



Universitatea
Transilvania
din Brașov

FACULTATEA DE MATEMATICĂ
ȘI INFORMATICĂ

LUCRARE DE LICENȚĂ

FamousAuctions

Aplicație de gestionare a licitațiilor online

Conducător științific:

Lect. dr. Vlad Monescu

Absolvent:

Ploșniță Valentin-Gabriel

BRAȘOV, 2025

Cuprins

1	Introducere	5
1.1	Contextul actual și importanța temei	5
1.2	Motivația alegerii temei	5
1.3	Obiectivele lucrării	5
1.4	Structura lucrării	6
1.5	Elemente de originalitate	6
2	Abordarea temei	8
2.1	Platforme de licitații existente – analiză comparativă	8
2.2	Necesitatea unei soluții personalizate	8
2.3	Cazuri de utilizare și domenii de aplicabilitate	9
2.4	Justificarea alegerii abordării propuse	9
3	Noțiuni teoretice și tehnologii folosite	10
3.1	ASP.NET Core	10
3.2	C#	11
3.3	.NET 9	11
3.4	PostgreSQL	12
3.5	Entity Framework Core (Code First + Migrations)	13
3.6	JWT (JSON Web Token) + Cookie-uri securizate	14
3.7	LINQ (Language Integrated Query)	14
3.8	Stripe API și Hangfire	15
3.9	MSTest	15
3.10	Swagger și Postman	16
3.11	Azure DevOps și fișiere YAML	17
3.12	Git și GitHub	17
3.13	Blazor	18
3.14	HTML	18
3.15	CSS	19
3.16	JavaScript	19
3.17	Bootstrap	20
3.18	MudBlazor	20
3.19	SignalR	21
3.20	Design pattern-uri și principii SOLID utilizate	22
4	Arhitectura aplicației	24
4.1	Arhitectura generală a aplicației	24

4.2	Modelul de date și schema bazei de date	25
4.3	Arhitectura aplicației API pe straturi	29
4.4	Autentificare și autorizare	32
4.5	Arhitectură frontend	34
4.6	Arhitectura și integrarea serviciilor Stripe	35
4.7	Integrarea SignalR pentru actualizări în timp real	38
4.8	Deploy în cloud și livrare continuă (CI/CD)	40
5	Contribuții personale	43
5.1	Modulul original de livrări și dashboard-ul dedicat șoferilor	43
5.2	Sistem de tranzacții automate securizate prin Stripe	44
5.3	Experiență real-time cu notificări bidirecționale prin SignalR	44
5.4	Automatizarea livrării și deploy-ul în cloud	44
6	Ghid de utilizare	46
6.1	Contul de User	47
6.1.1	Pagina de login pentru User	47
6.1.2	Pagina de înregistrare pentru User	47
6.1.3	Resetarea parolei	48
6.1.4	Dashboard pentru User	49
6.1.5	General Settings	50
6.1.6	Seller Settings	51
6.1.7	Customer Settings	53
6.1.8	Browse Auctions	55
6.1.9	Accesarea unei licitații	56
6.1.10	My Listings	58
6.1.11	Pagină pentru crearea unei licitații	59
6.1.12	My Bids	59
6.2	Contul de Admin	60
6.2.1	Pagina de login pentru Admin	60
6.2.2	Admins Dashboard	62
6.2.3	Pagină pentru crearea conturilor de Admin	63
6.2.4	Drivers Dashboard	64
6.2.5	Pagină pentru crearea conturilor de Driver	64
6.2.6	Categories Dashboard	65
6.2.7	Pagină pentru crearea unei categorii	66
6.2.8	Pagină pentru editarea unei categorii	66
6.2.9	Auctions Dashboard	67
6.2.10	Asignarea unui șofer pentru livrarea unei licitații	68
6.3	Contul de Driver	69
6.3.1	Pagina de login pentru Driver	69
6.3.2	Driver Dashboard	70
6.3.3	Pagină cu licitațiile asignate șoferului	70
6.3.4	Pagina de finalizare a livrării	71
7	Concluzii și perspective de dezvoltare	73
8	Bibliografie	75

Listă de figuri

4.1	Schema relațională a bazei de date	29
4.2	Pipeline Api	41
4.3	Pipeline Web App	42
6.1	Pagina principală	46
6.2	Pagina de login pentru useri	47
6.3	Pagina de register pentru useri	48
6.4	Pagina de recuperare a parolei	49
6.5	Pagina pentru resetarea parolei	49
6.6	User home page	50
6.7	Pagină de setări generale pentru utilizator	51
6.8	Formular pentru crearea contului de vânzător	52
6.9	Detaliile contului de vânzător	52
6.10	Pagina de editare a detaliilor contului de vânzător	53
6.11	Tab cu detaliile metodei curente de plată	54
6.12	Tab cu formularul pentru adăugarea metodei de plată	55
6.13	Pagină cu licitații	56
6.14	Pagină pentru plasarea ofertelor	57
6.15	Pagină pentru plasarea ofertelor	58
6.16	Pagină cu licitațiile create de utilizator	58
6.17	Pagină pentru crearea unei licitații	59
6.18	Pagină cu licitațiile la care utilizatorul a licitat	60
6.19	Pagina de login pentru Admin	61
6.20	Dashboard pentru Admin	62
6.21	Admins Dashboard	63
6.22	Pagină pentru crearea conturilor de Admin	63
6.23	Drivers Dashboard	64
6.24	Pagină pentru crearea conturilor de Driver	65
6.25	Categories Dashboard	65
6.26	Pagină pentru crearea unei categorii	66
6.27	Pagină pentru editarea unei categorii	67
6.28	Auctions Dashboard	68
6.29	Pagină pentru asignarea șoferului	68
6.30	Pagină pentru asignarea șoferului	69
6.31	Pagina de login pentru Driver	69
6.32	Driver Dashboard	70
6.33	Pagină cu licitațiile asignate șoferului	71
6.34	Pagina de finalizare a livrării	71

6.35 Pagina de finalizare a livrării	72
--	----

Capitolul 1

Introducere

1.1 Contextul actual și importanța temei

În era digitală, comerțul electronic a cunoscut o expansiune semnificativă, iar platformele de licitații online au devenit un instrument esențial pentru tranzacții rapide, transparente și competitive. De la obiecte de colecție și produse electronice, până la servicii sau active imobiliare, licitațiile web permit vânzătorilor și cumpărătorilor să interacționeze eficient într-un cadru virtual securizat.

Popularitatea acestui model de afacere este susținută de creșterea încrederii în sistemele de plată online și de nevoia utilizatorilor de a găsi oferte personalizate și avantajoase. Pandemia COVID-19 a accelerat această tendință, impunând o digitalizare accelerată în multe domenii, inclusiv în tranzacțiile comerciale. În acest context, dezvoltarea unei aplicații de licitații web se aliniază perfect cerințelor actuale ale pieței și nevoii de soluții digitale eficiente.

1.2 Motivația alegerii temei

Motivul alegerii acestei teme se bazează pe dorința de a înțelege și implementa un sistem web real, care implică provocări complexe de natură tehnică, logică și arhitecturală. Aplicațiile de licitații presupun gestionarea concurentă a mai multor utilizatori, validarea în timp real a ofertelor, asigurarea integrității și securității tranzacțiilor și, nu în ultimul rând, o experiență plăcută pentru utilizatorul final.

Această lucrare îmbină aspectele teoretice studiate în cadrul facultății cu aplicabilitatea practică într-un proiect concret. Totodată, reprezintă o oportunitate de aprofundare a unor concepte moderne precum WebSocket, JWT, arhitectura RESTful, managementul sesiunilor și UX design.

1.3 Obiectivele lucrării

Principalul obiectiv al acestei lucrări este dezvoltarea unei aplicații web pentru licitații, care să permită:

- înregistrarea și autentificarea utilizatorilor,

- listarea de produse de către vânzători,
- participarea la licitații în timp real,
- validarea automată a ofertei câștigătoare,
- stocarea sigură a informațiilor și istoricului tranzacțiilor.

Obiectivul secundar este evidențierea procesului complet de dezvoltare software: de la analiza cerințelor, alegerea tehnologiilor potrivite, până la implementare, testare și prezentare a funcționalității aplicației.

1.4 Structura lucrării

Lucrarea este împărțită în mai multe capitole, după cum urmează:

- **Capitolul 1** – Introducere, prezintă contextul general al temei, motivația și obiectivele urmărite.
- **Capitolul 2** – Abordarea temei, include analiza pieței actuale și argumentarea alegerii soluției propuse.
- **Capitolul 3** – Noțiuni teoretice și tehnologii folosite, detaliază fundamentele teoretice și tehnologiile utilizate în proiect.
- **Capitolul 4** – Arhitectura aplicației, descrie structura logică, tehnologică și modul de organizare a componentelor software.
- **Capitolul 5** – Contribuții personale, evidențiază implementările realizate, deciziile tehnice și soluțiile propuse.
- **Capitolul 6** – Ghid de utilizare, explică modul de funcționare al aplicației din perspectiva utilizatorului final și a administratorilor.
- **Capitolul 7** – Concluzii și perspective de dezvoltare, sintetizează rezultatele obținute și propune direcții viitoare de extindere a aplicației.
- **Capitolul 8** – Bibliografia, listează sursele consultate și materialele de referință.

1.5 Elemente de originalitate

Lucrarea de față propune o arhitectură completă pentru o aplicație de licitații web care nu se limitează doar la desfășurarea propriu-zisă a licitațiilor, ci integrează și un modul de livrare post-tranzacție, extinzând astfel lanțul funcțional al platformei.

Un prim element de originalitate îl reprezintă sistemul de roluri multiple, fiecare cu interfețe și responsabilități distincte:

- **Utilizatori normali** care pot licita și adăuga produse spre vânzare;
- **Administratori**, care gestionează licitațiile, utilizatorii și procesele post-vânzare;
- **Super-administratori**, care au drepturi extinse asupra platformei;
- **Șoferi**, responsabili de livrarea fizică a obiectelor adjudecate.

După încheierea unei licitații și desemnarea câștigătorului, un admin poate asigna un șofer pentru a efectua livrarea obiectului. Aplicația gestionează această operațiune în mod controlat, asociind fiecare tranzacție unui utilizator desemnat pentru livrare.

Șoferii beneficiază de un *dashboard* dedicat, în care au acces la:

- lista obiectelor alocate pentru livrare;
- datele relevante despre vânzător și cumpărător;
- informațiile esențiale despre produs (nume, locație, status livrare etc.).

Această abordare adaugă un modul logistic integrat într-o aplicație de licitații web, o funcționalitate rar întâlnită în platformele similare și extrem de utilă pentru gestionarea completă a unui flux de licitație. Sistemul este astfel conceput încât să poată fi extins cu funcționalități precum urmărirea livrărilor în timp real, generarea automată a AWB-urilor sau integrarea cu servicii externe de curierat.

În plus, platforma este construită cu accent pe modularitate, securitate și extensibilitate, permițând în viitor integrarea de funcții avansate precum notificări în timp real, evaluarea livrărilor sau gestionarea ratingurilor pentru șoferi.

Capitolul 2

Abordarea temei

2.1 Platforme de licitații existente – analiză comparativă

În ultimii ani, platformele de licitații online au evoluat de la simple forumuri de vânzare către ecosisteme digitale complexe, cu integrare de plăți, sisteme de reputație și livrare logistică. Printre cele mai cunoscute exemple se numără:

- **eBay** – probabil cea mai cunoscută platformă de licitații, oferă mecanisme solide pentru licitație, *buy-it-now*, rating-uri, protecție a cumpărătorului și integrare completă cu servicii de livrare. Cu toate acestea, este o platformă generalistă, aglomerată și uneori greu de personalizat pentru anumite nișe.
- **Licitatii.ro** – platformă românească destinată în principal achizițiilor publice sau bunurilor executate silit. Este mai tehnică și orientată spre B2B, cu interfețe mai puțin prietenoase pentru utilizatorii de rând.
- **AuctionZip** – platformă orientată către licitații live cu streaming și participare în timp real, dar care se bazează adesea pe infrastructură terță și nu integrează logistică post-tranzacție.

Aceste platforme acoperă majoritatea nevoilor de licitație, dar au și limitări evidente:

- Nu includ management logistic direct (ex: asignarea driverilor);
- Nu permit o personalizare fină a rolurilor de utilizatori;
- Nu oferă transparență operațională privind livrarea obiectelor;
- În general sunt centralizate și greu de adaptat pentru implementări particulare.

2.2 Necesitatea unei soluții personalizate

Având în vedere aceste neajunsuri, aplicația propusă își propune să acopere o nișă funcțională neacoperită: o platformă care gestionează nu doar procesul de licitație, ci și fluxul logistic imediat după adjudecare.

Printre avantajele unui astfel de sistem:

- Control complet asupra fluxului de licitație și livrare;
- Posibilitatea de a gestiona utilizatori cu roluri distincte și clare;
- Transparență și responsabilitate prin dashboarduri personalizate;
- O interfață adaptată la nevoile utilizatorilor finali și operatorilor logistici;
- Flexibilitate tehnologică pentru scalare sau integrare cu servicii externe (curierat, plăți, evaluări).

2.3 Cazuri de utilizare și domenii de aplicabilitate

Aplicația de licitații web cu modul logistic se poate aplica în multiple contexte:

- **Marketplace local** pentru produse second-hand între utilizatori din aceeași zonă;
- **Platforme de licitații caritabile**, unde transportul produselor e parte din proces;
- **Licitații B2B** (ex: lichidări de stocuri, echipamente), unde livrarea trebuie gestionată direct;
- **Licitații pentru servicii** (ex: lucrări, reparații), în care „obiectul” livrat este un serviciu realizat de o echipă mobilă (asimilabil cu rolul driverului).

2.4 Justificarea alegerii abordării propuse

Soluția propusă se diferențiază prin:

- Abordare **end-to-end**, de la postarea produsului, licitație, desemnarea câștigătorului, până la livrarea efectivă;
- Automatizare a asignării livrărilor de către administratori și informarea șoferilor prin dashboarduri clare;
- Utilizarea unor tehnologii moderne (*REST API, WebSocket, JWT*) care permit atât performanță ridicată, cât și scalabilitate și interoperabilitate cu alte servicii;
- Roluri granulare care permit segmentarea precisă a funcționalităților și a responsabilităților.

Prin aceste componente, aplicația nu este doar o clonă a platformelor existente, ci un ecosistem scalabil, care poate fi adaptat cu ușurință pentru piețe specifice, domenii verticale sau parteneriate comerciale.

Capitolul 3

Noțiuni teoretice și tehnologii folosite

3.1 ASP.NET Core

Introducere

ASP.NET Core este un framework open-source, dezvoltat de Microsoft, destinat construirii aplicațiilor web moderne, API-urilor REST și aplicațiilor de server performante. Lansat ca o versiune modulară și cross-platform a clasicului ASP.NET, acest framework reprezintă una dintre cele mai robuste și eficiente soluții pentru dezvoltarea aplicațiilor web enterprise, putând rula pe Windows, Linux și macOS.

ASP.NET Core este construit pe baza arhitecturii middleware pipeline și oferă funcționalități avansate precum routing personalizat, injectare de dependențe nativă, sisteme de autentificare/autorizare, integrare cu Entity Framework Core, suport pentru Web API și găzduire Kestrel performantă. Este complet interoperabil cu tehnologiile cloud moderne și susține practici DevOps precum CI/CD și containerizare.

Datorită performanței ridicate și a suportului pe termen lung oferit de Microsoft, ASP.NET Core este alegerea ideală pentru aplicații care necesită scalabilitate, securitate și mentenanță predictibilă în medii de producție complexe.

Rolul ASP.NET Core în aplicația de licitații

În cadrul aplicației de licitații web, ASP.NET Core constituie coloana vertebrală a componentei de backend (API-ul). Aici sunt implementate toate funcționalitățile critice care țin de logica de business, interacțiunea cu baza de date, validarea și procesarea datelor, securitatea accesului și gestionarea fluxurilor de lucru specifice aplicației.

Mai precis, ASP.NET Core este responsabil pentru:

- expunerea endpointurilor REST care pot fi apelate de interfața web construită cu Blazor;
- manipularea entităților precum Utilizator, Licitație, Ofertă, Livrare, Rol etc.;
- integrarea cu serviciul de plăți Stripe, prin servicii personalizate injectate în pipeline;
- utilizarea middleware-urilor pentru gestionarea autentificării pe bază de token JWT;

- validarea și autorizarea cererilor în funcție de rolul utilizatorului (Admin, SuperAdmin, Driver, User);
- controlul ciclului de viață al aplicației, inclusiv inițializarea bazei de date, configurarea serviciilor și alinierea cu pipeline-ul DevOps (prin integrare YAML).

Structura modulară a proiectului ASP.NET Core a permis separarea clară a serviciilor (Business Logic), controllerelor (Routing + API Contracts), și a accesului la date (Data Layer), în conformitate cu bunele practici de arhitectură software.

3.2 C#

Introducere

C# este un limbaj de programare modern, tipizat static și orientat pe obiect, dezvoltat de Microsoft ca parte a platformei .NET. De la lansarea sa în anul 2000, C# a evoluat constant, devenind unul dintre cele mai utilizate limbaje la nivel enterprise datorită expresivității, robusteții și suportului extins pentru dezvoltarea de aplicații web, desktop, mobile și cloud.

C# combină sintaxa clară și riguroasă inspirată din C/C++ și Java cu caracteristici avansate precum:

- programare asincronă (async/await),
- lambda expressions și LINQ pentru manipularea datelor,
- tuples, records, pattern matching și nullable reference types (în versiunile recente).

Prin integrarea nativă cu .NET Runtime, C# permite o interoperabilitate excelentă cu tehnologii moderne precum PostgreSQL, Azure, Blazor, ASP.NET, etc., și oferă un ecosistem bogat de librării și instrumente pentru dezvoltare profesională.

Rolul C# în aplicația de licitații

Limbajul C# a fost utilizat atât în partea de backend, cât și în frontendul Blazor, oferind o experiență de dezvoltare unificată, fără a fi nevoie de limbaje diferite pentru fiecare nivel al aplicației.

3.3 .NET 9

Introducere

.NET 9 este o versiune modernă și evoluată a platformei .NET, dezvoltată de Microsoft, care unifică toate tehnologiile .NET într-un singur framework. Începând cu .NET 5, Microsoft a demarat procesul de consolidare a variantelor .NET (Framework, Core și Mono) într-o platformă unificată, scalabilă și cross-platform, iar .NET 9 reprezintă una dintre cele mai stabile și performante iterații de până acum.

Această versiune aduce îmbunătățiri semnificative în domenii precum:

- performanța runtime-ului (CoreCLR);

- suportul extins pentru ARM64 și WebAssembly;
- optimizări în compilare JIT și AOT (Just-In-Time și Ahead-Of-Time);
- integrare naturală cu tehnologii moderne de cloud și DevOps.

De asemenea, .NET 9 continuă să dezvolte și să rafineze suportul pentru C# 12 și noile paradigme de programare declarativă, funcțională și reactivă.

Rolul .NET 9 în aplicația de licitații

Aplicația este construită în întregime folosind .NET 9, atât pe partea de API (ASP.NET Core), cât și pe partea de frontend (Blazor). Platforma a reprezentat nucleul pe care s-a bazat întreaga infrastructură de dezvoltare, compilare, rulare și publicare.

Contribuțiile principale ale .NET 9 în aplicație sunt:

- Cross-platform runtime: permite rularea aplicației atât local (pentru testare), cât și în cloud, în containere sau pe sisteme Linux/Windows;
- Performanță ridicată: aplicația gestionează mai multe tipuri de utilizatori (admin, user, driver, super admin) și procese concurente (licitații, plăți, notificări), fără a compromite viteza sau stabilitatea;
- Compatibilitate DevOps: integrarea cu Azure DevOps și pipeline-urile YAML este optimizată pentru .NET 9, ceea ce a permis livrarea rapidă și fiabilă a actualizărilor;
- Compatibilitate cu toate bibliotecile folosite: biblioteci externe precum Stripe SDK, Entity Framework Core, MudBlazor și Moq sunt complet compatibile cu această versiune .NET, fără nevoia de workaround-uri sau adaptări.

3.4 PostgreSQL

Introducere

PostgreSQL este un sistem de gestiune a bazelor de date relaționale open-source, recunoscut pentru conformitatea cu standardele SQL, extensibilitate și robustețe. Este utilizat pe scară largă în aplicații moderne care necesită integritate, performanță și flexibilitate în modelarea datelor.

Fiind un SGBD matur și stabil, PostgreSQL oferă:

- suport complet pentru tranzacții ACID (Atomicitate, Consistență, Izolare, Durabilitate);
- un limbaj SQL bogat și extensii pentru programare procedurală (PL/pgSQL);
- suport nativ pentru tipuri complexe (arrays, JSONB, hărți, enumuri);
- mecanisme solide de securitate: control pe baze de roluri, autentificare pe bază de certificat, criptare, audit;
- funcționalități avansate precum indexing performant (GIN, GiST), replicare, triggers, views, CTE-uri;

- compatibilitate excelentă cu ORMs precum Entity Framework Core prin providerul Npgsql.

Fiind cross-platform și open-source, PostgreSQL oferă independență tehnologică și o comunitate activă, ceea ce îl face potrivit pentru aplicații comerciale, academice și industriale.

Rolul PostgreSQL în aplicația de licitații

În cadrul platformei de licitații web, PostgreSQL a fost utilizat pentru a stoca toate datele esențiale într-o manieră relațională și coerentă. A fost ales datorită compatibilității directe cu Entity Framework Core și pentru performanța și stabilitatea oferită în scenarii complexe cu multe entități interconectate.

3.5 Entity Framework Core (Code First + Migrations)

Introducere

Entity Framework Core (EF Core) este un Object-Relational Mapper (ORM) modern, dezvoltat de Microsoft, care face parte din ecosistemul .NET. EF Core permite dezvoltatorilor să lucreze cu o bază de date relațională utilizând obiecte C# și clase, fără a scrie cod SQL direct pentru fiecare operațiune CRUD (Create, Read, Update, Delete).

Abordarea Code First din EF Core permite:

- definirea modelelor de date în codul C# (ca entități simple);
- generarea automată a structurii bazei de date pornind de la aceste modele;
- controlul și urmărirea modificărilor printr-un sistem de migrații;
- suport pentru LINQ ca metodă nativă de interogare.

EF Core oferă suport complet pentru lucrul cu PostgreSQL, permițând definirea modelelor de date într-un mod obiectual, maparea automată către structuri relaționale, gestionarea tranzacțiilor și configurarea migrațiilor. În cadrul acestei aplicații, PostgreSQL a fost integrat nativ cu EF Core, beneficiind de funcționalități precum mapări complexe, relații între entități, validări, încărcare lazy și eager, precum și controlul precis al evoluției bazei de date prin migrații.

Rolul EF Core în aplicația de licitații

În cadrul aplicației de licitații, Entity Framework Core a fost utilizat ca strat principal pentru accesul la date. S-a optat pentru abordarea Code First, ceea ce a permis o dezvoltare agilă, în paralel cu definirea modelelor de business în cod.

3.6 JWT (JSON Web Token) + Cookie-uri securizate

Introducere

JWT (JSON Web Token) este un standard deschis (RFC 7519) utilizat pe scară largă pentru schimbul securizat de informații între două părți, sub forma unui obiect compact și auto-conținut. Aceste token-uri sunt folosite în special în sistemele de autentificare stateless (fără sesiuni persistente pe server), cum ar fi API-urile moderne.

Un token JWT conține trei secțiuni codificate în Base64:

1. **Header:** definește algoritmul de semnare (ex: HS256);
2. **Payload:** conține datele despre utilizator (ID, roluri, email etc.);
3. **Signature:** este generată pe baza celorlalte două și a unei chei secrete, pentru a garanta integritatea.

JWT se remarcă prin:

- posibilitatea de a transporta date semnate și verificate;
- suport nativ în majoritatea frameworkurilor backend;
- compatibilitate cu HTTP Headers și cookie-uri.

Pentru a spori securitatea, este o practică recomandată ca JWT-urile să fie stocate în cookie-uri securizate de tip HTTP-only, care nu pot fi accesate din JavaScript, prevenind astfel atacurile de tip Cross-Site Scripting (XSS).

Rolul JWT și cookie-urilor în aplicația de licitații

În aplicația de licitații, autentificarea și autorizarea utilizatorilor sunt gestionate printr-un sistem bazat pe JWT.

3.7 LINQ (Language Integrated Query)

Introducere

LINQ (Language Integrated Query) este o componentă esențială a limbajului C#, introdusă în versiunea 3.0 a acestuia, care permite interogarea colecțiilor de date într-un mod declarat, expresiv și sigur din punct de vedere sintactic. LINQ unifică accesul la diverse surse de date — cum ar fi obiecte din memorie (LINQ to Objects), baze de date relaționale (LINQ to Entities), fișiere XML (LINQ to XML) sau chiar servicii — sub o singură sintaxă standardizată, integrată în limbaj.

Sintaxa LINQ oferă două forme:

- Query syntax (asemănătoare SQL);
- Method syntax (bazată pe funcții lambda și expresii fluent-chained).

Beneficiile LINQ includ:

- evitare erorilor de tip la compilare;
- intellisense și refactorizare completă în IDE-uri moderne;
- lizibilitate crescută a codului și eliminarea codului boilerplate repetitiv.

Rolul LINQ în aplicația de licitații

În cadrul aplicației tale, LINQ este utilizat pe scară largă în interacțiunile cu baza de date prin Entity Framework Core, dar și în manipularea colecțiilor de obiecte în memorie. LINQ permite scrierea clară și concisă a logicii de filtrare, sortare și proiectare a datelor, atât în backend, cât și în Blazor.

3.8 Stripe API și Hangfire

Introducere

Stripe este o platformă globală pentru procesarea plăților online, apreciată pentru ușurința integrării, gradul ridicat de securitate și fiabilitatea în scenarii de comerț electronic. Prin intermediul unui API REST bine documentat și a SDK-urilor oficiale, Stripe permite implementarea de fluxuri de plată complexe, incluzând suport pentru tranzacții cu cardul bancar, portofele electronice, transferuri directe și facturare recurentă.

În completare, Hangfire reprezintă un framework open-source specializat în executarea sarcinilor în fundal în aplicații ASP.NET Core. Acesta oferă un model de procesare asincronă bazat pe persistarea și planificarea sarcinilor într-o bază de date relațională. Printre funcționalitățile sale se numără: background jobs, scheduled jobs, recurring jobs și automatizarea retry-urilor pentru sarcini eșuate.

Rolul Stripe și Hangfire în aplicația de licitații

În aplicația dezvoltată, integrarea cu Stripe are rolul de a gestiona complet procesul de plată automată, care se declanșează imediat după încheierea unei licitații. Acest proces este automatizat prin intermediul unui job programat în Hangfire.

3.9 MSTest

Introducere

MSTest este framework-ul oficial de testare unitară dezvoltat de Microsoft pentru platforma .NET. Acesta permite scrierea, executarea și gestionarea testelor automate, fiind complet integrat în Visual Studio și compatibil cu sistemele de build automatizat, precum Azure DevOps și GitHub Actions.

Testarea automată este o practică esențială în dezvoltarea software modernă, asigurând:

- validarea corectitudinii logicii de business;
- detecția rapidă a regresiiilor;

- creșterea încrederii în codul scris;
- facilitarea mentenanței și extinderii aplicației.

Prin utilizarea testelor unitare, testelor de integrare și mock-urilor, pot fi simulate componentele externe și pot fi validate scenarii complexe în mod controlat, reproductibil și eficient.

Rolul MSTest în aplicația de licitații

În cadrul aplicației de licitații, MSTest a fost utilizat pentru validarea componentelor critice din ambele soluții: API-ul backend și interfața Blazor.

Testarea s-a desfășurat pe mai multe niveluri:

- Teste unitare: focalizate pe componente izolate (ex: servicii, validatori, helperi);
- Teste de integrare: axate pe interacțiunea între servicii, contextul EF Core și date reale dintr-o bază de date de test;
- Teste de validare a logicii de business: pentru scenarii precum determinarea câștigătorului unei licitații, actualizarea stării tranzacției, validarea sumelor etc.

Importanță și beneficii

Prin implementarea unei suite consistente de teste automate, aplicația capătă următoarele avantaje:

- siguranță în refactorizare, deoarece funcționalitatea este validată constant;
- detecția timpurie a erorilor, înainte de lansarea în producție;
- automatizare CI/CD, deoarece testele pot fi rulate automat în pipeline-urile Azure DevOps;
- documentarea comportamentului aplicației, întrucât testele definesc în mod clar așteptările funcționale.

3.10 Swagger și Postman

Introducere

În dezvoltarea și testarea serviciilor web, în special a celor de tip REST API, instrumentele care facilitează documentarea, explorarea și validarea endpointurilor sunt esențiale. Două dintre cele mai utilizate soluții în acest context sunt Swagger și Postman.

Swagger (parte a ecosistemului OpenAPI) este un set de unelte care permite descrierea, generarea și testarea interfețelor API printr-o specificație standardizată. În ASP.NET Core, integrarea Swagger este realizată prin biblioteca Swashbuckle, care generează automat o interfață web interactivă pentru toate endpointurile definite în aplicație.

Postman este o aplicație desktop și web folosită pentru testarea manuală a API-urilor. Aceasta oferă o interfață grafică intuitivă pentru trimiterea cererilor HTTP, inspectarea răspunsurilor, organizarea colecțiilor de endpointuri și rularea automată de scenarii prin testare scriptată.

Rolul Swagger și Postman în aplicația de licitații

Ambele unelte au fost integrate și utilizate complementar în dezvoltarea și testarea aplicației de licitații, fiecare având un rol distinct în diferite etape ale ciclului de viață al proiectului.

3.11 Azure DevOps și fișiere YAML

Introducere

Azure DevOps este o platformă dezvoltată de Microsoft care oferă un set complet de instrumente pentru planificarea, dezvoltarea, testarea și livrarea aplicațiilor software. Printre componentele sale principale se numără:

- Azure Repos – pentru controlul versiunilor (Git);
- Azure Pipelines – pentru automatizarea procesului de build, testare și deploy (CI/CD);
- Azure Boards, Test Plans, și Artifacts – pentru gestionarea proiectelor, testare manuală și livrarea de pachete.

Un aspect important al Azure Pipelines este suportul pentru definirea proceselor CI/CD prin fișiere YAML (Yet Another Markup Language), ceea ce permite o versiune declarativă, transparentă și reproductibilă a pașilor de build și deploy. Această abordare înlocuiește interfețele grafice tradiționale cu configurații versionate alături de codul sursă.

Rolul Azure DevOps și YAML în aplicația de licitații

În cadrul aplicației dezvoltate, ambele componente - atât API-ul ASP.NET Core, cât și aplicația Blazor - sunt livrate în cloud cu ajutorul pipeline-urilor automatizate definite în fișiere YAML și procesate de Azure DevOps.

3.12 Git și GitHub

Introducere

Git este un sistem de control al versiunilor distribuit, dezvoltat de Linus Torvalds în 2005, care a devenit standardul de facto în industria software pentru gestionarea codului sursă. Git oferă posibilitatea de a urmări modificările efectuate asupra fișierelor de-a lungul timpului, de a lucra în paralel pe ramuri diferite și de a reveni la stări anterioare în mod controlat.

Fiind un sistem distribuit, fiecare utilizator are o copie completă a repository-ului local, ceea ce permite lucrul offline, protecție împotriva pierderii de date și un flux eficient de colaborare.

GitHub, la rândul său, este o platformă cloud de găzduire a repository-urilor Git, care adaugă funcționalități sociale și colaborative precum:

- gestionarea problemelor (issues);
- pull requests și code review;
- acțiuni automate (GitHub Actions);

- vizualizare diferențe (diff);
- protecția ramurilor prin politici de integrare.

Rolul Git și GitHub în aplicația de licitații

În proiectul de licență, Git a fost utilizat pentru gestionarea controlată a codului sursă al celor două componente ale aplicației:

- API-ul ASP.NET Core (backend);
- aplicația Blazor (frontend).

Fiecare soluție a fost organizată într-un repository separat pe GitHub, facilitând separarea clară a responsabilităților și reducerea riscului de interferență între codurile celor două componente.

3.13 Blazor

Introducere

Blazor este un framework dezvoltat de Microsoft, care permite crearea de aplicații web interactive folosind limbajul C# în locul tradiționalului JavaScript. Numele „Blazor” provine din combinația „Browser” + „Razor” (motorul de template-uri din ASP.NET), reflectând capacitatea acestuia de a executa cod C# direct în browser prin intermediul WebAssembly sau pe server prin SignalR.

Există două modele principale de execuție:

- Blazor WebAssembly (WASM) – codul aplicației rulează complet în browser, într-un sandbox WebAssembly;
- Blazor Server – logica aplicației rulează pe server, iar actualizările UI sunt transmise în timp real prin conexiune SignalR.

Indiferent de modelul ales, Blazor oferă o experiență full-stack C#, permițând dezvoltatorilor .NET să construiască interfețe moderne fără a părăsi ecosistemul familiar.

Rolul Blazor în aplicația de licitații

În cadrul aplicației de licitații, componenta frontend este implementată integral folosind Blazor, în combinație cu Razor Components, MudBlazor, și JavaScript pentru funcționalități extinse.

3.14 HTML

Introducere

HTML (HyperText Markup Language) este limbajul standard de marcare utilizat pentru structurarea conținutului în paginile web. Acesta definește ierarhia și semantica elementelor vi-

zuale precum titluri, paragrafe, liste, butoane, tabele sau formulare. HTML este un pilon fundamental în dezvoltarea frontend, servind drept schelet pentru orice interfață web, indiferent de tehnologia utilizată pentru stilizare sau comportament dinamic.

Versiunea actuală – HTML5 – introduce numeroase elemente semantice suplimentare (<article>, <section>, <nav>, <footer>, etc.), suport pentru media nativă (<audio>, <video>) și o mai bună integrare cu JavaScript și CSS.

Rolul HTML în aplicația de licitații

În aplicația de licitații, HTML este utilizat extensiv în cadrul componentelor Razor din proiectul Blazor, fiind combinat cu directivele C# pentru a crea UI-uri dinamice și reactive. Fiecare componentă Blazor generează la final cod HTML, interpretat și redat în browser de către motorul de randare.

3.15 CSS

Introducere

CSS (Cascading Style Sheets) este limbajul standard utilizat pentru stilizarea paginilor web. Acesta separă conținutul (structurat prin HTML) de prezentare, permițând definirea aspectului vizual al elementelor UI: dimensiuni, poziționare, culori, fonturi, margini, tranziții, animații și comportament responsive. Prin sistemul său de moștenire (cascadare) și specificitate, CSS oferă un control detaliat asupra modului în care fiecare componentă este afișată într-un browser web.

CSS este compatibil cu toate tehnologiile moderne de frontend și este utilizat atât în aplicațiile clasice bazate pe HTML static, cât și în frameworkuri reactive precum Blazor, React, Angular sau Vue.

Rolul CSS în aplicația de licitații

În aplicația de licitații web, CSS este utilizat pentru a oferi un aspect modern, coerent și prietenos pentru utilizator.

3.16 JavaScript

Introducere

JavaScript este limbajul de programare standard pentru dinamizarea și interactivitatea aplicațiilor web. Este unul dintre cele trei piloni fundamentali ai dezvoltării web moderne, alături de HTML și CSS. JavaScript rulează nativ în toate browserele moderne și permite manipularea Document Object Model (DOM), gestionarea evenimentelor, comunicarea asincronă cu serverul (AJAX, fetch API), precum și integrarea cu biblioteci și frameworkuri moderne (ex: React, Vue, jQuery).

Deși frameworkuri precum Blazor permit dezvoltarea frontend exclusiv cu C#, JavaScript rămâne esențial în scenariile unde se dorește control fin asupra comportamentului interfeței, animații avansate sau interacțiuni directe cu API-urile browserului.

Rolul JavaScript în aplicația de licitații

În aplicația de licitații web, JavaScript este folosit complementar tehnologiei Blazor pentru a implementa funcționalități vizuale avansate și animații personalizate care nu pot fi obținute nativ doar cu C# sau Razor Components.

3.17 Bootstrap

Introducere

Bootstrap este un framework open-source de front-end dezvoltat inițial de Twitter, care oferă un set complet de componente UI predefinite și un sistem puternic de layout bazat pe grid. Scopul principal al Bootstrap este de a accelera dezvoltarea interfețelor moderne, responsive și coerente, reducând nevoia de a scrie CSS personalizat pentru fiecare componentă vizuală.

Bootstrap este construit pe baza limbajelor HTML, CSS și JavaScript, oferind componente preconstruite precum:

- butoane,
- formulare,
- meniuri de navigație,
- tabele,
- alerte,
- carusele,
- și multe altele.

De asemenea, oferă suport nativ pentru responsive design, asigurând afișarea optimă pe dispozitive de diferite dimensiuni (desktop, tabletă, mobil).

3.18 MudBlazor

Introducere

MudBlazor este o bibliotecă open-source de componente UI construită special pentru frameworkul Blazor. Bazată pe principiile designului Material Design dezvoltat de Google, MudBlazor oferă un set extins de componente moderne, optimizate pentru aplicații C# full-stack.

Spre deosebire de alte frameworkuri care necesită integrare cu JavaScript, MudBlazor este scris complet în C# și Razor, fiind complet compatibil cu ciclul de viață Blazor. Astfel, oferă o experiență nativă și fluidă în ecosistemul .NET, fără dependențe externe.

Rolul MudBlazor în aplicația de licitații

În cadrul aplicației de licitații, MudBlazor a fost adoptat pentru:

- crearea unor componente vizuale atractive și moderne;
- îmbunătățirea experienței de utilizare prin animații fluide și feedback vizual;
- integrarea rapidă a componentelor funcționale precum dialoguri, carduri, snackbars, tabele și meniuri de navigație.

3.19 SignalR

Introducere

SignalR este o bibliotecă dezvoltată de Microsoft pentru ASP.NET și ASP.NET Core, care facilitează comunicarea bidirecțională în timp real între client (browser) și server. Aceasta abstractizează complexitatea protocoalelor de comunicare web (WebSockets, Server-Sent Events, Long Polling), oferind o interfață simplificată și un model de evenimente pentru trimiterea de date de la server către mai mulți clienți simultan.

SignalR este ideal pentru aplicații care necesită sincronizare în timp real a stării aplicației, precum:

- chat-uri,
- tablouri de licitații,
- aplicații financiare live,
- dashboard-uri dinamice.

Rolul SignalR în aplicația de licitații

În cadrul aplicației de licitații, SignalR este utilizat pentru a oferi actualizare instantanee a ofertelor licitate într-o sesiune de licitație activă.

Scenariul principal este următorul:

1. mai mulți utilizatori accesează simultan o anumită licitație;
2. atunci când unul dintre aceștia plasează o ofertă, serverul procesează acțiunea;
3. prin intermediul unui hub SignalR, aplicația notifică toți ceilalți clienți conectați că a apărut o ofertă nouă;
4. interfața tuturor utilizatorilor este actualizată automat, fără ca aceștia să fie nevoiți să reîncarce pagina.

Această abordare îmbunătățește semnificativ experiența utilizatorului, creând senzația de interacțiune live, esențială pentru o platformă de licitații.

3.20 Design pattern-uri și principii SOLID utilizate

Dezvoltarea aplicației de licitații web s-a realizat ținând cont de principii moderne de arhitectură software, precum și de utilizarea unor pattern-uri consacrate care asigură claritate, testabilitate și extensibilitate.

Principiile SOLID

Aplicația respectă principiile SOLID, care ghidează dezvoltarea orientată pe obiect:

- S – Single Responsibility Principle** Fiecare clasă are o singură responsabilitate. De exemplu, serviciile (`AuctionService`, `StripeService`, `AuthService`) sunt clar separate de logica de acces la date (repositories) sau de controlerele web.
- O – Open/Closed Principle** Codul este deschis pentru extindere, dar închis pentru modificare. Noile funcționalități pot fi adăugate prin implementări suplimentare ale interfețelor existente, fără a modifica logica existentă.
- L – Liskov Substitution Principle** Interfețele definite (ex: `IAuctionService`, `IStripeService`) permit injectarea implementărilor concrete fără a afecta comportamentul aplicației.
- I – Interface Segregation Principle** Interfețele sunt specifice și bine definite. De exemplu, serviciile nu definesc metode inutile — fiecare interfață reflectă exact funcționalitățile oferite.
- D – Dependency Inversion Principle** Codul de nivel superior (ex: controlerele) depinde de abstracții (`IAuctionService`, `IAuthService`), nu de implementări concrete. Toate dependențele sunt injectate prin containerul de Dependency Injection nativ din ASP.NET Core.

Design pattern-uri utilizate

■ Repository Pattern

Fiecare entitate (ex: `Auction`, `Bid`, `User`) are un repository asociat, responsabil pentru accesul la date. Acest pattern izolează logica de persistență și oferă un nivel suplimentar de abstractizare.

■ Service Layer Pattern

Logica de business este organizată în servicii care coordonează validările, conversiile și apelurile la repositories. Acest strat este crucial pentru separarea clară între controller și date.

■ DTO Pattern (Data Transfer Object)

Aplicația utilizează DTO-uri pentru a transfera date între straturi. Acestea sunt distincte de modelele de business și modelele din baza de date și oferă un control mai bun asupra contractelor publice.

■ Mapping cu AutoMapper

Pentru a reduce codul boilerplate de conversie între entități, DTO-uri și modele de UI, a fost utilizată biblioteca AutoMapper. Mapările sunt declarative și întreținerea codului devine mai simplă.

■ Result Pattern pentru gestionarea erorilor

În locul excepțiilor standard, metodele de business returnează obiecte de tip `Result` sau `Result<T>`, care conțin câmpuri pentru succes/eșec și lista de erori, dacă este cazul. Acest pattern permite un control mai clar asupra fluxului de execuție și evită excepțiile necontrolate.

■ Unit of Work (implicit prin EF Core)

Salvarea entităților se face într-un singur apel `SaveChangesAsync`, ceea ce garantează tranzaționalitatea modificărilor și încadrarea într-un context de lucru coerent.

■ Dependency Injection Pattern

Aplicația utilizează sistemul nativ de DI al framework-ului ASP.NET Core. Toate serviciile, repositoryile și componentele sunt înregistrate în container și injectate automat prin constructori.

Capitolul 4

Arhitectura aplicației

4.1 Arhitectura generală a aplicației

Introducere

Aplicația de licitații web a fost proiectată conform principiilor de arhitectură modernă multi-strat, cu separarea clară a responsabilităților între componentele de prezentare, logică de business și acces la date. Soluția este împărțită în două proiecte principale:

- aplicația web (frontend) – realizată cu Blazor;
- API-ul (backend) – construit cu ASP.NET Core și C#.

Această abordare permite dezvoltarea independentă a fiecărei componente, testarea modulară și scalarea eficientă a aplicației.

Structura principală

Arhitectura generală este alcătuită din următoarele componente:

Blazor Web App Reprezintă interfața cu utilizatorul. Este implementată cu Blazor și oferă pagini și componente UI pentru: autentificare, vizualizarea licitațiilor, plasarea de oferte, urmărirea comenzilor și interacțiunea cu platforma. Comunică cu backend-ul prin HttpClient și semnale WebSocket transmise prin SignalR.

ASP.NET Core Web API Expune un set de endpointuri RESTful prin care sunt gestionate datele aplicației. API-ul oferă funcționalități precum înregistrare, autentificare, administrarea licitațiilor, procesarea plăților și gestionarea livrărilor. Autorizarea se face pe bază de roluri.

Baza de date Postgres SQL Conține datele persistente ale aplicației: utilizatori, licitații, oferte, tranzacții, livrări. Interacțiunea cu baza de date este realizată prin Entity Framework Core folosind modelul Code First și migrații automate.

Stripe Payment Service Integrează funcționalități de plată online. După finalizarea unei licitații, plata este procesată automat printr-un job programat care utilizează Stripe API. Clienții trebuie să aibă cont Stripe atașat.

SignalR Permite actualizarea în timp real a interfeței pentru toți utilizatorii care vizualizează simultan aceeași licitație. Atunci când o ofertă este plasată, ea apare instantaneu pe ecranele tuturor participanților, fără reîncărcarea paginii.

Hangfire + Background Joburi Un sistem de joburi programate folosit pentru procesarea asincronă a logicii de finalizare a licitațiilor. La expirarea unei licitații, un job verifică câștigătorul și inițiază procesul de plată. În această etapă, banii sunt transferați din contul cumpărătorului în contul platformei de licitații, prin intermediul Stripe. Acest mecanism garantează că doar cumpărătorii reali, care dispun de fondurile necesare, pot câștiga o licitație.

Ulterior, în momentul în care produsul este marcat ca „livrat cu succes” de către șofer (driver), se declanșează un al doilea transfer: din contul platformei în contul vânzătorului.

Sistem de protecție a tranzacțiilor Mecanismul de plată al aplicației a fost conceput astfel încât să protejeze atât cumpărătorul, cât și vânzătorul:

- După ce o licitație este câștigată, banii nu sunt transferați direct către vânzător, ci sunt reținuți temporar în contul platformei de licitații.
- Acest lucru oferă o garanție că vânzătorul nu este plătit până când produsul este efectiv livrat și confirmat.
- Pe de altă parte, cumpărătorul nu este expus riscului de a plăti pentru un produs care nu va fi expediat, deoarece fondurile rămân gestionate de platformă până la confirmarea livrării.

Această arhitectură de tip escrow automatizat asigură încredere și transparență în procesul de licitație și livrare, fără intervenție manuală.

Autentificare și autorizare Sistemul se bazează pe tokenuri JWT și cookie-uri securizate. După autentificare, utilizatorii rămân logați pentru o perioadă de timp. Accesul la funcționalități este controlat prin roluri: SuperAdmin, Admin, User, Driver.

Deploy în cloud Aplicația este hostată în Azure: atât API-ul cât și interfața web sunt publicate în Web Apps, iar baza de date PostgreSQL este găzduită în Railway. Procesul de deploy este complet automatizat prin pipeline-uri definite în YAML, în Azure DevOps.

4.2 Modelul de date și schema bazei de date

Modelul de date al platformei de licitații web este proiectat relațional, respectând principiile normalizării și ale coerenței semantice. Arhitectura bazei de date susține funcționalitățile specifice aplicației: înregistrarea utilizatorilor, gestionarea licitațiilor, ofertarea, integrarea plăților și livrarea produselor. Implementarea a fost realizată utilizând Postgres SQL și Entity Framework Core, urmând abordarea Code First cu migrații automate.

Structura bazei de date

Baza de date este compusă din următoarele entități principale:

1. Users

Reprezintă utilizatorii obișnuiți ai platformei (vânzători și cumpărători).

Câmpuri:

- ▣ Id – cheie primară;
- ▣ Role – identifică tipul de utilizator (ex. User);
- ▣ FirstName, LastName, Address, Email, PasswordHash;
- ▣ StripeConnectedAccountId, StripeCustomerId – identificatori Stripe pentru vânzare/cumpărare.

Relații:

- ▣ 1:N cu Auctions (ca vânzători);
- ▣ 1:N cu Bids (ca ofertanți);
- ▣ 1:1 cu PasswordResetTokens.

2. Auctions

Reprezintă licitațiile create în platformă.

Câmpuri:

- ▣ Id, Title, Description, StartingPrice, CurrentPrice;
- ▣ MinBidIncrement, StartTime, EndTime, Status;
- ▣ SellerId – FK către Users;
- ▣ DriverId – FK către Drivers;
- ▣ CategoryId – FK către Categories.

Relații:

- ▣ 1:N cu Bids;
- ▣ 1:N cu AuctionImages;
- ▣ M:1 cu Users, Categories, Drivers.

3. Bids

Reprezintă ofertele utilizatorilor asupra licitațiilor.

Câmpuri:

- ▣ Id, AuctionId, BidderId, Amount, Date.

Relații:

- M:1 cu Users;
- M:1 cu Auctions.

4. Categories

Grupări tematice ale licitațiilor.

Câmpuri:

- Id, Name, ImageUrl.

Relație:

- 1:N cu Auctions.

5. AuctionImages

Conține imaginile atașate unei licitații.

Câmpuri:

- Id, AuctionId, ImageUrl.

Relație:

- M:1 cu Auctions.

6. Drivers

Utilizatori cu rol de șoferi responsabili de livrarea produselor.

Câmpuri:

- Id, Role, Email, PasswordHash.

Relație:

- 1:N cu Auctions (prin DriverId).

7. Admins

Reprezintă administratorii platformei.

Câmpuri:

- Id, Role, Email, PasswordHash.

8. PasswordResetTokens

Tabel pentru resetarea parolelor utilizatorilor.

Câmpuri:

- ▣ Id, Token, ExpiresAt, UserId.

Relație:

- ▣ M:1 cu Users.

Relațiile între entități

Structura bazei de date este guvernată de relații de tipul:

- ▣ 1:N între Users și Auctions (vânzător);
- ▣ 1:N între Users și Bids (cumpărător);
- ▣ 1:N între Auctions și Bids;
- ▣ 1:N între Auctions și AuctionImages;
- ▣ 1:N între Categories și Auctions;
- ▣ 1:N între Drivers și Auctions;
- ▣ 1:1 între Users și PasswordResetTokens.

Diagramă relațională (ERD)

În figura de mai jos este reprezentată diagrama relațională utilizată în implementarea aplicației:

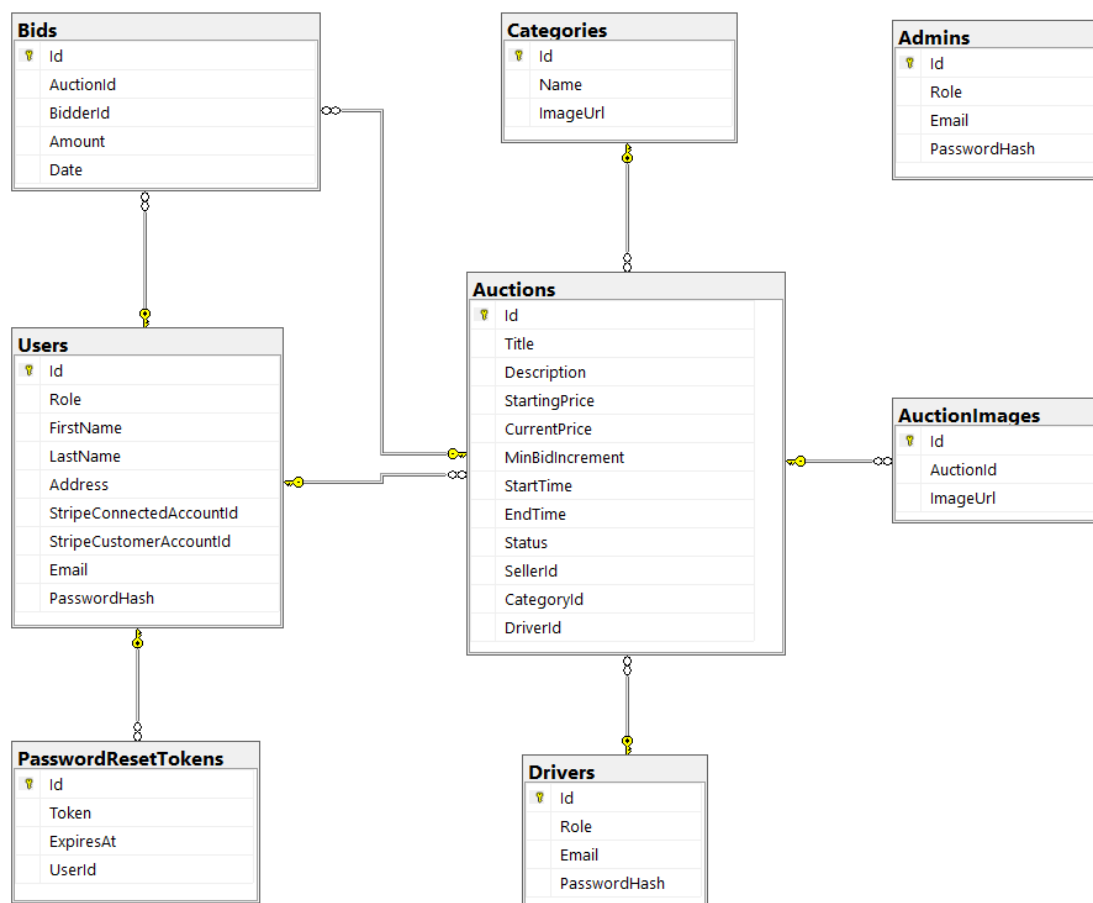


Figura 4.1: Schema relațională a bazei de date

Implementare tehnică

Modelarea bazei de date a fost realizată utilizând:

- ▣ Abordare **Code First** cu Entity Framework Core;
- ▣ Mapări definite prin atribute și Fluent API;
- ▣ Enumuri pentru câmpuri precum Role, Status, DeliveryStatus;
- ▣ Navigații virtuale pentru acces rapid la date relaționate;
- ▣ Migrații automate pentru menținerea sincronizării între cod și baza de date.

4.3 Arhitectura aplicației API pe straturi

Pentru a asigura claritate, modularitate și testabilitate în dezvoltarea backend-ului aplicației de licitații, soluția API a fost proiectată folosind o arhitectură multi-strat. Acest model de organizare a codului permite separarea responsabilităților și respectarea principiilor de bună practică din ingineria software, cum ar fi principiile SOLID, Dependency Inversion Principle, și utilizarea de interfețe, servicii, mapări și design pattern-uri consacrate.

Structura pe straturi

Aplicația backend este împărțită logic în trei straturi majore, fiecare având roluri bine definite:

1. **Stratul Infrastructure** Stratul Infrastructure gestionează interacțiunile cu resursele externe ale aplicației, cum ar fi baza de date și serviciile terțe.

Conținutul acestui strat include:

- Repository-uri: implementări ale interfețelor pentru accesul la date;
- Entitățile persistente (corespunzătoare tabelor din Postgres SQL);
- Contextul bazei de date (DbContext) – definește seturile de entități și relațiile dintre acestea;
- Serviciul Stripe – gestionează crearea de conturi pentru vânzători și cumpărători, efectuarea tranzacțiilor inițiale și finale, actualizarea datelor de plată și metode de plată;
- Jobul Hangfire – programat să fie executat automat la finalul unei licitații, responsabil cu inițierea tranzacției Stripe de la cumpărător către platformă;
- Mapări între entitățile de bază de date și modelele de business din stratul Core (și invers).

2. **Stratul Core (Application Layer)** Acest strat cuprinde logica de business a aplicației și este nucleul decizional al întregului sistem.

Componente:

- Interfețele serviciilor și ale repository-urilor – definesc contractele funcționale;
- Implementările serviciilor de business – în care este aplicată logica specifică platformei de licitații;
- Modelele de business – structuri care reflectă datele manipulate de servicii;
- DTO-uri de business – utilizate pentru transferul de date între straturi;
- Mapări între DTO-uri și modele de business;
- Validări și reguli de integritate aplicate asupra logicii de ofertare, licitații, livrări, etc.

3. **Stratul Presentation (API)** Acest strat este responsabil cu expunerea funcționalităților aplicației prin intermediul unor endpointuri HTTP (RESTful API).

Structură:

- Controlere ASP.NET Core – pentru fiecare funcționalitate majoră (ex: AuctionController, BidController);
- DTO-uri de prezentare – utilizate ca parametri sau răspunsuri în endpointuri;
- Mapări între DTO-urile de prezentare și cele din stratul de business;
- Injecția serviciilor definite în Core pentru delegarea logicii.

Principii și pattern-uri aplicate

- **Lucrul cu interfețe:** toate serviciile și repository-urile sunt definite prin interfețe și implementate separat, ceea ce facilitează testarea și extensibilitatea.
- **Injectia de dependență:** toate interfețele sunt injectate prin containerul built-in din ASP.NET Core (`IServiceCollection`), permițând controlul centralizat al instanțierii și ciclului de viață.
- **Repository pattern:** folosit pentru abstractizarea accesului la baza de date.
- **Automapper:** utilizat pentru maparea automată între entități, modele de business și DTO-uri, reducând codul boilerplate și riscul de erori manuale.
- **Patternul Result:** în locul aruncării de excepții în logica de business, a fost implementat un obiect `Result<T>` care conține eventualele erori și datele returnate. Această abordare favorizează controlul centralizat al erorilor și evită excepțiile runtime costisitoare.

Listing 4.1: Result class

```
public class Result
{
    public List<string> Errors { get; set; } = new List<string>();

    public bool HasErrors => Errors.Any();

    public override string ToString()
    {
        var errorMessage = string.Join(Environment.NewLine, Errors);
        return HasErrors ? errorMessage : "Result is successful with no errors.";
    }
}

public class Result<T> : Result
{
    public T? Data { get; set; }
}
```

Testabilitate și modularitate

Arhitectura stratificată a permis implementarea unei strategii solide de testare, împărțită astfel:

- **Teste de integrare pentru *Infrastructure*** – un total de **52 teste**, validate pe o bază de date de test, acoperind interacțiunile cu contextul EF Core și persistarea datelor;
- **Teste unitare pentru *Core*** – **56 de teste** au fost dezvoltate pentru serviciile de business logic, testate independent prin folosirea de repository-uri simulate (*mocked*);
- **Teste unitare pentru *Presentation*** – **128 teste** axate pe comportamentul controlerelor, validarea adnotărilor de model și a atributelor personalizate.

Comunicarea cu frontend-ul

Notificările în timp real între backend și interfața Blazor sunt realizate cu ajutorul SignalR, integrat în infrastructură. Atunci când o ofertă este plasată pe o licitație activă, backend-ul notifică imediat toți utilizatorii conectați la acea pagină, fără a fi necesar un refresh manual al interfeței.

4.4 Autentificare și autorizare

Securitatea accesului la funcționalitățile aplicației de licitații este asigurată printr-un sistem de autentificare bazat pe JSON Web Tokens (JWT) și autorizare pe bază de roluri. Mecanismul implementat respectă principiile de autentificare stateless, securizare prin cookie-uri HttpOnly, și control granular al accesului la resurse prin validări atât la nivel de API, cât și la nivel logic.

Mecanismul de autentificare

Autentificarea utilizatorilor se realizează printr-un formular de login standard, în care utilizatorii introduc adresa de email și parola. Această metodă este comună tuturor tipurilor de utilizatori (User, Admin, Driver, SuperAdmin).

După autentificarea cu succes, serverul generează un token JWT semnat cu o cheie secretă și cu algoritmul HmacSha256. Tokenul conține următoarele claims:

- **sub** – ID-ul utilizatorului;
- **email** – adresa de email;
- **role** – rolul utilizatorului (ca string, ex: "Admin", "User").

Tokenul are o durată de viață fixă și este returnat către client sub formă de cookie securizat, setat cu următoarele politici:

Listing 4.2: Configurare cookie-uri HttpOnly & Secure

```
opts.MinimumSameSitePolicy = SameSiteMode.Strict;  
opts.HttpOnly = HttpOnlyPolicy.Always;  
opts.Secure = CookieSecurePolicy.Always;
```

Aceste setări împiedică accesul JavaScript-ului la cookie, protejează împotriva atacurilor XSS și asigură transmiterea doar prin conexiuni HTTPS.

Autorizarea pe bază de roluri

Aplicația definește patru roluri distincte:

- **User** – utilizator obișnuit (cumpărător/vânzător);
- **Admin** – utilizator cu drepturi de gestionare a licitațiilor și invitații pentru noi conturi;
- **Driver** – utilizator responsabil de livrare;
- **SuperAdmin** – are drepturi complete asupra aplicației.

Accesul la endpointurile din API este reglementat prin atributele `Authorize` din ASP.NET Core, de forma:

Listing 4.3: Exemplu de atribut `Authorize`

```
[Authorize(Roles = "Admin")]
```

Controlul accesului la resurse

Pe lângă restricțiile bazate pe roluri, aplicația impune și verificări contextuale, pentru a asigura că utilizatorii pot accesa doar propriile resurse. De exemplu:

- un **User** poate edita o licitație doar dacă `SellerId` coincide cu `UserId` extras din token;
- în serviciile de business, aceste verificări sunt făcute explicit, pentru a preveni manipulări neautorizate.

Această abordare consolidează securitatea logică a aplicației și elimină riscul accesării datelor altor utilizatori.

Înregistrare, logout și identificare utilizator

- Înregistrarea utilizatorilor simpli se face printr-un endpoint public decorat cu `[AllowAnonymous]`, cu validări de email și parolă.
- Autentificarea este urmată de salvarea tokenului JWT într-un cookie securizat.
- Endpointul Logout elimină cookie-ul și invalidează sesiunea.
- Endpointul `AuthenticateMe` permite clientului (frontend) să afle identitatea și rolul utilizatorului logat, fără a face parsing local al tokenului.

Crearea conturilor speciale de către administratori

Pentru rolurile privilegiate precum `Admin` și `Driver`, aplicația utilizează un sistem de creare controlată, gestionat exclusiv de utilizatorii existenți cu permisiuni suficiente:

- Doar un `Admin` sau `SuperAdmin` poate iniția procesul de creare a unui nou cont special.
- Adminul specifică adresa de email, parola și rolul utilizatorului (`Admin` sau `Driver`).
- După confirmarea datelor, contul este creat direct în platformă și este asociat cu rolul desemnat.
- Utilizatorul creat va putea ulterior să se autentifice folosind emailul și parola furnizate.

Acest flux oferă un control strict și direct asupra distribuirii rolurilor privilegiate în aplicație, prevenind înregistrările neautorizate și reducând riscul de compromitere a securității administrative.

4.5 Arhitectură frontend

Partea de frontend a aplicației de licitații a fost realizată utilizând tehnologia Blazor, într-o manieră componentizată, modulară și extensibilă. Arhitectura vizuală este organizată în jurul unor componente Razor reutilizabile, fiecare responsabilă de o funcționalitate distinctă, cu interfață proprie și logică de control separată.

Comunicarea dintre interfața utilizator și server se realizează prin două canale:

- **HttpClient**, pentru operațiuni asincrone tradiționale (GET, POST, PUT, DELETE);
- **SignalR**, pentru recepționarea notificărilor în timp real (ex: actualizarea listelor de oferte fără reîncărcarea paginii).

Pentru fiecare controller expus în API-ul backend, aplicația frontend definește un client HTTP dedicat. Acestea sunt implementări concrete care corespund unu-la-unu cu funcționalitățile controllerelor. Fiecare astfel de client implementează o interfață specifică, respectând

principiile programării orientate pe contract. Aceste interfețe sunt apoi injectate în servicii Blazor, care acționează ca un strat intermediar între logica de UI și funcționalitățile de infrastructură.

Serviciile Blazor sunt responsabile de:

- maparea datelor între ViewModel-urile folosite în UI și modelele de transport (DTO-uri) ale clienților HTTP;
- validarea și pregătirea datelor înainte de apelul endpointurilor;
- gestionarea răspunsurilor și convertirea acestora în formate ușor de utilizat în interfață.

Modelele utilizate în comunicarea cu serverul sunt construite în oglindă față de cele din stratul de Presentation al API-ului. Această aliniere structurală facilitează dezvoltarea sincronizată a celor două componente și reduce riscul incompatibilităților.

Interfețele serviciilor frontend sunt injectate în fișierele de tip code-behind ale componentelor Razor. Astfel, logica de prezentare este separată de logica funcțională, în concordanță cu principiul *Separation of Concerns*. Fiecare componentă Razor își gestionează starea locală, declanșează acțiuni prin intermediul serviciilor injectate și actualizează vizual interfața în funcție de rezultatele obținute.

Pentru a promova reutilizarea codului și a evita redundanța, majoritatea elementelor vizuale sunt implementate sub forma unor componente Razor reutilizabile. Acestea includ formulare de autentificare, carduri de licitație, tabele de livrări, notificări și bannere informative. Reutilizarea acestora în diverse pagini contribuie la consistența designului aplicației și facilitează mentenanța pe termen lung.

4.6 Arhitectura și integrarea serviciilor Stripe

Integrarea cu Stripe oferă aplicației de licitații un sistem modern, sigur și automatizat de gestionare a tranzacțiilor financiare între cumpărători, platformă și vânzători. Stripe este utilizat pentru crearea conturilor de vânzător și cumpărător, salvarea metodelor de plată, procesarea plăților și efectuarea transferurilor de fonduri, conform unui mecanism de tip escrow.

Componentele principale

Serviciul `StripeService`, localizat în stratul Infrastructure, implementează interfața `IStripeService` și este responsabil de interacțiunea directă cu Stripe SDK. Metodele principale sunt:

- `CreateConnectedAccountAsync` – creează conturi Stripe pentru vânzători;
- `CreateCustomerAccountAsync` – creează conturi pentru cumpărători;
- `TransferFromCustomerToPlatform` – inițiază un `PaymentIntent` pentru plata către platformă;
- `TransferFromPlatformToSeller` – transferă fondurile către vânzător, după livrare;
- `UpdateCustomerPaymentMethodAsync` – actualizează metoda implicită de plată;

- `CreateAccountLinkAsync` – generează linkuri pentru onboarding;
- Metode auxiliare pentru consultarea conturilor și metodelor de plată.

Toate metodele folosesc o clasă `Result<T>`, evitând aruncarea excepțiilor și facilitând gestionarea controlată a erorilor.

Automatizarea cu Hangfire

Pentru automatizarea procesului de încheiere a unei licitații și inițiere a plății, aplicația utilizează biblioteca Hangfire. La crearea unei licitații, este programat un job care va rula la momentul expirării:

Listing 4.4: Programarea jobului Hangfire în metoda `AddAsync` (metodă în care se creează o licitație nouă)

```
public async Task<PreviewAuction> AddAsync(PreviewAuction auction, bool
    callScheduler = true)
{
    var auctionDb = _mapper.Map<AuctionDb>(auction);
    await _context.Auctions.AddAsync(auctionDb);
    await _context.SaveChangesAsync();

    if (callScheduler)
    {
        var delay = auctionDb.EndTime.ToUniversalTime() - DateTime.UtcNow;
        BackgroundJob.Schedule<AuctionJobService>(svc => svc.
            ProcessAuctionEnd(auctionDb.Id), delay);
    }

    var createdAuction = _mapper.Map<PreviewAuction>(auctionDb);
    return createdAuction;
}
```

Jobul programat apelează clasa `AuctionJobService`, unde metoda `ProcessAuctionEnd` execută pașii de procesare:

Listing 4.5: Clasa `AuctionJobService` și metoda `ProcessAuctionEnd`

```
public class AuctionJobService(AuctionDbContext _context, IStripeService
    _stripeService)
{
    [AutomaticRetry(Attempts = 5, DelaysInSeconds = new[] { 60 })]
    public async Task ProcessAuctionEnd(int auctionId)
    {
        var auction = await _context.Auctions.FirstOrDefaultAsync(a =>
            a.Id == auctionId);
        if (auction == null || auction.Status != AuctionStatus.
            InProgress)
            return;

        var winningBid = await _context.Bids
            .Where(b => b.AuctionId == auctionId)
            .OrderByDescending(b => b.Amount)
```

```
.Include(b => b.Bidder)
.FirstOrDefaultAsync();
if (winningBid == null || string.IsNullOrEmpty(winningBid.
    Bidder.StripeCustomerId))
{
    auction.Status = AuctionStatus.Canceled;
    await _context.SaveChangesAsync();
    return;
}

var transferDto = new CustomerToPlatformTransferDto
{
    CustomerAccountId = winningBid.Bidder.
        StripeCustomerId,
    Amount = (long)(winningBid.Amount * 100)
};

var transferResult = await _stripeService.
    TransferFromCustomerToPlatform(transferDto);
if (transferResult.HasErrors)
{
    auction.Status = AuctionStatus.Canceled;
    await _context.SaveChangesAsync();
    return;
}

auction.Status = AuctionStatus.InTransit;
await _context.SaveChangesAsync();
}
}
```

Înregistrarea Hangfire în aplicație

În fișierul Program.cs, Hangfire este înregistrat astfel:

Listing 4.6: Configurare Hangfire în Program.cs

```
builder.Services.AddHangfire(cfg => cfg
    .UseSimpleAssemblyNameTypeSerializer()
    .UseRecommendedSerializerSettings()
    .UsePostgreSqlStorage(options =>
    {
        options.UseNpgsqlConnection(connectionString);
    }));

builder.Services.AddHangfireServer();
```

Această configurație pornește automat un server Hangfire care rulează joburile programate, folosind ca mediu de stocare o bază de date Postgres SQL.

Mecanismul escrow

Stripe este integrat pentru a proteja atât cumpărătorii, cât și vânzătorii, în conformitate cu următoarele reguli:

- După câștigarea unei licitații, banii sunt transferați din contul cumpărătorului în contul platformei.
- Ulterior, în momentul în care șoferul confirmă livrarea, se face al doilea transfer: din contul platformei în contul vânzătorului.

Această abordare asigură:

- că vânzătorul nu este plătit în avans fără a livra produsul;
- că cumpărătorul plătește doar pentru un produs ce va fi expediat;
- că aplicația funcționează ca intermediar de încredere.

4.7 Integrarea SignalR pentru actualizări în timp real

Pentru a oferi o experiență interactivă și fluentă utilizatorilor care participă la o licitație, aplicația implementează o funcționalitate de actualizare în timp real a ofertelor folosind SignalR – o bibliotecă dezvoltată de Microsoft pentru comunicare bidirecțională între client și server.

Rolul SignalR în contextul aplicației

În cadrul platformei, atunci când mai mulți utilizatori sunt conectați la aceeași licitație, este esențial ca fiecare nouă ofertă plasată să fie reflectată instantaneu pe interfețele tuturor participanților. SignalR permite transmiterea acestor notificări fără necesitatea reîncărcării paginii, eliminând astfel nevoia de polling sau refresh manual.

Configurarea Hub-ului

În backend, a fost creată o clasă `AuctionHub`, care moștenește `Hub` din SignalR. Aceasta definește două metode principale pentru gestionarea grupurilor:

Listing 4.7: Clasa `AuctionHub`

```
public class AuctionHub : Hub
{
    public Task JoinAuctionGroup(int auctionId)
    => Groups.AddToGroupAsync(Context.ConnectionId, $"auction-{auctionId}");

    public Task LeaveAuctionGroup(int auctionId)
    => Groups.RemoveFromGroupAsync(Context.ConnectionId, $"auction-{auctionId}");
}
```

Astfel, fiecare utilizator este asociat cu un grup corespunzător ID-ului licitației pe care o vizualizează, permițând trimiterea direcționată a mesajelor către acel grup.

Trimiterea notificărilor din backend

La crearea unei noi oferte (bid), endpointul `CreateBidAsync` utilizează instanța de `IHubContext< AuctionHub >` pentru a trimite notificarea către grupul de utilizatori conectat la licitația respectivă:

Listing 4.8: Trimiterea notificării către grupul SignalR

```
await _hub.Clients.Group($"auction-{createBidRequest.AuctionId}")
    .SendAsync("ReceiveBid", bidResponse);
```

Această metodă transmite către toți utilizatorii din grupul `auction-{id}` obiectul `BidResponse` corespunzător ofertei nou plasate.

Inițializarea conexiunii SignalR în frontend

În aplicația Blazor, conexiunea la hub este definită și înregistrată în containerul de servicii:

Listing 4.9: Configurarea HubConnection în DI

```
builder.Services.AddTransient(sp =>
{
    var js = sp.GetRequiredService<IJSRuntime>();
    var hubUrl = new Uri(new Uri(apiBase), "hubs/auction");

    return new HubConnectionBuilder()
        .WithUrl(hubUrl, options =>
            options.AccessTokenProvider = () =>
                js.InvokeAsync<string>("blazorExtensions.GetCookie", "access_token").
                    AsTask()
        )
        .WithAutomaticReconnect()
        .Build();
});
```

Aceasta permite utilizarea serviciului SignalR în paginile Blazor care au nevoie de comunicare în timp real.

Logica din pagina de licitație (**BidAuction.razor.cs**)

În codul `BidAuction.razor.cs`, conexiunea este activată în `OnInitializedAsync`, iar clientul se abonează la mesajele primite prin metoda `"ReceiveBid"`:

Listing 4.10: Abonarea la mesajele ReceiveBid

```
Hub.On<BidResponse>("ReceiveBid", bidResp =>
{
    if (bidResp.AuctionId != AuctionId)
        return;

    var createdBid = Mapper.Map<BidViewModel>(bidResp);
    UpdateAuctionAfterSuccessfulBid(createdBid);
    StateHasChanged();
});
```


Tot aici, utilizatorul se alătură grupului corespunzător licitației:

Listing 4.11: Alăturarea la grupul licitației

```
await Hub.StartAsync();  
await Hub.SendAsync("JoinAuctionGroup", AuctionId);
```

La părăsirea paginii, utilizatorul este eliminat din grup prin metoda `LeaveAuctionGroup`.

Beneficii arhitecturale

Integrarea SignalR oferă următoarele avantaje:

- elimină necesitatea interogării frecvente a serverului (polling);
- îmbunătățește performanța aplicației;
- asigură sincronizare în timp real a ofertelor;
- oferă o experiență de utilizare modernă, specifică aplicațiilor web interactive.

4.8 Deploy în cloud și livrare continuă (CI/CD)

Aplicația este distribuită în cloud, utilizând infrastructura Azure pentru găzduirea componentelor web și Railway pentru baza de date PostgreSQL. Structura de distribuție este împărțită astfel:

- API-ul (ASP.NET Core) este găzduit în Azure Web App cu numele `auction-api-test`;
- Frontend-ul (Blazor WebAssembly) este găzduit într-un alt Azure Web App numit `auction-web-test`;
- Baza de date PostgreSQL este hostată separat în Railway.

CI/CD cu Azure DevOps

Livrarea continuă este automatizată prin două pipeline-uri definite în fișiere YAML, configurate în Azure DevOps: unul pentru API și unul pentru interfața web.

Pipeline API:

1. Instalarea SDK-ului .NET 9;
2. Restaurarea pachetelor;
3. Build în configurația `ReLease`;
4. Rularea testelor unitare din proiectele `Auction.Api.Tests` și `Auction.Core.Tests`;
5. Publicarea aplicației în format ZIP;
6. Deploy automat în Azure Web App folosind metoda `zipDeploy`.

Secretele (ex: connection string-uri, chei Stripe) sunt păstrate în `Environment Variables` din configurarea aplicației din Azure.

În figura de mai jos este ilustrat pipeline-ul de livrare continuă configurat pentru proiectul API. Acesta evidențiază pașii parcurși de la build până la deploy-ul automat în Azure Web App.

The screenshot displays the Azure DevOps pipeline interface. On the left, a table lists the steps of the pipeline for job #20250615.9. All steps are marked as successful with green checkmarks. On the right, a detailed view of the selected 'Job' step shows its execution details, including the agent used, start time, duration, and the fact that 100% of tests passed.

Jobs in run #20250615.9		
Gabi-Plosnita.Auction-API		
Jobs		
✓	Job	1m 16s
✓	Initialize job	<1s
✓	Checkout Gabi-Plosnita...	2s
✓	Install .NET SDK	<1s
✓	Restore NuGet packages	4s
✓	Build solution	15s
✓	Run unit tests	20s
✓	Publish unit test results	<1s
✓	Publish API	12s
✓	Publish build artifact	3s
✓	Deploy to Azure Web ...	14s
✓	Post-job: Checkout Ga...	<1s
✓	Finalize Job	<1s

Job Details:

- 1 Pool: `my-agent`
- 2 Agent: `DESKTOP-03D6454`
- 3 Started: Today at 5:27 AM
- 4 Duration: 1m 16s
- 5
- 6 ▶ Job preparation parameters
- 31 📦 1 artifact produced
- 32 🏆 100% tests passed

Figura 4.2: Pipeline Api

Pipeline Frontend:

1. Instalarea SDK-ului .NET 9;
2. Restaurarea pachetelor;
3. Înlocuirea variabilelor din fișierul `appsettings.json` cu ajutorul task-ului `FileTransform`;
4. Build și publicare în format ZIP;
5. Rularea testelor unitare (`AuctionWebAppTests`);

6. Deploy automat în Azure Web App (auction-web-test).

Secretele frontend-ului sunt păstrate în Azure DevOps Pipeline Variables și marcate ca secret pentru criptare.

Figura următoare prezintă pipeline-ul pentru aplicația frontend Blazor, inclusiv etapele de build, testare, înlocuire a setărilor de configurare și publicare automată în Azure.

The screenshot displays the Azure DevOps pipeline interface. On the left, a sidebar titled 'Jobs in run #20250614.7' for 'Gabi-Plosnita.AuctionWebApp' lists the pipeline steps. The 'Job' step is selected, showing a duration of 1m 48s. The steps listed are: Initialize job (<1s), Checkout Gabi-Plosnita... (2s), Install .NET SDK (<1s), Restore packages (3s), Substitute JSON varia... (<1s), Build project (21s), Run WebApp unit tests (10s), Publish project (44s), Publish artifact (5s), Deploy to Azure Web ... (19s), Post-job: Checkout Ga... (<1s), and Finalize Job (<1s). All steps are marked with a green checkmark, indicating success. On the right, the 'Job' details panel shows the following information: Pool: my-agent, Agent: DESKTOP-03D6454, Started: Today at 2:24 AM, Duration: 1m 48s. It also shows 'Job preparation parameters' and a summary of results: '1 artifact produced' and '100% tests passed'.

Step	Duration
Initialize job	<1s
Checkout Gabi-Plosnita...	2s
Install .NET SDK	<1s
Restore packages	3s
Substitute JSON varia...	<1s
Build project	21s
Run WebApp unit tests	10s
Publish project	44s
Publish artifact	5s
Deploy to Azure Web ...	19s
Post-job: Checkout Ga...	<1s
Finalize Job	<1s

Figura 4.3: Pipeline Web App

Avantaje

Această arhitectură de livrare oferă:

- Automatizare completă a deploy-ului;
- Separarea proceselor pentru API și frontend;
- Rulare automată a testelor înainte de livrare;
- Scalabilitate individuală pentru fiecare componentă;
- Control și securitate crescută prin separarea secreteleor.

Capitolul 5

Contribuții personale

Platforma de licitații dezvoltată integrează funcționalități avansate care depășesc structura clasică a unui sistem de comerț digital. Pe lângă componenta de bază de plasare a ofertelor și adjudecare a licitațiilor, aceasta include extensii originale precum gestionarea livrărilor, automatizarea tranzacțiilor și notificări în timp real, toate implementate cu tehnologii moderne și abordări arhitecturale robuste. În continuare sunt prezentate principalele contribuții distinctive ale platformei, care reflectă atât profunzimea funcțională, cât și preocuparea pentru experiența utilizatorului, securitate și scalabilitate.

5.1 Modulul original de livrări și dashboard-ul dedicat șoferilor

Una dintre cele mai importante contribuții originale constă în extinderea arhitecturii aplicației pentru a include un modul complet de livrări, gestionat printr-o colaborare între administratori și șoferi. Spre deosebire de aplicațiile clasice de licitații, în această platformă:

- Administratorii au acces la o secțiune specială unde pot vizualiza toate licitațiile care trebuie livrate;
- Interfața oferă opțiuni de filtrare avansată: Any Driver, No Driver Assigned, sau No Filter;
- Pentru fiecare licitație, administratorul poate vizualiza datele detaliate ale vânzătorului, ale câștigătorului și (dacă există) ale șoferului;
- În baza acestor informații, administratorul poate asigna manual un șofer dintr-un dropdown cu toți șoferii înregistrați în platformă.

Această funcționalitate permite o gestionare centralizată și eficientă a procesului logistic, fiind o extensie naturală, dar rar întâlnită, în aplicațiile de acest tip.

Pe partea operațională, șoferii autentificați dispun de un dashboard personalizat, în care pot vizualiza toate licitațiile care le-au fost alocate. Acest spațiu dedicat le permite să consulte detaliile legate de produs, cumpărător și vânzător, asigurând astfel o livrare eficientă.

5.2 Sistem de tranzacții automate securizate prin Stripe

Un alt aport semnificativ îl reprezintă arhitectura plăților online, construită în jurul platformei Stripe și extinsă printr-un mecanism complet de procesare automată:

- Utilizatorii trebuie să furnizeze informații extinse înainte de a publica licitații: email, număr de telefon, adresă, IBAN, website;
- Cei care doresc să plaseze oferte trebuie să salveze o metodă de plată, care este securizată și stocată de Stripe;
- La încheierea unei licitații, dacă există un câștigător, este programat un job care declanșează transferul automat al fondurilor de la cumpărător către platforma de licitații;
- Ulterior, când șoferul confirmă livrarea, se execută un al doilea transfer automat de la platformă către vânzător.

Această logică a fost implementată cu grijă pentru a reproduce un sistem de tip escrow digital, în care fondurile sunt securizate de o entitate intermediară (platforma de licitații) până la livrarea cu succes a produsului. Acest mecanism:

- Protejează cumpărătorul, evitând plățile pentru produse nelivrate;
- Protejează vânzătorul, asigurându-i plata doar după îndeplinirea obligațiilor;
- Se realizează complet automatizat, fără intervenția utilizatorilor, prin joburi programate cu Hangfire.

5.3 Experiență real-time cu notificări bidirecționale prin SignalR

O contribuție majoră în ceea ce privește interactivitatea și experiența de utilizare este reprezentată de integrarea tehnologiei SignalR pentru actualizări în timp real.

Astfel, în momentul în care mai mulți utilizatori vizualizează aceeași licitație, orice ofertă nou plasată este transmisă instantaneu către toți participanții prin conexiune WebSocket, fără reîncărcarea paginii. Acest comportament:

- Oferă o experiență imersivă și dinamică, asemănătoare platformelor comerciale consacrate;
- Este implementat complet bidirecțional, cu grupuri de utilizatori pe licitație și huburi SignalR gestionate în backend;
- A fost testat cu succes în condiții reale și optimizează implicarea utilizatorilor în procesul de licitație.

5.4 Automatizarea livrării și deploy-ul în cloud

O contribuție importantă a fost configurarea procesului de livrare continuă (CI/CD) și implementarea arhitecturii de cloud pentru aplicație. Aplicația este structurată în două compo-

nente majore:

- **Backend-ul (API-ul)** este hostat în *Azure App Service*, oferind scalabilitate și integrare completă cu Azure DevOps.
- **Frontend-ul Blazor** este, de asemenea, hostat în *Azure App Service*, livrând o aplicație client modernă.
- **Baza de date PostgreSQL** este hostată în *Railway*, fiind integrată eficient cu infrastructura aplicației.

Pentru livrare continuă, au fost definite două fișiere YAML pentru Azure DevOps:

- *Pipeline-ul API-ului* include etape pentru restaurarea pachetelor, compilare, testare unitară, publicare artifact și deploy automat.
- *Pipeline-ul frontend-ului* gestionează build-ul aplicației Blazor, înlocuirea valorilor din fișierele de configurare, testarea și livrarea codului.

Securitate

- Secretele API-ului sunt stocate în *App Settings* în Azure App Service.
- Variabilele frontend-ului sunt definite ca variabile în Azure DevOps și înlocuite automat în fișierele JSON.

Prin această arhitectură, s-a obținut un proces de livrare automatizat, fiabil și reproductibil, care asigură testare automată, deploy rapid și reducerea riscurilor de eroare în mediu de producție. Această soluție susține dezvoltarea continuă și livrarea frecventă a îmbunătățirilor aplicației.

Capitolul 6

Ghid de utilizare

Interfața aplicației a fost concepută astfel încât să ofere o experiență intuitivă și adaptată fiecărui tip de utilizator: admin, driver și user final. Platforma diferențiază clar rolurile și drepturile fiecărei categorii, oferind funcționalități și interfețe distincte pentru fiecare.

Primul punct de acces în aplicație este reprezentat de o pagină principală, unde utilizatorii sunt invitați să aleagă tipul de autentificare corespunzător. Această pagină afișează trei opțiuni explicite:

- **Login as Admin** – pentru accesul în zona de administrare;
- **Login as Driver** – pentru accesul șoferilor către livrările alocate;
- **Login as User** – pentru utilizatorii finali care doresc să participe la licitații (ca vânzători sau cumpărători).

Această selecție inițială are rolul de a direcționa corect utilizatorul către interfața specifică rolului său, eliminând ambiguitatea și simplificând procesul de autentificare. Fiecare rol are un traseu personalizat în aplicație, cu funcționalități corespunzătoare responsabilităților și nevoilor sale.

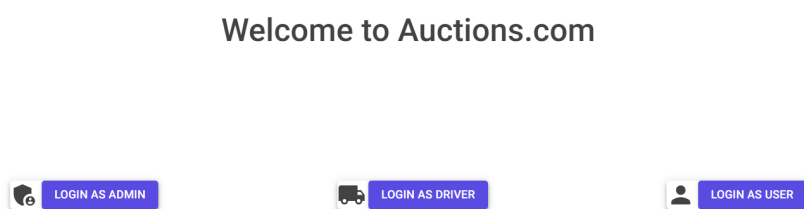


Figura 6.1: Pagina principală

6.1 Contul de User

6.1.1 Pagina de login pentru User

După selectarea opțiunii „Login as User” din pagina principală, utilizatorul este direcționat către o interfață dedicată autentificării. Formularul de login este simplu și intuitiv, solicitând adresa de email și parola utilizatorului. Aceste câmpuri sunt validate automat, iar utilizatorul are opțiunea de a vizualiza parola introdusă prin intermediul funcției „Show password”.

În partea inferioară a formularului este disponibil un link de redirecționare pentru utilizatorii care nu dețin încă un cont în platformă. Prin accesarea acestui link, aceștia sunt ghidați către o pagină de înregistrare, unde pot introduce informațiile necesare pentru a crea un cont de tip cumpărător sau vânzător.

Această pagină de autentificare separată pentru utilizatori contribuie la securizarea accesului și la personalizarea traseului fiecărui rol în cadrul aplicației.

Figura 6.2: Pagina de login pentru useri

6.1.2 Pagina de înregistrare pentru User

După accesarea linkului de înregistrare, utilizatorul este redirecționat către o pagină dedicată creării unui cont nou. Formularul de înregistrare este structurat simplu și intuitiv, solicitând completarea următoarelor informații obligatorii:

- **Email** – adresa de email validă, utilizată ulterior pentru autentificare;
- **Password** – parola de acces în cont;
- **First Name** – prenumele utilizatorului;
- **Last Name** – numele de familie;
- **Address** – adresa de domiciliu.

Butonul **Register** finalizează procesul de înregistrare, trimițând datele către API pentru validare și salvare. În partea de jos a formularului există un link alternativ pentru utilizatorii care au deja cont, permițându-le să revină rapid la pagina de login.

Această etapă asigură crearea contului de tip „user” (cumpărător sau vânzător), oferindu-i acces la funcționalitățile platformei: participarea la licitații, plasarea de oferte sau crearea propriilor anunțuri de licitație.

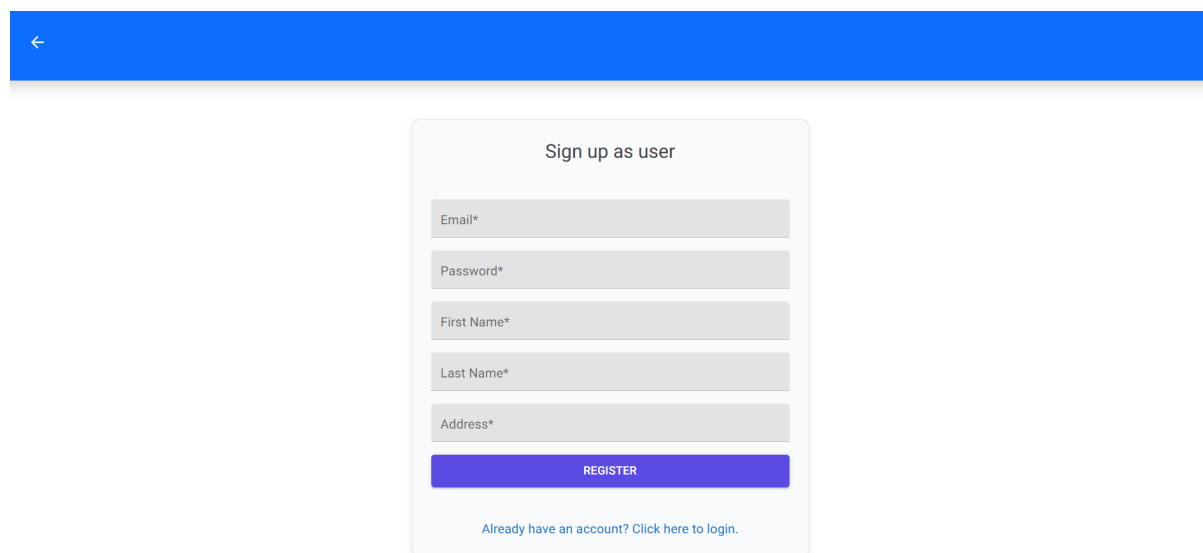
The image shows a mobile application interface for user registration. At the top is a blue header bar with a white back arrow icon. Below the header is a white card with rounded corners titled "Sign up as user". The card contains five text input fields stacked vertically, each with a placeholder label and an asterisk: "Email*", "Password*", "First Name*", "Last Name*", and "Address*". Below these fields is a blue button with the text "REGISTER" in white. At the bottom of the card, there is a link in blue text that reads "Already have an account? Click here to login."

Figura 6.3: Pagina de register pentru useri

6.1.3 Resetarea parolei

Platforma oferă un mecanism de recuperare a parolei pentru utilizatorii care și-au uitat datele de autentificare. Pe această pagină, utilizatorul trebuie să introducă adresa de email asociată contului. Dacă adresa este validă și există în sistem, se trimite automat un link de resetare a parolei pe email. Acest link permite setarea unei noi parole și recuperarea accesului la cont.

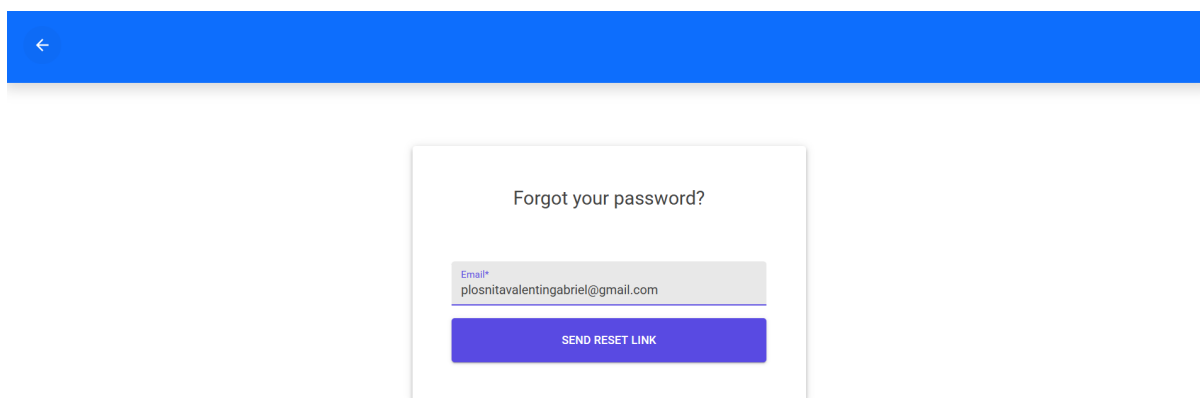


Figura 6.4: Pagina de recuperare a parolei

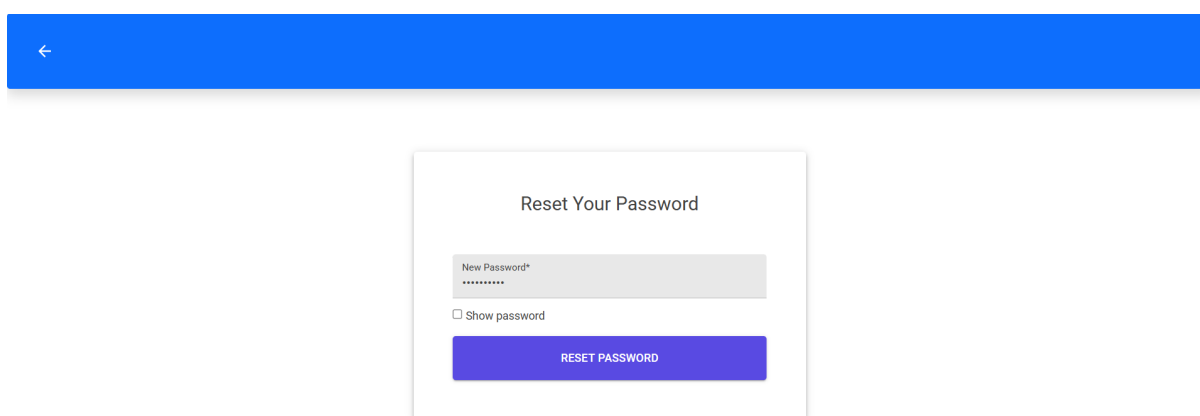


Figura 6.5: Pagina pentru resetarea parolei

6.1.4 Dashboard pentru User

După autentificare, utilizatorul este redirectionat către dashboard-ul principal, care oferă acces rapid la toate funcționalitățile platformei, structurate într-un meniu de navigare intuitiv în partea superioară a ecranului. Acesta include următoarele secțiuni:

- **Home** – revenirea la pagina principală a utilizatorului;
- **Browse Auctions** – afișează lista licitațiilor active, unde utilizatorul poate analiza detalii, plasa oferte și urmări progresul licitației;
- **My Listings** – conține toate licitațiile create de utilizatorul curent, în calitate de vânzător;

- **My Bids** – oferă o privire de ansamblu asupra tuturor licitațiilor la care utilizatorul a participat prin plasarea unei oferte.

Pe partea stângă a interfeței este disponibil un meniu vertical cu secțiuni de configurare personalizate:

- **General Settings** – utilizatorul poate vizualiza și actualiza informațiile de bază ale contului: prenume, nume și adresă;
- **Seller Settings** – aici sunt afișate detaliile contului Stripe conectat (pentru utilizatorii care vând produse), cu opțiuni pentru actualizarea acestora;
- **Customer Settings** – utilizatorul poate vedea metoda de plată salvată în contul Stripe (folosită pentru a plasa oferte) și are opțiunea de a înlocui metoda de plată curentă cu una nouă.

Această structură de interfață permite un control complet asupra activității desfășurate în platformă, atât în calitate de cumpărător, cât și de vânzător, asigurând o experiență coerentă, organizată și sigură.

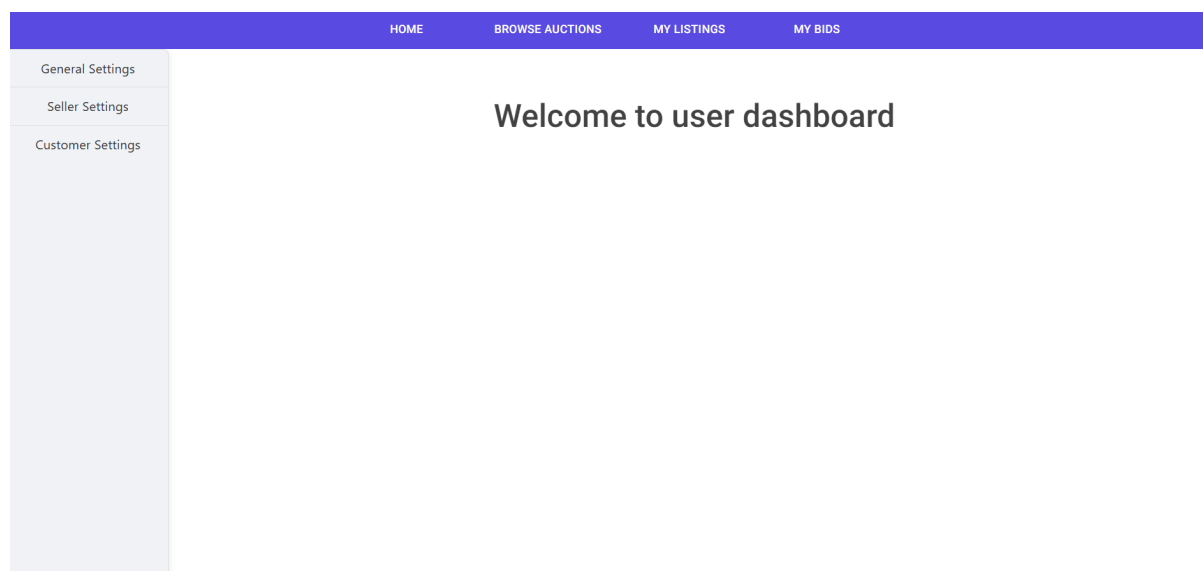


Figura 6.6: User home page

6.1.5 General Settings

În cadrul dashboardului, accesând secțiunea **General Settings**, utilizatorul are posibilitatea de a consulta și modifica informațiile sale personale stocate în platformă. Interfața este simplă și clară, oferind câmpuri editabile pentru:

- **First Name** – prenumele utilizatorului;
- **Last Name** – numele de familie;
- **Address** – adresa de domiciliu (utilizată în contextul livrării obiectelor câștigate).

După modificarea datelor, acestea pot fi salvate prin intermediul butonului **Save Changes**, care actualizează informațiile atât în baza de date, cât și în interfața aplicației.

Tot din această pagină, utilizatorul are acces și la butonul Logout, poziționat vizibil în partea inferioară a interfeței. Acesta oferă o metodă sigură și rapidă de a încheia sesiunea activă.

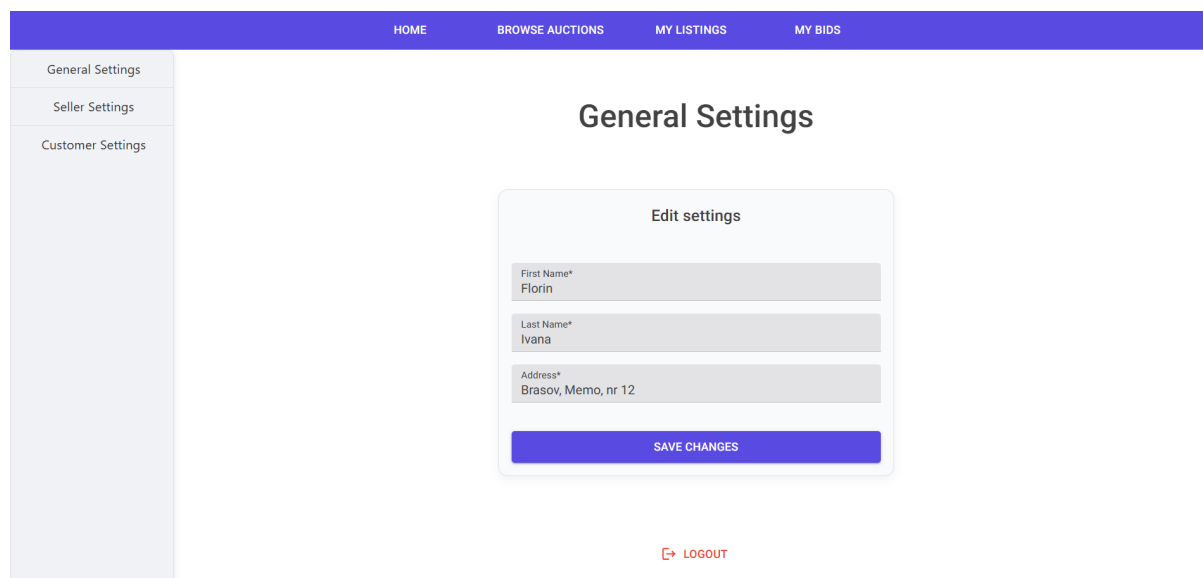


Figura 6.7: Pagină de setări generale pentru utilizator

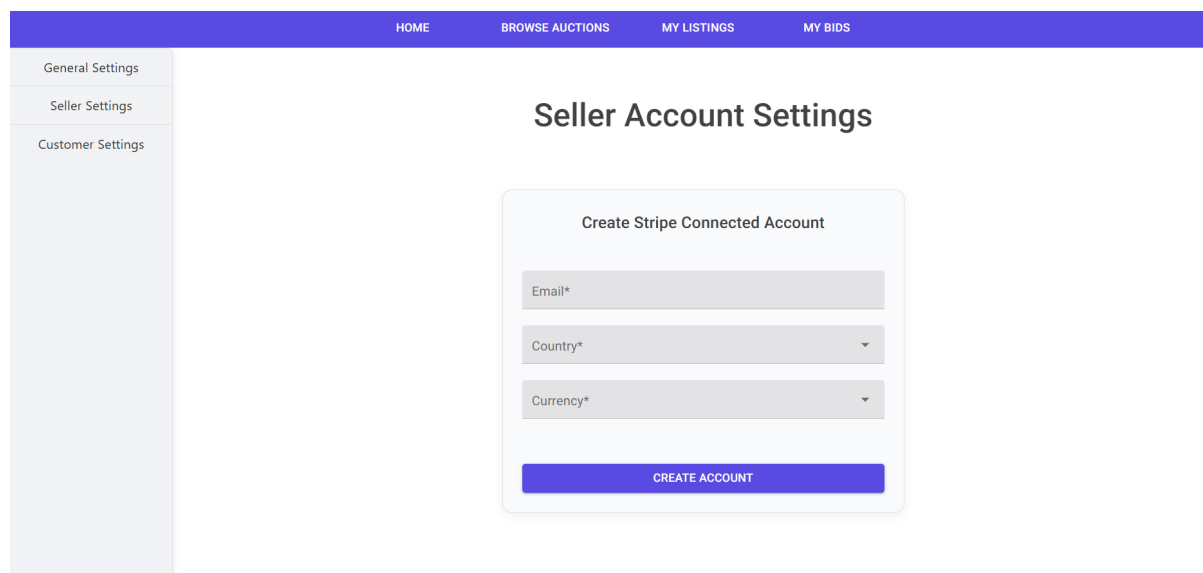
6.1.6 Seller Settings

Accesând secțiunea **Seller Settings**, utilizatorul poate crea sau administra contul Stripe necesar pentru a vinde produse în cadrul platformei. Comportamentul acestei pagini variază în funcție de starea contului curent:

Crearea unui cont de vânzător: Dacă utilizatorul nu are încă asociat un cont de tip vânzător, i se afișează un formular simplificat cu următoarele câmpuri obligatorii:

- **Email** – adresa care va fi asociată contului Stripe;
- **Country** – țara de rezidență a utilizatorului;
- **Currency** – moneda în care vor fi efectuate plățile.

După completarea formularului, prin acționarea butonului **Create Account**, este inițiată automat procedura de creare a unui cont Stripe conectat (*Connected Account*), necesar pentru a primi fonduri în urma vânzării prin licitație.



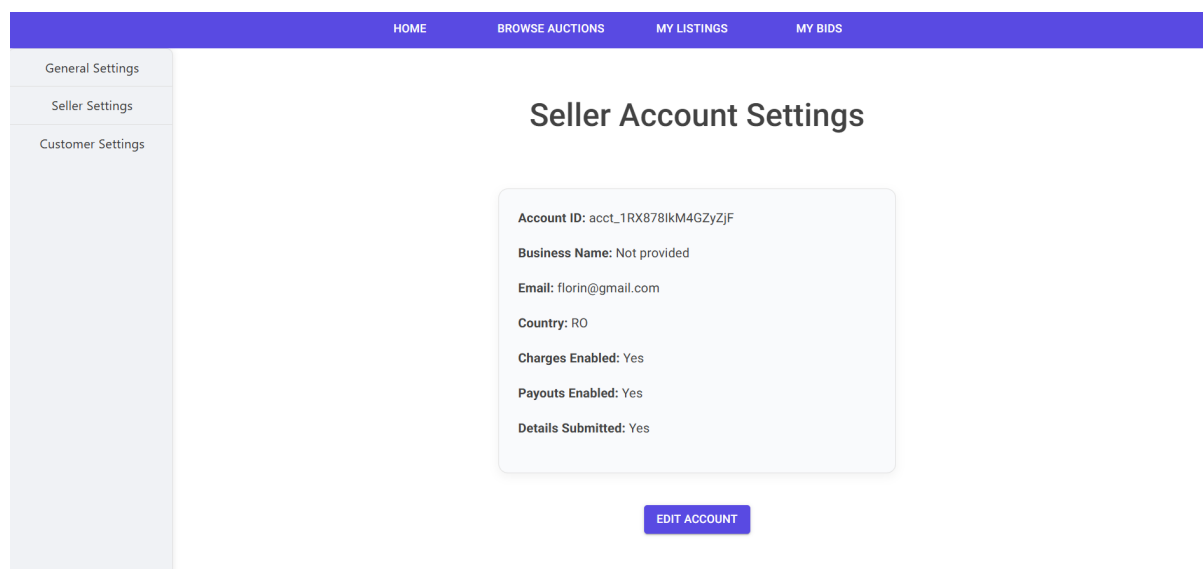
The screenshot shows the 'Seller Account Settings' page. On the left is a sidebar with 'General Settings', 'Seller Settings', and 'Customer Settings'. The main content area is titled 'Seller Account Settings' and contains a form titled 'Create Stripe Connected Account'. The form has three input fields: 'Email*', 'Country*', and 'Currency*', each with a dropdown arrow. At the bottom of the form is a blue button labeled 'CREATE ACCOUNT'.

Figura 6.8: Formular pentru crearea contului de vânzător

Vizualizarea contului existent În cazul în care un cont Stripe a fost deja creat, interfața afișează detaliile contului sub formă structurată:

- ID-ul contului;
- Adresa de email asociată;
- Țara și starea activă a contului (ex: „Charges Enabled”, „Payouts Enabled”).

Utilizatorul are posibilitatea de a revizui sau modifica informațiile contului folosind butonul **Edit Account**.



The screenshot shows the 'Seller Account Settings' page with the same sidebar. The main content area displays account details in a structured list:

- Account ID: acct_1RX878IkM4GZyZjF
- Business Name: Not provided
- Email: florin@gmail.com
- Country: RO
- Charges Enabled: Yes
- Payouts Enabled: Yes
- Details Submitted: Yes

At the bottom of the details box is a blue button labeled 'EDIT ACCOUNT'.

Figura 6.9: Detaliile contului de vânzător

Editarea contului prin Stripe La apăsarea butonului de editare, utilizatorul este redirecționat către un formular oficial oferit de Stripe, unde poate actualiza:

- Datele personale (nume, adresă, telefon, dată naștere);
- Detaliile bancare (IBAN, conturi bancare suplimentare);
- Detalii publice de business.

Acest formular este protejat și integrat în fluxul Stripe, iar validarea are loc în timp real.

The screenshot shows the Stripe Customer Settings page for a user named 'Auction Test'. The page is divided into three main sections: Public details, Personal details, and Payout details. The Public details section shows the public descriptor 'AUCTIONS.COM'. The Personal details section shows the user's name 'Florin Iana', email address 'florin@gmail.com', date of birth 'Born on 2 June 2003', and address 'Memorandului 12, 123456 Brasov RO'. The Payout details section shows the payout method 'STRIPE TEST BANK' with a currency of 'RON' and a card number '1100000000 0000 0000 0000 0000'. There are 'Edit' links for each section. At the bottom, there is a 'Confirm' button and a note: 'By confirming, you agree that the information provided is complete and correct.'

Figura 6.10: Pagina de editare a detaliilor contului de vânzător

6.1.7 Customer Settings

Secțiunea **Customer Settings** permite utilizatorului să gestioneze metodele de plată utilizate pentru participarea la licitații. Acest modul este esențial pentru funcționarea sistemului de oferte, deoarece fiecare utilizator trebuie să aibă o metodă de plată validă salvată în Stripe înainte de a putea plasa oferte. Interfața acestei secțiuni este organizată în două taburi distincte:

Card Details: În această vizualizare, utilizatorul poate consulta metoda de plată curentă asociată contului său Stripe. Informațiile sunt afișate sub formă de card virtual, incluzând:

- tipul cardului (ex: VISA);
- ultimele cifre din numărul cardului;
- data de expirare.

Această reprezentare vizuală oferă o confirmare clară a metodei active, fără a expune detalii sensibile.

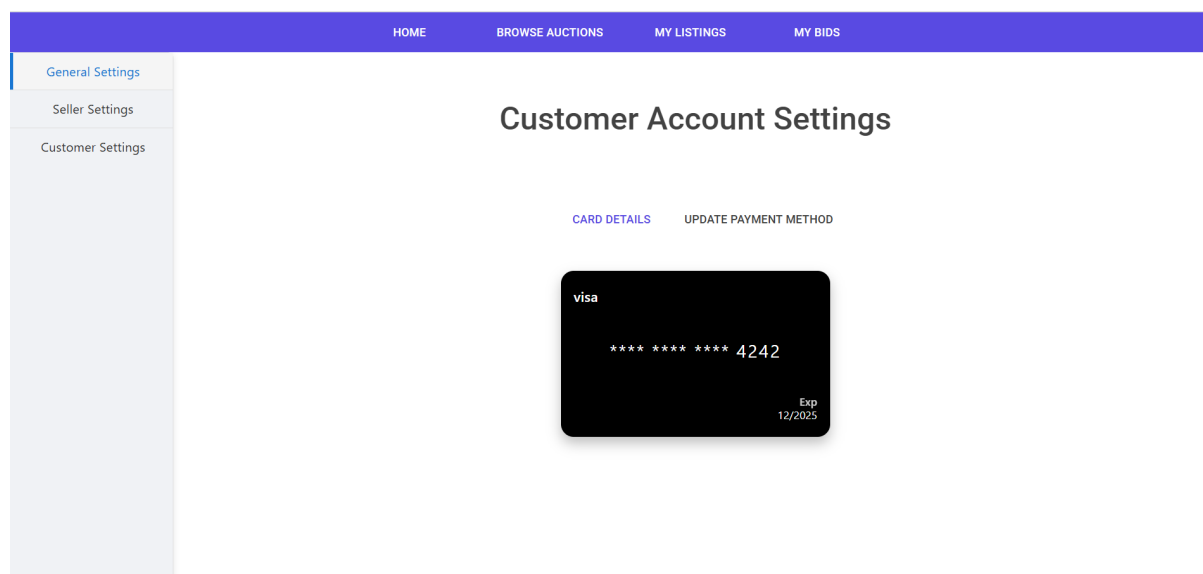
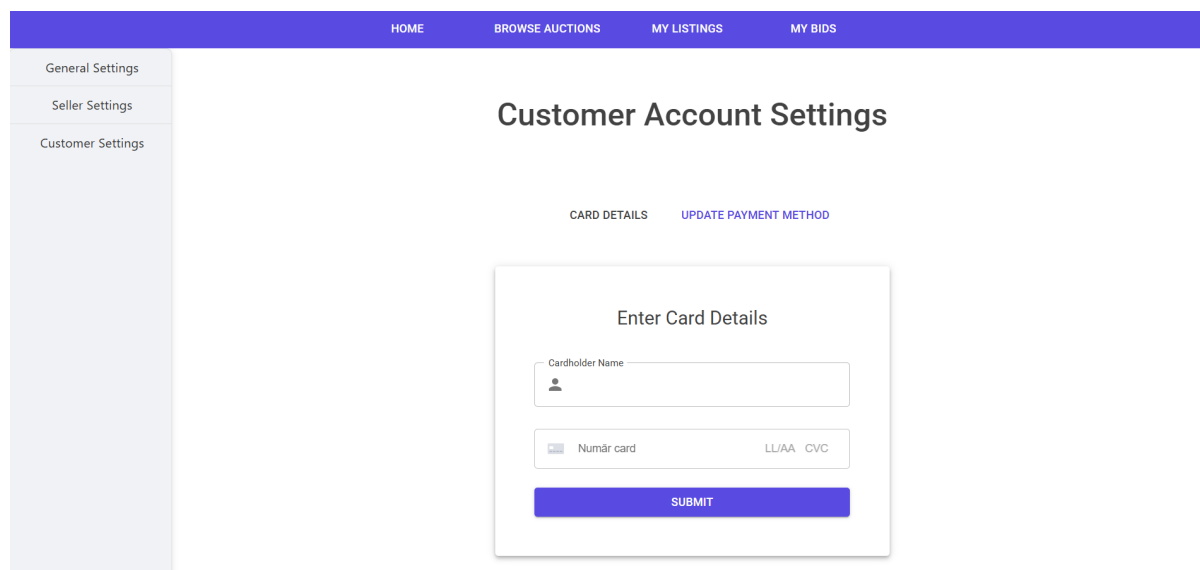


Figura 6.11: Tab cu detaliile metodei curente de plată

Update Payment Method: Dacă utilizatorul dorește să adauge sau să înlocuiască metoda de plată existentă, poate accesa al doilea tab. Aici este prezentat un formular securizat, unde se introduc:

- numele titularului cardului;
- numărul complet al cardului;
- data de expirare;
- codul CVC.

Odată trimis formularul prin butonul **Submit**, datele sunt transmise direct către Stripe, care le salvează în siguranță și asigură conformitatea cu standardele internaționale de securitate (PCI-DSS). Aplicația păstrează doar identifiantii cardului, fără a stoca local detalii bancare sensibile.



The screenshot shows the 'Customer Account Settings' page. On the left is a sidebar with 'General Settings', 'Seller Settings', and 'Customer Settings'. The main content area has a title 'Customer Account Settings' and two tabs: 'CARD DETAILS' (active) and 'UPDATE PAYMENT METHOD'. Below the tabs is a form titled 'Enter Card Details' with fields for 'Cardholder Name', 'Număr card', and 'LL/AA CVC', followed by a 'SUBMIT' button.

Figura 6.12: Tab cu formularul pentru adăugarea metodei de plată

Această funcționalitate este crucială pentru buna desfășurare a licitațiilor, asigurând că fiecare ofertă este garantată financiar și poate fi convertită automat într-o tranzacție validă în momentul câștigării.

6.1.8 Browse Auctions

Secțiunea **Browse Auctions**, accesibilă din bara principală de navigare, oferă utilizatorului o prezentare vizuală și filtrabilă a tuturor licitațiilor active în platformă. Această pagină reprezintă punctul de plecare pentru explorarea produselor disponibile și plasarea de oferte.

Interfața este împărțită în două componente principale:

Filtru pe categorii: În partea superioară, sunt afișate categoriile principale ale produselor: ceasuri, băuturi, cărți, artă, sport, antichități, bijuterii, modă etc. Utilizatorul poate selecta o anumită categorie pentru a restrânge afișarea doar la licitațiile din acel domeniu. Navigarea între categorii este facilitată prin butoane de scroll lateral.

Grid-ul cu licitații: Sub zona de categorii, sunt afișate carduri individuale pentru fiecare licitație disponibilă. Fiecare card include:

- imaginea produsului;
- denumirea acestuia;
- prețul curent (valoarea celei mai mari oferte active).

Cardurile sunt interactive și pot fi selectate pentru a accesa detalii complete despre produsul licitat, istoricul ofertelor și opțiunea de a plasa o ofertă nouă.

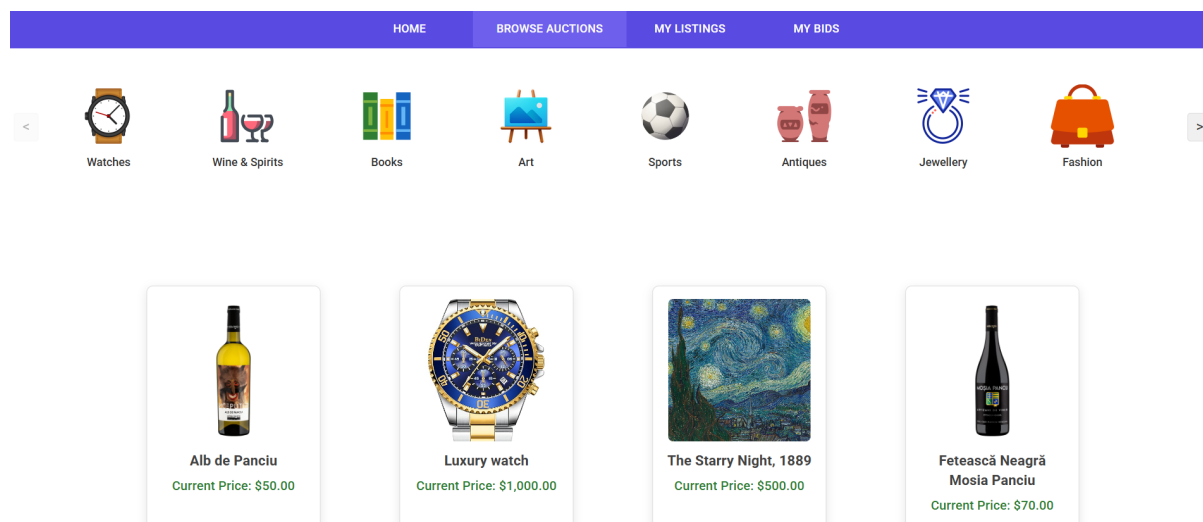


Figura 6.13: Pagină cu licitații

6.1.9 Accesarea unei licitații

Accesând o licitație din pagina „Browse Auctions”, utilizatorul este direcționat către o pagină dedicată, unde sunt afișate toate detaliile esențiale legate de produsul scos la licitație și progresul acesteia.

Structura paginii Structura paginii este organizată pe mai multe secțiuni:

Prezentarea generală a produsului

- Titlul licitației este afișat proeminent în partea superioară.
- În partea centrală, utilizatorul poate vizualiza imaginile produsului, printr-un slider interactiv.
- În dreapta imaginii este afișată descrierea licitației, redactată de către vânzător.

Informații despre vânzător

- adresa de email a vânzătorului;
- numele complet;
- adresa de domiciliu

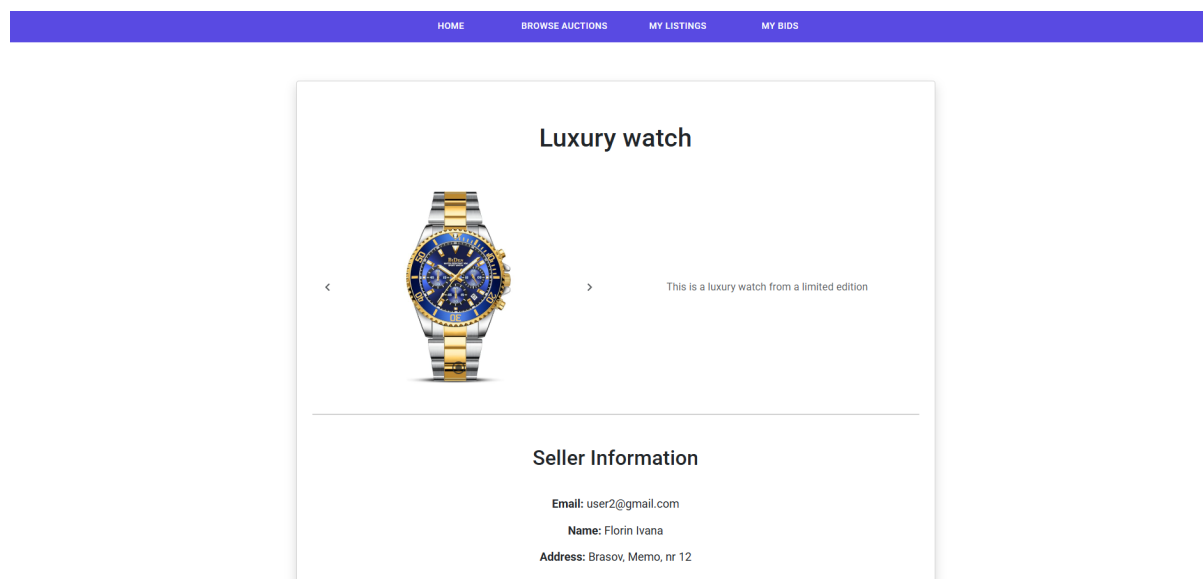


Figura 6.14: Pagină pentru plasarea ofertelor

Istoricul ofertelor

- Se afișează ultimele oferte plasate, cu:
 - identificatorul ofertantului;
 - momentul plasării;
 - suma oferită.

Numărătoare inversă

- Un cronometru vizibil indică timpul rămas până la închiderea licitației (zile, ore, minute, secunde).
- Acesta se actualizează în timp real și contribuie la dinamica procesului de licitare.

Informații financiare

- Prețul curent al licitației – reflectă cea mai mare ofertă activă;
- Incrementul minim de ofertare – valoarea minimă cu care trebuie depășită oferta precedentă.

Formular de creare a ofertei

- Utilizatorul poate introduce o sumă validă în câmpul dedicat („Your Bid Amount”) și poate confirma acțiunea prin apăsarea butonului Place Bid.
- Sistemul validează automat valoarea introdusă și aplică logica de incrementare stabilită de vânzător.

Această pagină este esențială în experiența utilizatorului cumpărător, oferind o imagine completă, transparentă și interactivă asupra fiecărei licitații. Datorită integrării cu SignalR, orice ofertă nouă plasată de alți utilizatori este vizibilă în timp real, fără a fi necesar un refresh manual al paginii.

Name: Florin Ivana
Address: Brasov, Memo, nr 12

Bidding History

Bidder 4 2 min ago \$1,200.00

Time Remaining

1d 3h 12m 16s

Current Price: \$1,200.00
Minimum Bid Increment: \$50.00

Your Bid Amount RON

PLACE BID

Figura 6.15: Pagină pentru plasarea ofertelor

6.1.10 My Listings

Această pagină afișează toate licitațiile create de utilizator. Informațiile sunt prezentate într-un tabel ce include: titlul licitației, imaginea produsului, prețul curent, data de încheiere și statusul (InProgress, InTransit, Completed, Canceled).

Utilizatorul are la dispoziție:


- un meniu de filtrare după statusul licitației;
- butonul View pentru vizualizarea detaliilor unei licitații;
- butonul Create New Auction pentru inițierea unei licitații noi.

HOME BROWSE AUCTIONS MY LISTINGS MY BIDS

My Listings

[+ CREATE NEW AUCTION](#)

Auction Status

Id	Title	Image	Current Price	End Time	Status	
1	Luxury watch		600	16.06.2025 23:59	InProgress	VIEW

Rows per page: 5 1-1 of 1 < > >|

Figura 6.16: Pagină cu licitațiile create de utilizator

6.1.11 Pagină pentru crearea unei licitații

Această pagină permite utilizatorilor cu cont Stripe de tip vânzător asociat să creeze o nouă licitație. Formularul conține următoarele câmpuri:

- Title – titlul licitației;
- Description – descrierea produsului;
- Starting Price – prețul de pornire;
- Min Bid Increment – valoarea minimă cu care poate fi crescută o ofertă;
- End Time – data și ora la care licitația se încheie;
- Category – selecția categoriei produsului;
- Image Upload – permite adăugarea de imagini pentru produsul licitat.

Butonul Create Auction salvează licitația doar dacă toate câmpurile sunt valide și utilizatorul are un cont Stripe conectat.

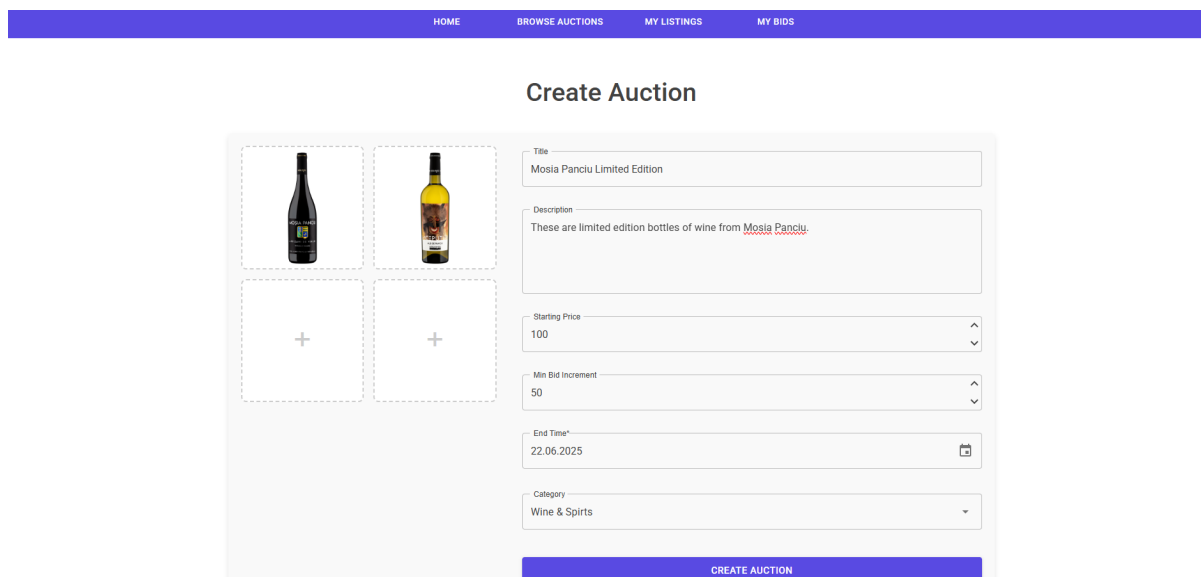


Figura 6.17: Pagină pentru crearea unei licitații

6.1.12 My Bids

Această pagină afișează toate licitațiile la care utilizatorul a plasat cel puțin o ofertă. Pentru fiecare licitație sunt prezentate informații esențiale precum:

- Id – identificatorul unic al licitației;
- Title – titlul produsului;
- Image – imagine reprezentativă;
- Current Price – prețul actual;
- End Time – data și ora de finalizare;

- **Status** – starea curentă (InProgress, Completed, Canceled etc.).

Utilizatorul poate:

- filtra licitațiile în funcție de status;
- naviga către pagina unei licitații apăsând pe butonul **View**, unde poate vedea toate detaliile sau plasa o nouă ofertă.

HOME	BROWSE AUCTIONS	MY LISTINGS	MY BIDS
------	-----------------	-------------	---------


My Bids						
Auction Status						
Id	Title	Image	Current Price	End Time	Status	
1	Luxury watch		600	16.06.2025 23:59	InProgress	VIEW
Rows per page: 5 1-1 of 1 < < > >						

Figura 6.18: Pagină cu licitațiile la care utilizatorul a licitat

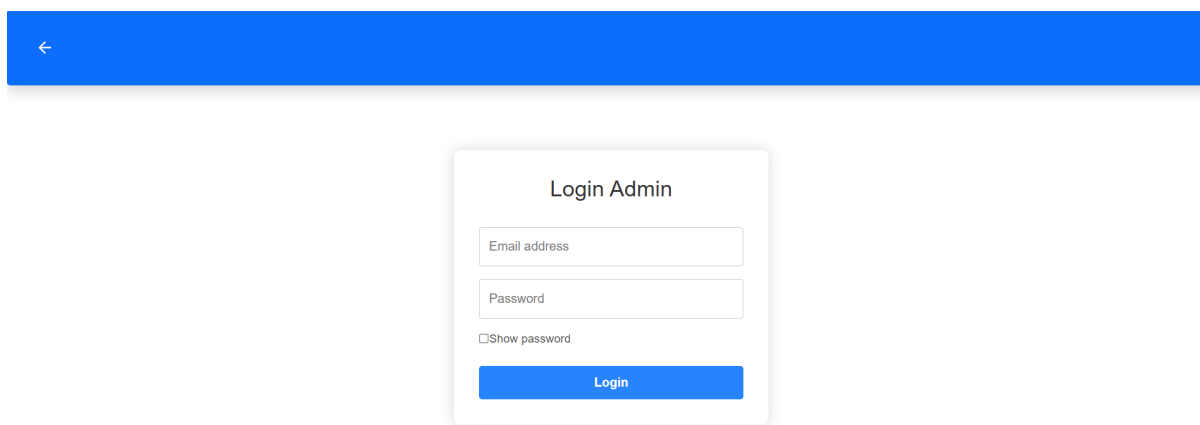
6.2 Contul de Admin

6.2.1 Pagina de login pentru Admin

Pagina de autentificare dedicată administratorilor permite accesul în interfața de administrare a platformei. Formularul conține două câmpuri esențiale:

- **Email address** – adresa de email a contului de administrator;
- **Password** – parola asociată contului.

Este disponibilă opțiunea de afișare a parolei prin checkbox-ul „Show password”. După completarea corectă a datelor, administratorul este redirecționat către dashboard-ul de administrare, unde are acces la funcționalitățile specifice rolului său.



The image shows a web interface for an admin login. At the top is a solid blue horizontal bar containing a white left-pointing arrow. Below this bar is a white rectangular login form with a subtle drop shadow. The form is titled 'Login Admin' in a small, dark font. It contains two input fields: 'Email address' and 'Password'. Below the password field is a checkbox labeled 'Show password'. At the bottom of the form is a solid blue button with the word 'Login' in white text.

Figura 6.19: Pagina de login pentru Admin

Dashboard pentru Admin După autentificare, administratorul este redirecționat către pagina principală a dashboard-ului, unde are acces rapid la funcționalitățile principale prin bara de navigare superioară:

- **Home** – întoarcere la această pagină de start;
- **Admins** – gestionarea conturilor de administratori;
- **Drivers** – gestionarea conturilor de șoferi;
- **Categories** – gestionarea categoriilor de produse licitate;
- **Auctions** – gestionarea licitațiilor active.

În centrul paginii este afișat un mesaj de întâmpinare, iar în partea inferioară, butonul Logout permite deconectarea rapidă din contul curent. Această pagină acționează ca punct de plecare pentru toate operațiunile administrative.

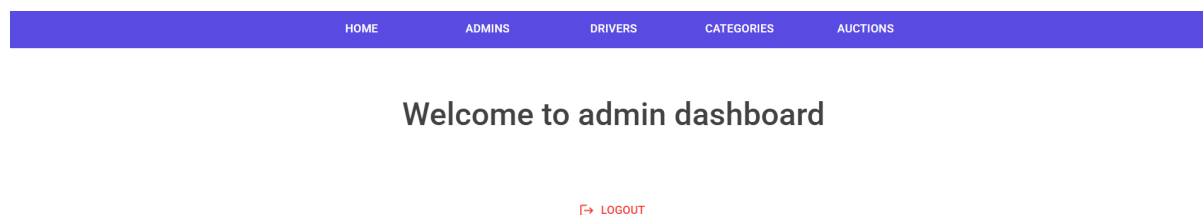


Figura 6.20: Dashboard pentru Admin

6.2.2 Admins Dashboard

Această secțiune este accesibilă doar SuperAdminilor și permite gestionarea utilizatorilor cu rol de administrator. În cadrul acestei pagini:


- Sunt afișate toate conturile de tip Admin și SuperAdmin, împreună cu date precum ID-ul, emailul și rolul;
- SuperAdminii au posibilitatea de a șterge conturi Admin existente folosind iconița roșie de tip coș de gunoi din coloana Delete;
- Conturile de tip SuperAdmin nu pot fi șterse;
- Prin apăsarea butonului Create New Admin, se poate crea un nou cont de administrator.

Această funcționalitate contribuie la un control centralizat al accesului administrativ în cadrul aplicației.

HOME	ADMINS	DRIVERS	CATEGORIES	AUCTIONS
------	--------	---------	------------	----------

Admins

+ CREATE NEW ADMIN

Id	Email	Role	Delete
1	admin1@gmail.com	SuperAdmin	
3	admin2@gmail.com	Admin	

Rows per page: 5 1-2 of 2 |< < > >|

Figura 6.21: Admins Dashboard

6.2.3 Pagină pentru crearea conturilor de Admin

Această pagină permite administratorilor să creeze noi conturi de administrator în platformă. Formularul de creare conține următoarele câmpuri:

- Email – adresa de email a noului administrator;
- Password – parola contului;
- Show password – opțional, pentru afișarea parolei introdusă.

După completarea formularului și apăsarea butonului Create, se creează un nou cont cu rolul de Admin, care va putea accesa dashboardul administrativ.

HOME	ADMINS	DRIVERS	CATEGORIES	AUCTIONS
------	--------	---------	------------	----------

Create new Admin

Email*

Password*

☐ Show password

CREATE

Figura 6.22: Pagină pentru crearea conturilor de Admin

6.2.4 Drivers Dashboard



Această pagină este accesibilă administratorilor și afișează lista completă a conturilor de șoferi din aplicație. Tabelul conține adresa de email și opțiunea de ștergere pentru fiecare șofer. Administratorii pot efectua următoarele acțiuni:

- Vizualizarea tuturor șoferilor înregistrați;
- Ștergerea unui șofer existent;
- Crearea unui nou șofer prin apăsarea butonului **Create New Driver**, care deschide formularul de înregistrare.

[HOME](#)
[ADMINS](#)
[DRIVERS](#)
[CATEGORIES](#)
[AUCTIONS](#)

Drivers

+ CREATE NEW DRIVER

Id	Email	Delete
1	driver1@gmail.com	
2	driver2@gmail.com	

Rows per page:
5
1-2 of 2
|<
<
>
>|

Figura 6.23: Drivers Dashboard

6.2.5 Pagină pentru crearea conturilor de Driver

Această pagină este folosită de un administrator pentru a crea un cont nou de șofer. Formularul include două câmpuri obligatorii:

- Email – adresa de email a șoferului;
- Password – parola contului.

Există o opțiune de afișare a parolei introduse prin bifarea căsuței **Show password**, iar contul este creat prin apăsarea butonului **Create**. Accesul la această funcționalitate este restricționat doar pentru utilizatorii cu rol de Admin sau SuperAdmin.

[HOME](#) [ADMINS](#) [DRIVERS](#) [CATEGORIES](#) [AUCTIONS](#)

Create new Driver

Email*

Password*

☐ Show password

CREATE

Figura 6.24: Pagină pentru crearea conturilor de Driver

6.2.6 Categories Dashboard

Această pagină este disponibilă pentru utilizatorii cu rol de administrator și permite gestionarea categoriilor existente în aplicație. Fiecare rând din tabel afișează:

- **Id** – identificatorul unic al categoriei;
- **Name** – denumirea categoriei;
- **Image** – o imagine reprezentativă pentru categorie;
- **Edit** – un buton pentru modificarea denumirii sau imaginii categoriei.

Administratorii pot crea o categorie nouă folosind butonul **Create New Category**, poziționat deasupra tabelului.

[HOME](#) [ADMINS](#) [DRIVERS](#) [CATEGORIES](#) [AUCTIONS](#)

Categories

+ CREATE NEW CATEGORY







Id	Name	Image	Edit
8	Watches		
9	Wine & Spirits		
10	Books		

Figura 6.25: Categories Dashboard

6.2.7 Pagină pentru crearea unei categorii

Această pagină este accesibilă administratorilor și permite adăugarea unei categorii noi în platformă. Formularul conține următoarele câmpuri:

- **Category Name** – numele categoriei;
- **Upload Image** – un buton care permite încărcarea unei imagini reprezentative (formate acceptate: JPG, PNG, dimensiune maximă: 2MB).

După completarea datelor, administratorul apasă butonul **Submit** pentru a salva noua categorie în sistem.

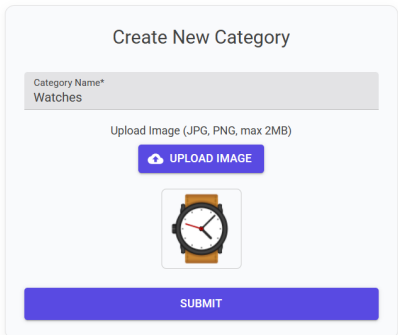


Figura 6.26: Pagină pentru crearea unei categorii

6.2.8 Pagină pentru editarea unei categorii

Această pagină este destinată administratorilor și permite editarea unei categorii existente. Sunt disponibile următoarele funcționalități:

- Modificarea numelui categoriei;
- Încărcarea unei noi imagini reprezentative (format JPG/PNG, maxim 2MB);
- Confirmarea modificărilor prin apăsarea butonului **Submit**.

După salvare, modificările sunt propagate în mod automat în toate zonele aplicației care folosesc categoria respectivă.

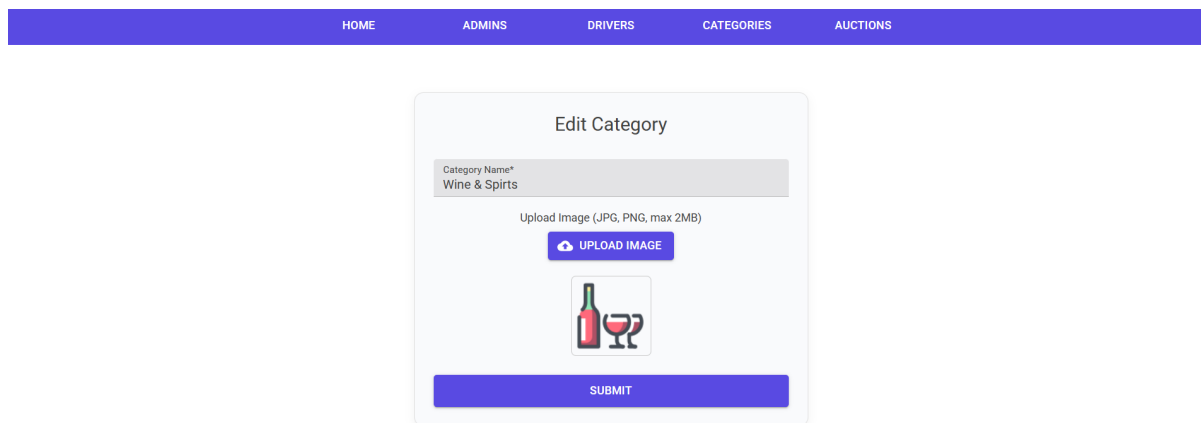


Figura 6.27: Pagină pentru editarea unei categorii

6.2.9 Auctions Dashboard

Această pagină este accesibilă doar administratorilor și afișează toate licitațiile aflate în starea *InTransit*, adică produsele vândute care trebuie livrate. Fiecare licitație este afișată sub formă de rând în tabel și include următoarele informații:

- Id, Titlu, imagine reprezentativă;
- Prețul curent, ora de finalizare și statusul;
- Butonul View care permite accesarea detaliilor și asignarea unui șofer pentru livrare.


În partea superioară există un meniu dropdown pentru filtrarea licitațiilor în funcție de statusul asignării șoferului:

- Ignore – afișează toate licitațiile fără filtru;
- No Driver – afișează doar licitațiile care nu au încă un șofer asignat;
- Any Driver – afișează doar licitațiile care au deja un șofer asociat.

HOME	ADMINS	DRIVERS	CATEGORIES	AUCTIONS
------	--------	---------	------------	----------

Auctions in transit

Driver Filter
Ignore

Id	Title	Image	Current Price	End Time	Status	
1	Luxury watch		600	16.06.2025 23:59	InTransit	VIEW

Rows per page: 5 1-1 of 1

Figura 6.28: Auctions Dashboard


6.2.10 Asignarea unui șofer pentru livrarea unei licitații

Această pagină permite administratorilor să asigneze un șofer pentru livrarea unui produs câștigat în urma unei licitații. Sunt afișate toate detaliile relevante ale licitației, inclusiv prețurile, perioada de desfășurare și informațiile despre vânzător și câștigător. Dacă licitația nu are un șofer asignat, în secțiunea *Driver Information* apare un mesaj corespunzător.

Administratorul poate selecta dintr-un dropdown adresa de email a șoferului dorit și poate confirma alegerea prin apăsarea butonului **Submit**. Odată ce șoferul este asignat, licitația va deveni vizibilă în dashboardul acestuia.

HOME	ADMINS	DRIVERS	CATEGORIES	AUCTIONS
------	--------	---------	------------	----------

Luxury watch



Luxury watch limited edition

Starting Price: 500

Current Price: 600

Start Time: duminică, 15 iunie 2025 03:04

End Time: luni, 16 iunie 2025 23:59

Status: InTransit

Figura 6.29: Pagină pentru asignarea șoferului

Starting Price: 500
Current Price: 600
Start Time: duminică, 15 Iunie 2025 03:04
End Time: luni, 16 Iunie 2025 23:59
Status: **In Transit**

Seller Information	Driver Information	Winner Information
Email: plosnitavalentingabriel@gmail.com Name: Gabi Plosnita Address: Bucuresti, Carpatilor, 13	No driver assigned.	Email: gabiplosnita6@gmail.com Name: Gabi Taylor Address: Vrancea, Focsani, Eroilor, 14

Current Driver

Assign Driver
driver1@gmail.com

SUBMIT

Figura 6.30: Pagină pentru asignarea șoferului

6.3 Contul de Driver

6.3.1 Pagina de login pentru Driver

Interfața conține:

- Câmp pentru adresa de email;
- Câmp pentru parolă;
- Checkbox pentru afișarea parolei în clar;
- Buton Login care validează credențialele.

←

Login Driver

Email address

Password

☐ Show password

Login

Figura 6.31: Pagina de login pentru Driver

6.3.2 Driver Dashboard

Aceasta este pagina principală afișată după autentificarea unui utilizator cu rol de *Driver*. Interfața este simplificată și destinată exclusiv șoferilor care livrează produsele câștigate prin licitații.

Funcționalități:

- Afișarea unui mesaj de întâmpinare;
- Meniu de navigare minimalist (Home, Auctions);
- Buton de logout.

Pagina asigură accesul rapid la comenzile active care au fost asignate șoferului curent. Din această pagină, șoferul poate naviga către licitațiile care trebuie livrate.

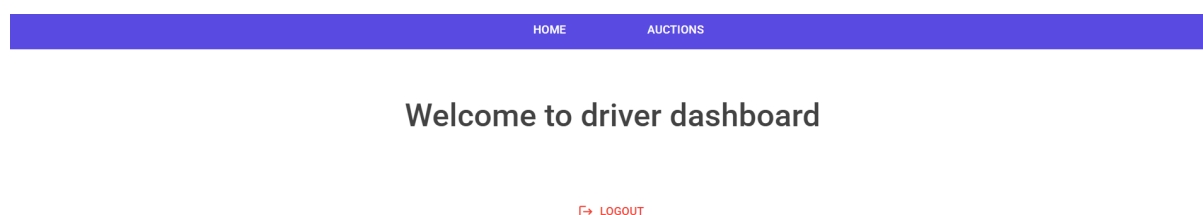


Figura 6.32: Driver Dashboard

6.3.3 Pagină cu licitațiile asignate șoferului

Aceasta este pagina în care un utilizator cu rolul *Driver* poate vizualiza toate licitațiile care i-au fost asignate pentru livrare. Licitațiile afișate au statusul *InTransit* și sunt deja câștigate de un cumpărător.

Funcționalități principale:

- Listarea tuturor licitațiilor asignate șoferului;
- Afișarea detaliilor de bază pentru fiecare licitație: titlu, imagine, preț curent, dată de finalizare, status;
- Buton *View* care trimite către pagina cu detalii ale licitației, unde șoferul poate marca livrarea ca fiind completă.


HOME AUCTIONS						
Assigned Auctions						
Id	Title	Image	Current Price	End Time	Status	
1	Luxury watch		600	16.06.2025 23:59	InTransit	VIEW
Rows per page: 5 1-1 of 1 < > >						

Figura 6.33: Pagină cu licitațiile asignate șoferului

6.3.4 Pagina de finalizare a livrării

După ce o licitație i-a fost asignată unui șofer, aceasta apare în dashboard-ul său. Accesând butonul View, șoferul este direcționat către pagina de detalii a licitației, unde poate vizualiza toate informațiile necesare: imaginea produsului, detalii despre prețuri și perioadă, precum și informații despre vânzător și câștigător.

În partea de jos a paginii se află un buton Complete, care permite șoferului să confirme că produsul a fost livrat cu succes către cumpărător. Apăsarea acestui buton marchează licitația ca fiind Completed, iar în backend se declanșează procesul de transfer al banilor din contul platformei de licitații către contul vânzătorului, prin Stripe.

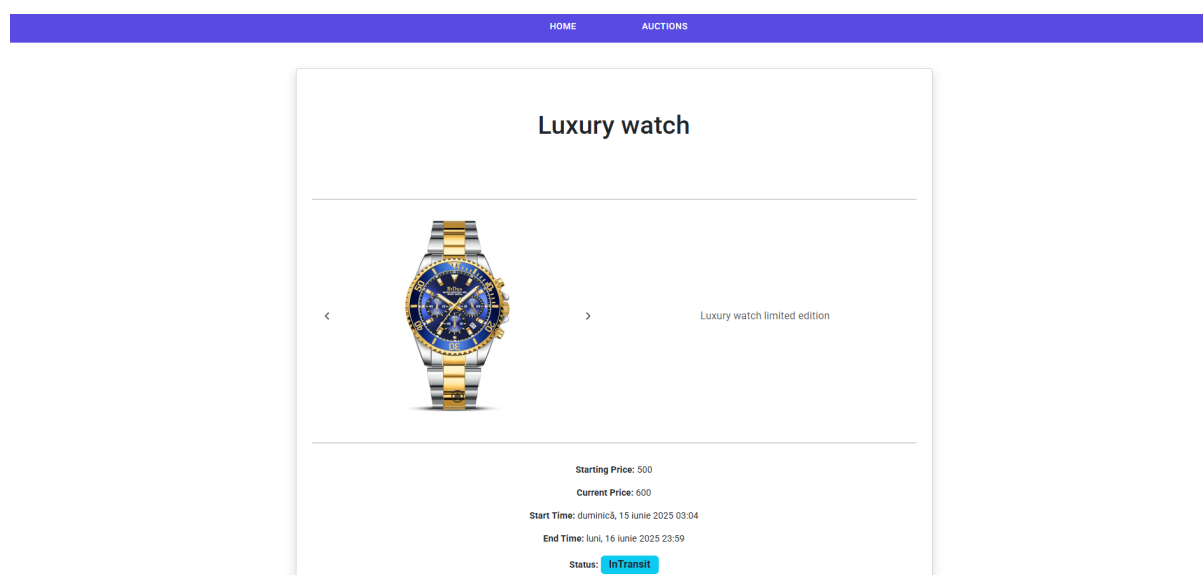



Figura 6.34: Pagina de finalizare a livrării



Luxury watch limited edition

Starting Price: 500

Current Price: 600

Start Time: duminică, 15 iunie 2025 03:04

End Time: luni, 16 iunie 2025 23:59

Status: In Transit

Seller Information

Email: plosnitavalentingabriel@gmail.com

Name: Gabi Plosnita

Address: Bucuresti, Carpatilor, 13

Driver Information

ID: 1

Email: driver1@gmail.com

Winner Information

Email: gabiplosnita6@gmail.com

Name: Gabi Taylor

Address: Vrancea, Focsani, Eroilor, 14

COMPLETE

Figura 6.35: Pagina de finalizare a livrării

Pagina 72

Capitolul 7

Concluzii și perspective de dezvoltare

Concluzii

Lucrarea de față a avut ca obiectiv dezvoltarea unei aplicații web moderne de tip platformă de licitații, destinată vânzării și achiziționării de produse prin mecanisme competitive de ofertare. Proiectul a fost implementat utilizând tehnologii de actualitate precum BŁazor, ASP.NET Core, Entity Framework Core și PostgreSQL, oferind o soluție scalabilă, sigură și eficientă.

Principalele funcționalități implementate sunt:

- autentificarea și autorizarea utilizatorilor pe roluri (User, Admin, Driver, SuperAdmin);
- listarea, vizualizarea și crearea de licitații;
- sistem de ofertare în timp real cu actualizări prin SignalR;
- integrarea cu Stripe pentru procesarea securizată a plăților;
- sistem de livrare gestionat prin asignarea șoferilor;
- interfețe separate pentru utilizatori, administratori și șoferi;
- deploy automat în Azure prin Azure DevOps, cu baze de date hostate în Railway.

Modelul de arhitectură multi-strat și utilizarea de background jobs cu Hangfire au permis o separare clară a responsabilităților și o execuție fiabilă a logicii critice.

Perspective de dezvoltare

Platforma dezvoltată oferă o bază solidă care poate fi extinsă și îmbunătățită pe termen mediu și lung. Printre direcțiile viitoare de dezvoltare se pot enumera:

- **Modul de notificări:** integrarea unui sistem de notificări prin email sau push (ex: notificare la o nouă ofertă, câștigarea unei licitații, confirmarea livrării).
- **Sistem de evaluare și recenzii:** utilizatorii ar putea oferi feedback pentru vânzători și șoferi, contribuind astfel la un ecosistem bazat pe încredere.

- **Optimizarea UI/UX:** adaptarea interfeței pentru dispozitive mobile și îmbunătățirea accesibilității.
- **Extinderea metodei de plată:** adăugarea de suport pentru alte platforme de plată (ex: PayPal, Revolut).
- **Sistem de licitație automată:** oferirea posibilității de a seta o ofertă maximă, iar sistemul să liciteze automat în numele utilizatorului.
- **Export și raportare:** generarea de rapoarte administrative privind numărul de licitații, venituri, livrări, etc.
- **Monitorizarea activităților:** integrarea cu servicii externe (ex: Application Insights, Log Analytics) pentru audit și depanare.

Concluzie generală

Realizarea acestei aplicații a reprezentat o oportunitate valoroasă de aprofundare a cunoștințelor în dezvoltarea web modernă, în modelarea bazelor de date relaționale și în integrarea serviciilor externe. Platforma poate servi drept fundament pentru un produs real scalabil și comercializabil în viitor, cu multiple posibilități de extindere.

Capitolul 8

Bibliografie

Bibliografia cuprinde o selecție de surse relevante, actuale și variate, care au contribuit la documentarea și fundamentarea teoretică și tehnologică a lucrării. Sunt incluse cărți de specialitate, articole științifice, documentații oficiale și resurse web utilizate în procesul de dezvoltare și analiză.

1. Cărți și lucrări de specialitate

1. Freeman, A., *Pro ASP.NET Core MVC 2*, Apress, 2017.
2. Esposito, D., *Programming ASP.NET Core*, Microsoft Press, 2018.
3. Troelsen, A., Japikse, P., *Pro C# 9 with .NET 5*, Apress, 2021.
4. Richter, J., *CLR via C#*, Microsoft Press, 2012.

2. Articole și publicații academice

1. Smith, J. et al., "Online Auction Platforms: Trends and Security Challenges," *Journal of Web Engineering*, vol. 21, no. 3, 2021.
2. Doe, M., "Distributed Payment Systems in e-Commerce," *International Journal of Computer Applications*, vol. 165, 2022.

3. Documentație oficială și standarde

1. Microsoft Docs – ASP.NET Core: <https://learn.microsoft.com/en-us/aspnet/core>
2. PostgreSQL Documentation: <https://www.postgresql.org/docs>
3. Stripe Developer Docs: <https://stripe.com/docs>
4. Entity Framework Core Documentation: <https://learn.microsoft.com/en-us/ef/core/>

4. Resurse web și tutoriale tehnice

1. Dev.to – Articles on ASP.NET and Blazor: <https://dev.to/>
2. Medium – Topics: Software Architecture, Web Development: <https://medium.com>
3. Stack Overflow – Comunitate și soluții tehnice: <https://stackoverflow.com>
4. Railway PostgreSQL Hosting: <https://railway.app>
5. Azure DevOps YAML Pipelines Guide: <https://learn.microsoft.com/en-us/azure/devops/pipelines/yaml-schema>