UNIVERSITATEA DIN CRAIOVA
FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI ELECTRONICĂ

DEPARTAMENTUL DE CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

LUCRARE DE LICENȚĂ

Popescu Constantin-Mădălin

COORDONATOR ȘTIINȚIFIC

Cătălin Cerbulescu

Mai 2025
Craiova

UNIVERSITATEA DIN CRAIOVA
FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI ELECTRONICĂ

DEPARTAMENTUL DE CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

FastRide. Aplicație de ride sharing

Popescu Constantin-Mădălin

COORDONATOR ȘTIINȚIFIC

Cătălin Cerbulescu

Mai 2025
Craiova

Contents

1. INTRODUCERE

1.1 Scopul principal

Fast Ride este o aplicație de ride-sharing, concepută pentru a facilita legătura directă între clienți și șoferi, într-un mod rapid și eficient.

Scopul principal al aplicației Fast Ride este de a oferi un mediu online prin care utilizatorii (clienți) pot cere curse în timp real, iar șoferii disponibili le pot accepta într-un mod simplu și intuitiv. Aplicația urmărește să automatizeze complet procesul de conectare dintre cererea și oferta de transport urban, eliminând nevoia de apeluri telefonice.

1.2 Motivație

Tema aleasă pentru această lucrare pornește de la nevoia pentru accesul rapid și eficient la servicii de transport. Odată cu dezvoltarea orașelor și creșterea traficului, s-a observat o tendință accentuată spre utilizarea aplicațiilor de tip ride-sharing, care oferă o alternativă flexibilă la transportul clasic.

Aplicația Fast Ride propune o soluție simplă și eficientă pentru gestionarea curselor între clienți și șoferi. Ideea de a dezvolta această aplicație a venit din dorința de a construi un proiect complet, în care să se regăsească elemente din tot ce înseamnă dezvoltare software modernă: o interfață prietenoasă, comunicare în timp real, stocare în cloud și integrare cu servicii externe.

Tehnologiile alese – Blazor WebAssembly pentru partea de frontend, Azure Durable Functions pentru backend și SignalR pentru actualizări live – au permis realizarea unei aplicații distribuite, capabile să răspundă în timp real cerințelor utilizatorilor. De asemenea, integrarea cu Stripe pentru validarea cardurilor și autentificarea prin cont Google au contribuit la crearea unei experiențe cât mai fluide și sigure.

Prin aceasta lucrare mi-am propus dezvoltarea unei aplicații funcționale, care poate fi ușor extinsă și adaptată în viitor, dar și învățarea unor tehnologii moderne folosite în proiecte reale.

2. TEHNOLOGII ȘI FRAMEWORK-URI FOLOSITE

2.1 C#

2.2 .NET

2.3 Azure Function

Azure Functions este o platformă serverless dezvoltată de Microsoft care permite rularea codului în cloud fără a fi nevoie să gestionezi infrastructura de servere. Acest model facilitează dezvoltarea rapidă a aplicațiilor și serviciilor scalabile, concentrându-te doar pe logică, nu pe administrarea resurselor.(**azureFunctions**)
Principalele caracteristici ale Azure Functions sunt:

- Execuție event-driven: Funcțiile sunt declanșate automat de evenimente, cum ar fi modificări în baza de date, mesaje din cozi, cereri HTTP, cronometre sau alte surse.
- Scalabilitate automată: Azure gestionează în mod automat scalarea funcțiilor în funcție de cerere, asigurând performanță optimă indiferent de volumul de trafic.
- Model de plată pay-as-you-go: Se plătește doar pentru timpul efectiv în care codul rulează, fără costuri fixe legate de infrastructură.
- Suport pentru mai multe limbaje: C#, JavaScript, Python, Java, PowerShell și altele pot fi folosite pentru a scrie funcțiile.

(**azureFunctions**)

2.3.1 Azure Durable Functions

Azure Durable Functions reprezintă o extensie a platformei Azure Functions, dezvoltată de Microsoft pentru a facilita crearea de fluxuri de lucru pe termen lung și orchestrarea funcțiilor serverless într-un mod eficient și scalabil. Lansată oficial în 2017, această extensie adaugă capabilități suplimentare funcțiilor serverless tradiționale, permițând dezvoltatorilor să gestioneze procese complexe care implică mai multe funcții interdependente și executate în timp.(**azureDurableFunctions**)

Azure Functions, lansat cu un an mai devreme (2016), face parte din suita de servicii serverless computing din Azure și permite rularea de cod fără a fi necesară administrarea explicită a infrastructurii. Totuși, aceste funcții standard au fost concepute pentru execuții rapide, de scurtă durată, ceea ce le făcea mai puțin potrivite pentru procesele care necesită menținerea stării și coordonarea pe termen lung.(**azureDurableFunctions**) Pentru a răspunde acestor nevoi, Durable Functions oferă următoarele capabilități:

- Orchestrarea funcțiilor de scurtă durată, într-un mod automatizat și declarativ;
- Gestionarea stării între apelurile funcțiilor, pe parcursul execuției unui proces complex;

- Retry automat și suport pentru scenarii de compensare în caz de eșec.

(**azureDurableFunctions**) Durable Functions este construit pe baza Durable Task Framework, permițând dezvoltatorilor să scrie cod orchestrat într-un stil secvențial, dar care este transformat automat în execuție asincronă și distribuită, cu păstrarea stării între pași.(**azureDurableFunctions**)
Scenarii comune de utilizare:

- Orchestrarea fluxurilor de lucru O funcție orchestrator coordonează apelurile către alte funcții, în funcție de anumite condiții sau răspunsuri. De exemplu, poate apela o funcție care extrage date de la un API, apoi, în funcție de rezultat, lansează alte funcții în lanț.
- Function Chaining (lanțuri de funcții) Mai multe funcții sunt apelate secvențial, iar rezultatul fiecărei funcții este transmis mai departe către următoarea funcție din lanț.
- Fan-out/Fan-in O funcție poate declanșa mai multe funcții în paralel (fan-out), după care agregă rezultatele într-un punct comun (fan-in). Acest model este ideal pentru procesarea paralelă a unor seturi mari de date.
- Gestionarea proceselor de lungă durată Durable Functions permite execuția de fluxuri care pot dura ore, zile sau chiar luni, menținând starea între pași. Este util în scenarii precum aprobări, procese de onboarding, migrare de date sau procese distribuite în timp.
- Compensarea acțiunilor (Saga Pattern) În fluxuri unde mai multe acțiuni trebuie executate într-o ordine strictă, Durable Functions permite implementarea de logici de rollback sau compensare în caz de eșec, asigurând consistența procesului.
- Funcții temporizate (Timer Functions) Orchestratorii pot programa funcții să ruleze după o întârziere sau la anumite intervale. Acestea sunt utile în procese automate, monitorizări periodice sau trimiterea de notificări programate.

(**azureDurableFunctions**)

2.4 Azure Table Storage

2.4.1 Azurite

2.5 LINQ

2.6 HyperText Markup Language (HTML)

2.7 Cascading Style SHeets (CSS)

2.8 Bootstrap

2.9 Blazor

Blazor este un framework open-source dezvoltat de Microsoft, lansat oficial în 2018, ca parte a ecosistemului .NET. Numele „Blazor” este format din cuvintele „Browser” și „Razor”, evidențiind utilizarea motorului Razor pentru redarea componentelor web direct în browser.(**blazor**)

Scopul principal al Blazor este de a permite dezvoltatorilor .NET să creeze aplicații web interactive fără a apela la JavaScript, oferind o alternativă la framework-uri front-end precum React, Angular sau Vue.js. Utilizând

limbajul C# și întreg ecosistemul .NET, Blazor a devenit rapid o opțiune atractivă pentru dezvoltatorii familiarizați cu tehnologiile Microsoft, facilitând dezvoltarea de aplicații full-stack doar cu .NET.(**blazor**)
Blazor este disponibil în două variante principale:

- Blazor Server (2019): Aplicația rulează pe server, iar interacțiunea cu utilizatorul este gestionată în timp real prin SignalR. Această variantă oferă performanțe ridicate și un consum redus de resurse pe client, dar necesită o conexiune constantă la server.
- Blazor WebAssembly (2020): Codul C# este compilat în WebAssembly și rulează direct în browser, eliminând nevoia unei conexiuni continue la server. Aceasta permite dezvoltarea de aplicații web care pot funcționa și offline.

(**blazor**) Utilizări principale ale Blazor:

- Aplicații web interactive (SPA - Single Page Applications): Blazor permite dezvoltarea de aplicații de tip SPA, în care navigarea și interacțiunile cu utilizatorul se realizează fără reîncărcarea completă a paginii, oferind o experiență fluidă și modernă.
- Aplicații WebAssembly: Cu Blazor WebAssembly, aplicațiile pot rula complet în browser, reducând latențele și oferind posibilitatea de a crea aplicații offline sau cu funcționare locală.
- Aplicații server-side: Blazor Server este ideal pentru aplicații care necesită control sporit asupra datelor și un răspuns în timp real. Prin SignalR, modificările din UI sunt reflectate instant, fără apeluri repetate la server.
- Aplicații enterprise: Datorită integrării excelente cu ecosistemul .NET, Blazor este preferat în mediul enterprise pentru reutilizarea codului existent, integrarea ușoară a logicii de business, autentificării, bazelor de date și serviciilor API.

(**blazor**)

2.10 MudBlazor

2.11 SignalR

WebSockets este un protocol de comunicație care permite o conexiune bidirecțională, persistentă și full-duplex între un client (de exemplu, un browser web) și un server, facilitând transmiterea rapidă și continuă a datelor în timp real fără a reîncarca pagina.(**signalR**)

SignalR este o bibliotecă dezvoltată de Microsoft care facilitează comunicarea în timp real între aplicațiile web, mobile sau desktop și servere. Lansată în 2011 și integrată ulterior în ecosistemul ASP.NET Core, SignalR permite actualizări și notificări instantanee fără a fi nevoie să reîncarci paginile web.(**signalR**)
Înainte de SignalR, comunicarea bidirecțională în timp real era dificilă și adesea implementată prin tehnici ineficiente precum polling sau long-polling, care consumau multe resurse. SignalR a simplificat acest proces prin integrarea automată a protocolului WebSockets, care oferă o conexiune persistentă și eficientă între client și server.(**signalR**)
Cu apariția ASP.NET Core, SignalR a fost reproiectat pentru a fi mai performant și scalabil, suportând diverse metode de transport și oferind o experiență optimă indiferent de mediu.(**signalR**)
SignalR folosește automat cel mai potrivit mecanism de comunicare, în funcție de capabilitățile clientului și ale serverului:

- WebSockets: Protocolul principal, oferind o conexiune rapidă și bidirecțională.

- Server-Sent Events (SSE): Permite serverului să trimită actualizări către client printr-o conexiune HTTP deschisă.
- Long Polling: Metoda de rezervă, în care clientul face cereri repetate pentru a verifica noutățile atunci când celelalte opțiuni nu sunt disponibile.

(**signalR**) SignalR este ideal pentru aplicații care necesită actualizări în timp real, cum ar fi chat-uri, notificări, dashboard-uri live sau colaborare online.

2.11.1 SignalR Server Emulator

2.12 Refit API

2.13 Leaflet și Routing Machine

2.14 Stripe

Stripe este o platformă globală de procesare a plăților, fondată în 2010 de frații Patrick și John Collison. Aceasta a fost creată pentru a permite afacerilor și dezvoltatorilor să accepte plăți online în mod simplu și sigur. Stripe a devenit rapid unul dintre cei mai populari furnizori de soluții de plăți digitale, datorită ușurinței de integrare, suportului pentru diverse metode de plată și disponibilității în multiple țări.(**stripe**)

Stripe a fost fondată într-o perioadă în care comerțul online era în creștere, dar soluțiile de plată disponibile erau adesea greoaie sau complexe de implementat. Frații Collison au observat o oportunitate de a crea o platformă care să facă procesul de integrare a plăților în site-uri web și aplicații mult mai simplu pentru dezvoltatori. Stripe a fost lansată oficial în 2011, cu misiunea de a moderniza plățile online și de a oferi soluții intuitive pentru afaceri de toate dimensiunile.(**stripe**)

3. ELEMENTE SOFTWARE FOLOSITE

3.1 JetBrains Rider IDE

3.2 Microsoft Azure Storage Explorer

Azure Storage Explorer este un instrument grafic, dezvoltat de Microsoft, care permite accesul și gestionarea datelor stocate în Azure Storage, fie că este vorba despre conturi de stocare, containere, fișiere, tabele sau cozi de mesaje. Lansat pentru a simplifica interacțiunea cu serviciile de stocare din Azure, Azure Storage Explorer oferă o interfață ușor de utilizat pentru dezvoltatori și administratori, permițându-le să gestioneze eficient datele și resursele de stocare în cloud.(**azureStorageExplorer**)

Azure Storage Explorer a fost lansat de Microsoft în 2015, ca un instrument destinat dezvoltatorilor și profesioniștilor IT care utilizează serviciile de stocare oferite de Azure. Înainte de apariția acestuia, interacțiunea cu Azure Storage se realiza în principal prin intermediul Azure Portal, folosind interfețe web sau scripturi și API-uri. Pentru a oferi o soluție mai intuitivă și accesibilă pentru gestionarea resurselor de stocare, Microsoft a dezvoltat Azure Storage Explorer, un client desktop disponibil pentru Windows, macOS și Linux.(**azureStorageExplorer**)

Acesta a evoluat de-a lungul timpului, integrându-se cu alte servicii din Azure și oferind funcționalități extinse, cum ar fi gestionarea datelor offline, suport pentru acces la mai multe conturi de stocare și securizarea accesului prin autentificare bazată pe Azure Active Directory AAD.(**azureStorageExplorer**)

3.3 Docker

4. SPECIFICAȚII ȘI REPREZENTAREA APLICAȚIEI

poze doar cu cod

4.1 Arhitectura aplicației

4.1.1 Stocare

ER si relation diagram ce tabele avem si cum populam

4.1.2 Server-side

durable orchestrations si cum functioneaza activitatile si restul

4.1.3 Client-side

blazor wasm si cum merge, cum apeleaza backendul despre map si background serv§ices

4.2 Comunicarea între componente

diagrama de cum comunica aplicatia

4.2.1 Autentificare și autorizare

4.2.2 Pagina admin-ului

5. UTILIZAREA APLICAȚIEI

poze doar cu UI

5.1 Instalarea aplicației

ce este in readme

5.2 Manual de utilizare

User-flows

5.2.1 Experiența utilizatorului

user case flow

5.2.2 Experiența șoferului

driver case flow

5.2.3 Experiența admin-ului

admin case flow

6. CONCLUZII

Lucrarea a avut ca obiectiv dezvoltarea unei aplicații moderne de tip ride-sharing, Fast Ride, care gestionează într-un mod eficient relația dintre clienți și șoferi. Aplicația pune accent pe simplitate, stabilitate și integrare între mai multe tehnologii actuale, reușind să acopere un flux funcțional complet: de la autentificarea utilizatorului până la vizualizarea unei curse finalizate.

Pe parcursul realizării proiectului, s-a urmărit nu doar implementarea funcționalităților de bază, ci și înțelegerea profundă a modului în care diferite componente software pot comunica între ele. Utilizarea Blazor WebAssembly a oferit o experiență fluidă în browser, în timp ce Azure Durable Functions a permis gestionarea logicii din spatele aplicației într-un mod scalabil și bine structurat. Componentele precum SignalR, Leaflet, Stripe și integrarea cu contul Google au completat ecosistemul aplicației, transformând-o într-un produs coerent și pregătit pentru utilizare reală.

Un alt obiectiv atins a fost acela de a dezvolta o aplicație ușor de extins. Arhitectura aleasă permite adăugarea de noi funcționalități, precum sistemul de rating deja integrat sau potențiale îmbunătățiri în zona de comunicare între utilizatori, istoric detaliat al curselor sau notificări avansate.

Pe lângă dezvoltarea tehnică, acest proiect a contribuit semnificativ la consolidarea cunoștințelor privind conceperea aplicațiilor web, interacțiunea cu serviciile cloud, organizarea codului și gestionarea datelor. A reprezentat o ocazie bună de a lucra cu tehnologii moderne într-un scenariu real, apropiat de cerințele industriei.

În concluzie, Fast Ride nu este doar un exercițiu academic, ci un exemplu de proiect funcțional, care poate servi ca punct de plecare pentru dezvoltări viitoare și care reflectă atât efortul tehnic depus, cât și dorința de a construi aplicații utile, stabile și ușor de folosit.

7. Tabel de indexare