibility Guidelines (WCAG) 2.1.” 5 iunie 2018. <https://www.w3.org/WAI/WCAG21/quickref/> . Accesat în 9 iunie 2025.
- [4.8] ACM. “Computing Education Research.” ACM Transactions on Computing Education, 2024. <https://dl.acm.org/journal/toce> . Accesat în 9 iunie 2025.
- [4.9] Vercel. “Next.js Architecture Guide.” Vercel Inc., 2025. <https://nextjs.org/docs/architecture> Accesat în 9 iunie 2025.
- [4.10] Next.js. “App Router Documentation.” Vercel Inc., 2025. <https://nextjs.org/docs/app> . Accesat în 9 iunie 2025.
- [4.11] Firebase. “Firebase Documentation.” Google, 2025. <https://firebase.google.com/docs> . Accesat în 9 iunie 2025.
- [4.12] React. “React Documentation.” Meta Platforms Inc., 2025. <https://react.dev/> . Accesat în 9 iunie 2025.
- [4.13] React. “React Hooks Documentation.” Meta Platforms Inc., 2025. <https://react.dev/reference/react> . Accesat în 9 iunie 2025.
- [4.14] Firebase. “Firebase Architecture Guide.” Google, 2025. <https://firebase.google.com/docs/guides> . Accesat în 10 iunie 2025.
- [4.15] Firebase. “Security Rules Documentation.” Google, 2025. <https://firebase.google.com/docs/rules> . Accesat în 10 iunie 2025.
- [4.16] Judge0. “Judge0 Documentation.” Judge0 Technologies, 2025. <https://ce.judge0.com/> . Accesat în 9 iunie 2025.
- [4.17] React. “React Design Principles.” Meta Platforms Inc., 2025. <https://react.dev/learn/thinking-in-react> . Accesat în 9 iunie 2025.
- [4.18] Google. “Web Performance Best Practices.” Google Developers, 2025. <https://web.dev/performance/> . Accesat în 9 iunie 2025.
- [5.1] Brown, T. (2019). Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation. Harper Business.
- [5.2] Google Cloud. (2024). Firestore Documentation. Disponibil la: <https://cloud.google.com/firestore/docs> . Accesat în 13 iunie 2025.
- Nielsen Norman Group. (2024). UX Design for Learning Platforms. NN/g Research.
- [5.3] Stack Overflow. (2024). Developer Survey Results. Disponibil la: <https://survey.stackoverflow.co> Accesat în 13 iunie 2025.
- Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2), 257-285.
- [5.4] Tidwell, J., Brewer, C., & Valencia, A. (2020). Designing Interfaces: Patterns for Effective Interaction Design (3rd ed.). O'Reilly Media. .

[6.1] Google Developers. “Core Web Vitals - Essential metrics for a healthy site,” Google, 2024. [Online]. <https://web.dev/vitals/> . Accesat în 15 iunie 2025.

[6.2] DebugBear. “Website Performance Testing and Monitoring,” 2024. [Online]. <https://www.debugbear.com/> . Accesat în 15 iunie 2025.

[6.3] Firebase Team. “Firebase Documentation,” Google Cloud, 2024. [Online]. <https://firebase.google.com/docs> . Accesat în 13 iunie 2025.

[6.4] Judge0 Team. “Judge0 CE API Documentation,” 2024. [Online]. <https://ce.judge0.com/> Accesat în 13 iunie 2025.

[6.5] Vercel Inc. “Next.js 15 Documentation,” 2024. [Online]. <https://nextjs.org/docs> . Accesat în 14 iunie 2025.

[6.6] Microsoft Corporation. “Monaco Editor,” 2024. [Online]. <https://microsoft.github.io/monaco-editor/> . Accesat în 14 iunie 2025.

[6.7] TailwindCSS Team. “Utility-First CSS Framework,” 2024. [Online]. <https://tailwindcss.com/> . Accesat în 14 iunie 2025.

[6.8] TypeScript Team. “TypeScript Documentation,” Microsoft, 2024. [Online]. <https://www.typescriptlang.org/> . Accesat în 14 iunie 2025.