

**Facultatea Calculatoare Informatică şi Microelectronică**

**Departamentul Informatică și Ingineria Sistemelor**

**RAPORT**

**la**

**Practica în Producție**

**Tema:** Aplicația FoodDelivery

**A efectuat:**

Calancea Catalin , grupa MI-222

**A verificat:**

Branişte Rodica , asistent universitar

**Chişinău - 2023**

**CUPRINS**

[**1.INTRODUCERE 3**](#_Toc149245300)

[**2.FORMULAREA PROBLEMEI DE CERCETARE 4**](#_Toc149245301)

[**3. DOCUMENTAREA ASUPRA TEMEI 5**](#_Toc149245302)

[**4. ELABORAREA APLICATIEI 6**](#_Toc149245303)

[**4.1 BACKEND ( PYTHON/DJANGO) 7**](#_Toc149245304)

[**4.2 ELABORAREA APLICATIEI IN XCODE ( SWIFT ) 10**](#_Toc149245305)

[**4.2.1. PARTEA CLIENTULUI 13**](#_Toc149245320)

[**4.2.2. PARTEA ȘOFERULUI 17**](#_Toc149245321)

[**CONCLUZIE 20**](#_Toc149245322)

[**BIBLOGRAFIE 21**](#_Toc149245323)

[**ANEXE 22**](#_Toc149245324)

# 1.INTRODUCERE

Într-o lume în continuă mișcare, în care rapiditatea și accesibilitatea sunt aspecte cheie, industria serviciilor de livrare de mâncare a devenit un element vital al vieții cotidiene. Cu toate acestea, cu creșterea exponențială a cererii pentru astfel de servicii, devine tot mai importantă dezvoltarea unor soluții tehnologice inovatoare care să răspundă nevoilor clienților și să optimizeze procesul de comandă și livrare.

Proiectul "FoodDelivery" se naște din această necesitate crescută pentru servicii de livrare de mâncare eficiente și convenabile. Scopul fundamental al acestui proiect este de a crea o aplicație mobilă robustă, dezvoltată pe platforma iOS, care să ofere utilizatorilor o experiență de comandă de mâncare fără efort, conectându-i rapid cu restaurantele preferate și asigurând livrarea lor la ușa casei în cel mai scurt timp posibil. Necesitatea creării aplicației "FoodDelivery" devine evidentă în contextul actual, în care aglomerația din restaurante și programul încărcat al oamenilor fac ca serviciile de livrare să fie o soluție atractivă.

Pentru proiectul "FoodDelivery," este important să se stabilească obiective clare care să ghideze dezvoltarea și implementarea cu succes a aplicației. Iată câteva obiective esențiale pe care necesită atinse:

1. Dezvoltarea unei aplicații mobile iOS :

Obiectivul principal este crearea unei aplicații mobile iOS fiabile, cu o interfață de utilizator prietenoasă și o performanță excelentă. Aplicația trebuie să ofere o experiență de utilizare fluidă și fără probleme.

1. Integrarea cu restaurantelor partenere:

Trebuie să se stabilească parteneriate cu o varietate de restaurante locale și să se dezvolte o platformă de administrare a restaurantelor pentru a permite acestora să își gestioneze meniurile, să primească comenzi și să monitorizeze stocurile în timp real.

1. Sistem de autentificare și securitate:

Aplicația trebuie să ofere un sistem de autentificare sigur pentru utilizatori, inclusiv autentificare cu parole, autentificare socială și opțiuni de recuperare a parolei. Asigurarea securității datelor utilizatorilor este crucială.

1. Comenzi și livrare eficiente:

Obiectivul este de a permite utilizatorilor să plaseze comenzi rapid și să monitorizeze statusul acestora în timp real. De asemenea, trebuie să se dezvolte un sistem eficient de gestionare a livrărilor pentru a asigura livrări la timp și precisă.

1. Plată online și gestionarea tranzacțiilor:

Aplicația trebuie să permită utilizatorilor să plătească online, inclusiv prin carduri de credit/debit sau alte metode de plată digitale. Trebuie să se implementeze o soluție de gestionare a tranzacțiilor sigură și eficientă.

1. Monitorizare și analiză a performanței:

Trebuie să se implementeze un sistem de monitorizare și analiză a performanței aplicației pentru a identifica și remedia problemele în timp real și pentru a face ajustări pentru a îmbunătăți experiența utilizatorilor.

Aceste obiective reprezintă doar o parte din cadrul complex al proiectului "FoodDelivery," dar sunt esențiale pentru a asigura dezvoltarea și implementarea unei aplicații de livrare de mâncare de succes și satisfacerea nevoilor atât ale utilizatorilor, cât și ale restaurantelor partenere.

# 2.FORMULAREA PROBLEMEI DE CERCETARE

În lumea actuală, industria serviciilor de livrare de mâncare cunoaște o creștere semnificativă, alimentată de ritmul agitat al vieții cotidiene și de cerințele tot mai mari ale consumatorilor pentru comoditate și accesibilitate. Cu toate acestea, odată cu expansiunea acestei industrii, apar și provocări semnificative legate de gestionarea eficientă a comenzilor, livrarea rapidă și asigurarea unei experiențe de utilizare plăcute pentru clienți.

În acest context, proiectul "FoodDelivery" se naște ca răspuns la necesitatea crescută pentru servicii de livrare de mâncare eficiente și moderne. Problema de cercetare centrală care ghidează acest proiect poate fi formulată ca o întrebare fundamentală:

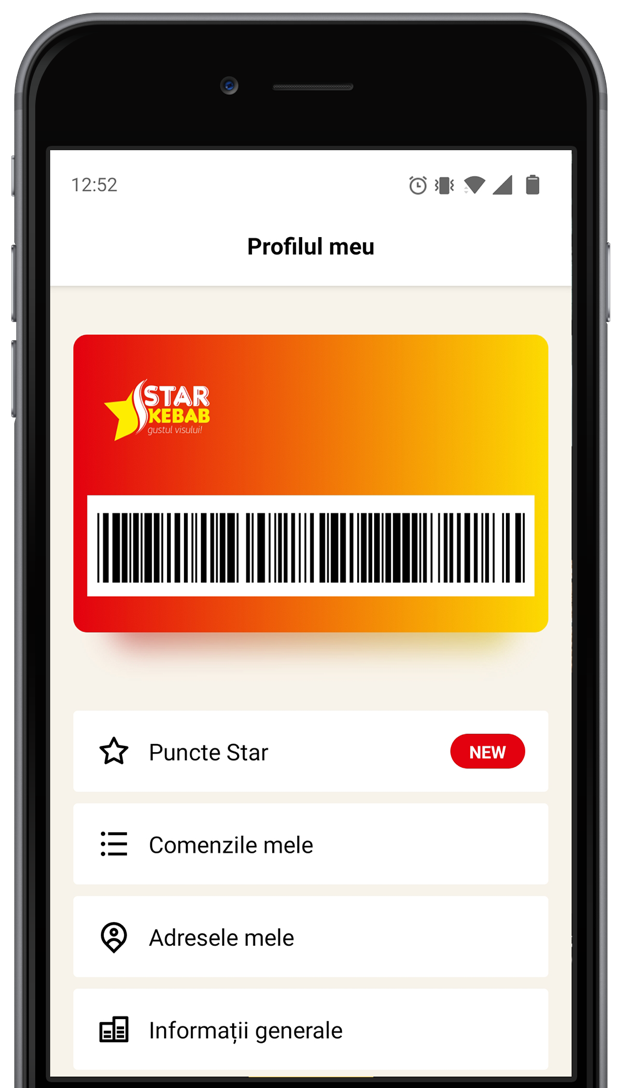
“Cum se poate dezvolta o aplicație mobilă pentru livrarea de mâncare pe platforma iOS, care să îndeplinească cu succes cerințele și așteptările utilizatorilor, să optimizeze procesul de comandă și livrare și să ofere o experiență de utilizare deosebită?”

Această întrebare de cercetare reprezintă o provocare complexă și actuală în dezvoltarea tehnologiei mobile. Ea implică nu doar crearea unei aplicații funcționale, ci și furnizarea unei soluții care să răspundă cu precizie și eficiență nevoilor utilizatorilor și ale restaurantelor partenere.

# 3. DOCUMENTAREA ASUPRA TEMEI

Pentru a dezvolta cu succes aplicația "FoodDelivery," este esențial să efectuăm o documentare exhaustivă asupra temei, pentru a înțelege în profunzime contextul și nevoile din industria serviciilor de livrare de mâncare. Pentru a începe documentarea, este crucial să efectuăm o analiză a pieței actuale și a cererii pentru serviciile de livrare de mâncare. Acest proces implică studiul evoluției industriei de livrare de mâncare, inclusiv creșterea cererii în diferite regiuni și perioade ale anului. Identificarea principalelor aplicații și servicii de livrare de mâncare prezente pe piață, analiza ofertei lor, și înțelegerea cum se poziționează în raport cu nevoile consumatorilor.Investigarea preferințelor și așteptărilor utilizatorilor în ceea ce privește experiența de comandă și livrare, precum și feedback-ul și recenziile disponibile pentru aplicațiile de livrare de mâncare existente.Analiza funcționalităților și caracteristicilor principale ale altor aplicații de livrare de mâncare, identificarea aspectelor care atrag utilizatorii și a celor care pot fi îmbunătățite.

După o analiză atentă a pieței aplicațiilor mobile și a serviciilor de livrare de mâncare, a fost identificată o aplicație specifică numită "StarKebab," care merită o atenție deosebită.



**Figura 3.1.** Aplicatia StarKebab

Una dintre punctele slabe ale aplicației "StarKebab " este faptul că utilizatorii pot comanda doar dintr-un singur local. Aceasta restricționează varietatea de opțiuni disponibile pentru utilizatori și limitează posibilitatea de a alege dintr-un spectru mai larg de restaurante sau pizzerii. Lipsa diversității poate determina utilizatorii să își caute alternative și să nu folosească în mod regulat aplicația.

Un alt punct slab se referă la necesitatea ca fiecare șofer să treacă printr-un control specific. Deși controlul poate fi o măsură de securitate necesară pentru a proteja siguranța și integritatea procesului de livrare, acesta poate fi un proces costisitor și de timp pentru companie. De asemenea, poate duce la întârzieri în recrutarea și angajarea de noi șoferi, ceea ce poate afecta eficiența serviciului de livrare.

În contextul pieței de livrare de mâncare în continuă creștere, există numeroase restaurante, precum "Andy's Pizza," "TorroKebab," și "Beef Kebab," care nu dispun încă de o aplicație de livrare dedicată. Aceasta reprezintă o oportunitate semnificativă pentru a satisface nevoile acestor restaurante și ale clienților lor.Prin dezvoltarea unei aplicații de livrare, restaurantele pot să își extindă prezența online, să își promoveze meniurile și să ajungă la un număr mai mare de clienți, inclusiv la cei care preferă să comande mâncare de acasă.Cu o aplicație de livrare, clienții au acces la o gamă variată de restaurante și meniuri, permițându-le să exploreze și să aleagă diverse opțiuni culinare.Prin intermediul aplicațiilor de livrare, clienții pot să urmărească în timp real statusul comenzilor lor și să primească notificări despre momentul livrării.

În concluzie, dezvoltarea de aplicații de livrare pentru restaurante poate aduce beneficii semnificative atât pentru industria alimentară, cât și pentru consumatori. Este o oportunitate de a satisface cerințele actuale ale pieței și de a contribui la creșterea sectorului de livrare de mâncare.

# 4. ELABORAREA APLICATIEI

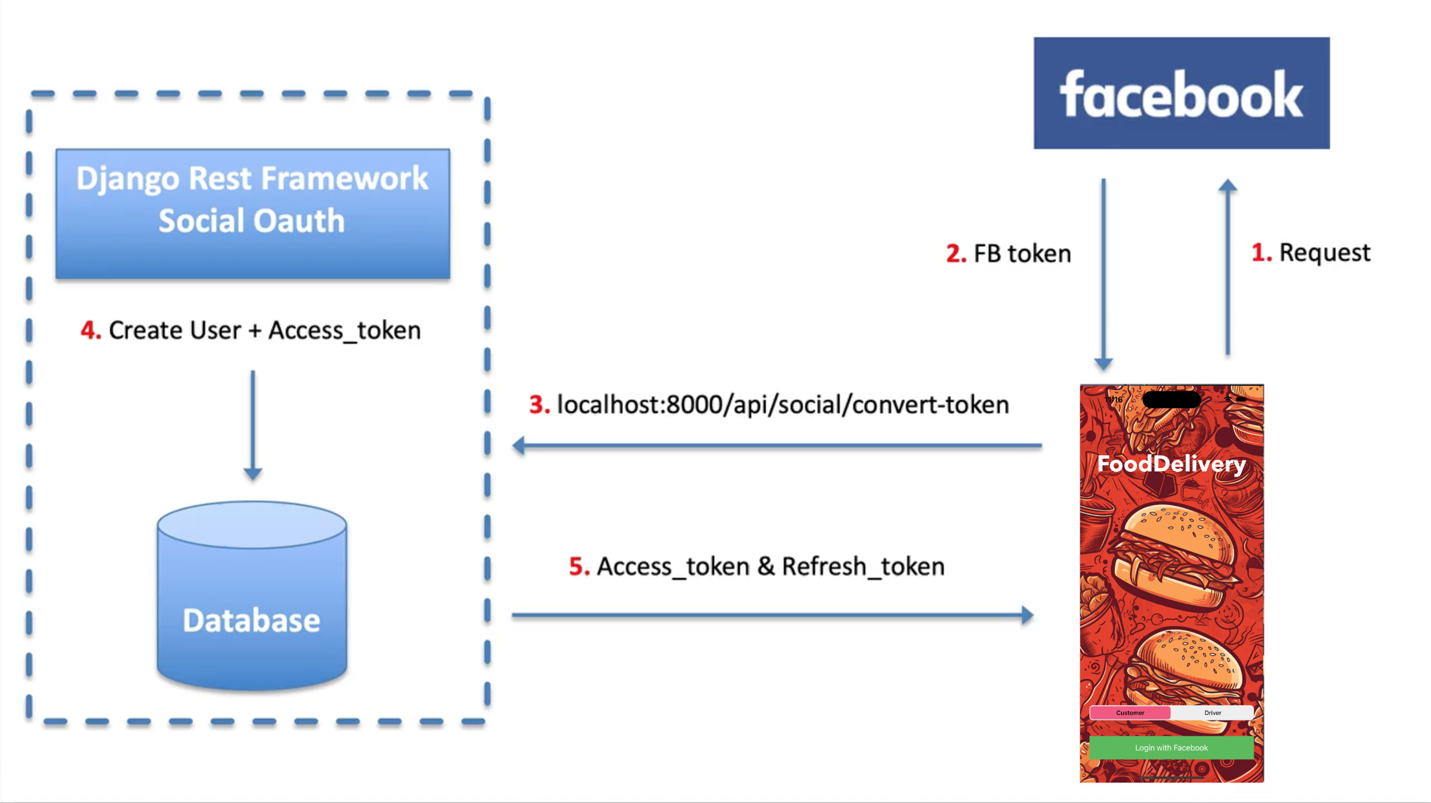
Elaborarea aplicației "FoodDelivery" pe platforma iOS și utilizarea limbajului Swift, în combinație cu Python și framework-ul Django, reprezintă o alegere strategică în concordanță cu analizele și evaluările făcute până acum.

* Alegerea platformei iOS și limbajului Swift:

Platforma iOS are o bază mare de utilizatori loiali și pretențioși, ceea ce oferă o oportunitate semnificativă de a ajunge la un public divers și calitativ, iar Swift, limbajul de programare pentru iOS, este recunoscut pentru simplitatea sa și capacitatea de a crea aplicații cu performanțe excelente. Acest lucru va contribui la dezvoltarea unei aplicații iOS robuste și prietenoase pentru utilizatori.

* Utilizarea Python și Django pentru backend:

Python este un limbaj flexibil și ușor de învățat, ceea ce face dezvoltarea backend-ului mai eficientă. Django, ca framework, oferă un cadru solid pentru gestionarea bazelor de date și a logicii de server, având în vedere necesitatea protejării datelor utilizatorilor și a restaurantelor partenere, alegerea unui limbaj și a unui framework care pun accent pe securitate este esențială.



**Figura 4.1.** Logica de bază a proiectului

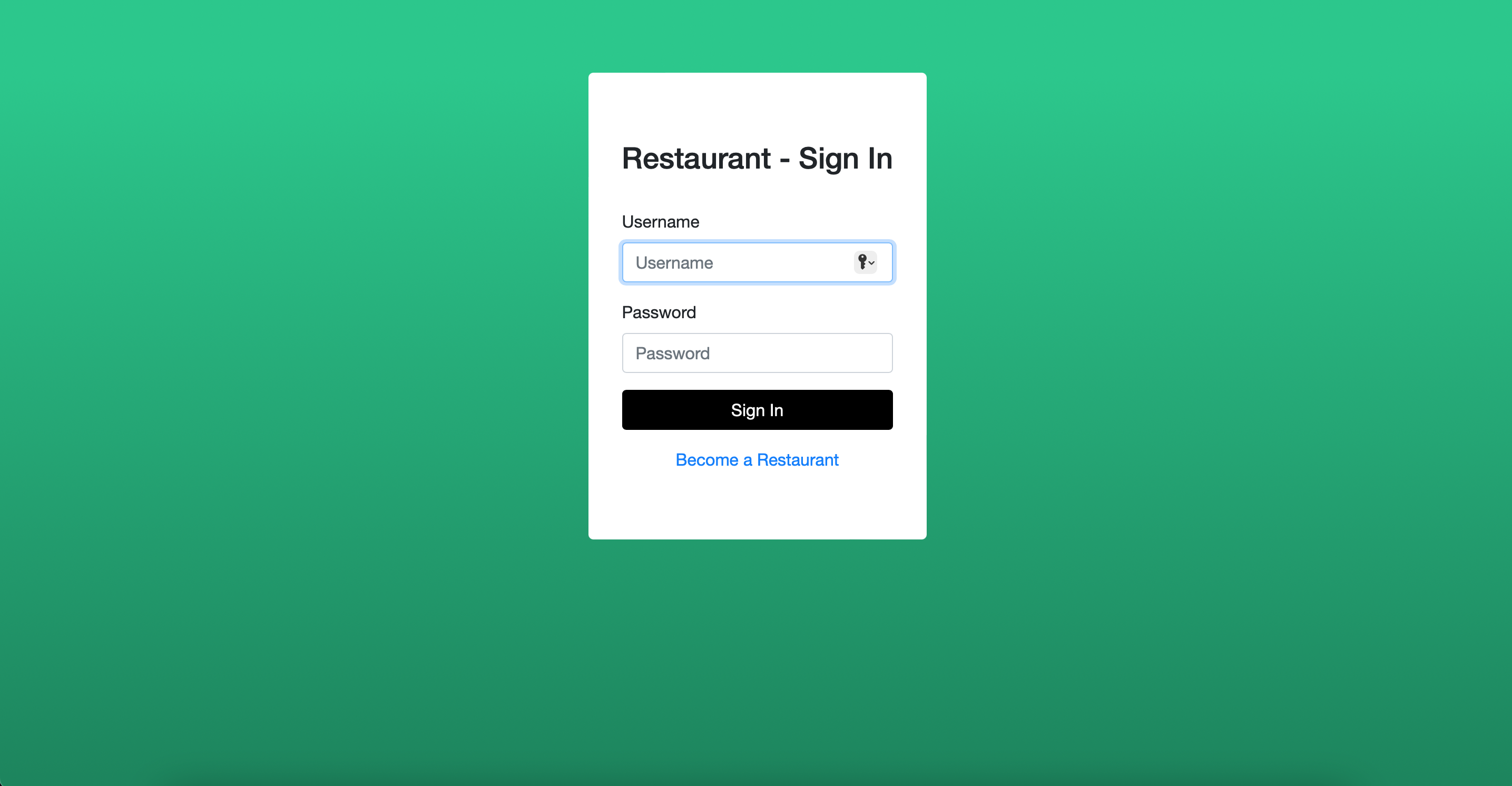
Proiectul "FoodDelivery" se bazează pe identificarea unei oportunități clare în piața serviciilor de livrare de mâncare. Dezvoltarea unei aplicații care să faciliteze livrarea de mâncare pentru restaurantele fără o soluție de livrare proprie este un exemplu de inovare și adaptare la cerințele pieței.

Prin combinarea acestor factori și luând în considerare cele menționate mai sus, "FoodDelivery" își propune să dezvolte o aplicație de livrare de mâncare pe platforma iOS și Python Django care să ofere o experiență excelentă utilizatorilor și să contribuie la satisfacerea cerințelor dinamice ale pieței de livrare de mâncare.

# 4.1 BACKEND ( PYTHON/DJANGO)

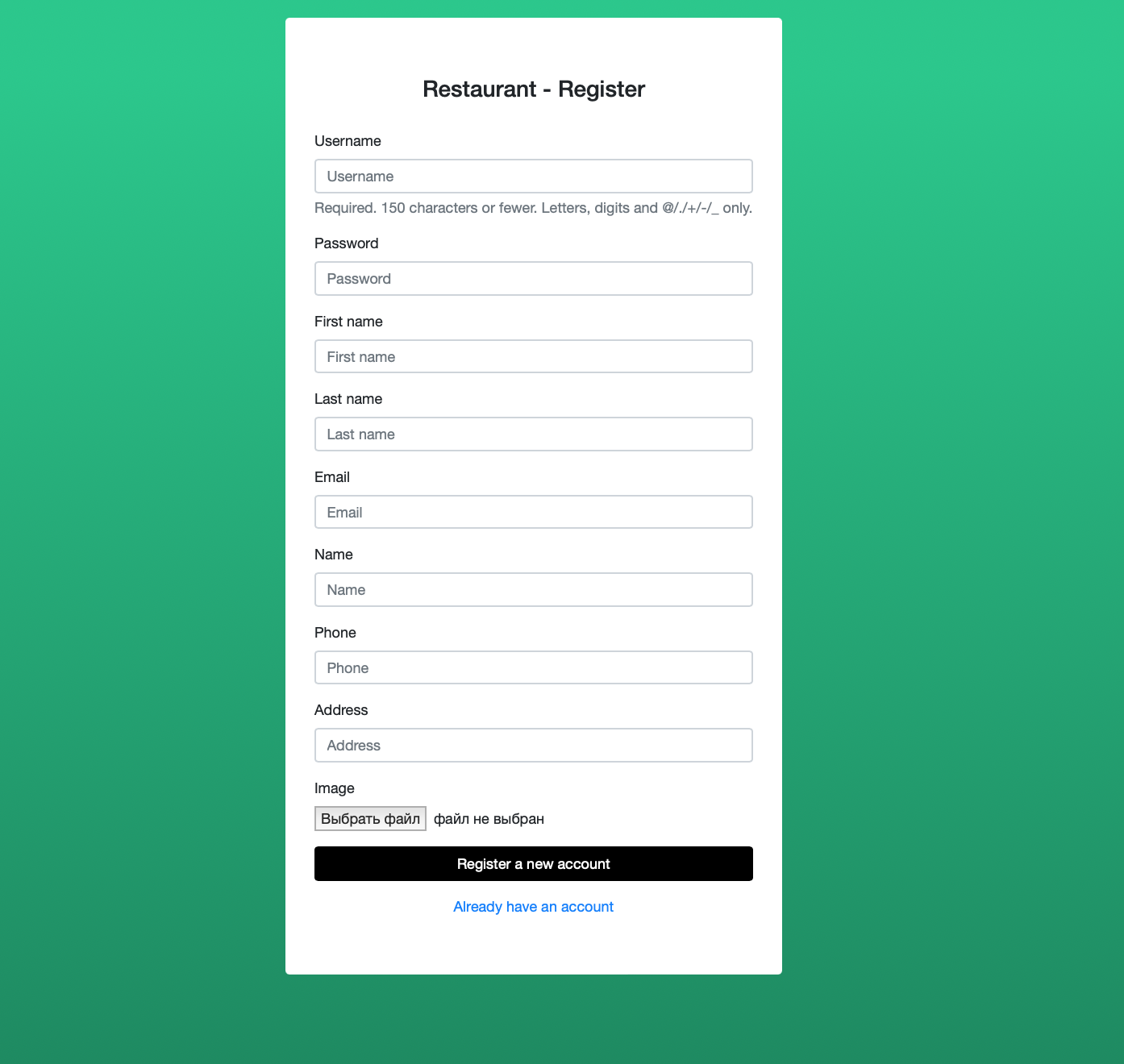
Crearea unei pagini de start (front-end) și a unui panou de control Django pentru proiectul "FoodDelivery" implică dezvoltarea atât a părții vizuale, cât și a celei de back-end.

1. Crearea Proiectului: A fost inițiat un proiect Django numit "FoodDelivery" pentru a dezvolta o aplicație de livrare de mâncare: “django-admin startproject FoodDelivery”.
2. Crearea Paginii de Start: Pagina de start a fost dezvoltată pentru a oferi o primă impresie a aplicației. Aceasta include un design atractiv, structură HTML/CSS și conținut relevant.



**Figura 4.1.1** Pagina de start

1. Integrarea Bootstrap: Framework-ul Bootstrap a fost integrat pentru a facilita stilizarea și crearea unei interfețe de utilizator responsive.
2. Configurarea Panoului de Control Django: A fost creat un panou de control Django dedicat pentru administrarea restaurantelor, meniurilor și comenzilor.
3. Autentificare și Deautentificare (Login / Logout): Funcționalitățile de autentificare și deautentificare au fost implementate pentru utilizatori și proprietarii de restaurante, gestionându-se sesiunile de utilizator cu Django.
4. Încărcarea Imaginilor în Cloud (Cloudinary): A fost integrat serviciul Cloudinary pentru încărcarea imaginilor (de ex. imagini de restaurant sau produse) în aplicație. De asemenea, s-a configurat și utilizat biblioteca Python pentru interacțiunea cu Cloudinary.
5. Pagina de Înregistrare: A fost creată o pagină de înregistrare pentru ca proprietarii de restaurante să poată adăuga restaurantele lor în aplicație.
6. Formularul de Înregistrare: A fost dezvoltat un formular de înregistrare care colectează informațiile necesare pentru crearea unui cont de restaurant.
7. Funcționalitatea de Înregistrare: Funcționalitatea de înregistrare a fost implementată pentru a valida datele introduse și a crea conturile de restaurant în baza de date.
8. A fost dezvoltată o aplicație Facebook dedicată pentru autentificarea și conectarea utilizatorilor prin intermediul conturilor lor Facebook.
9. Configurarea Autentificării cu Facebook: S-a configurat și integrat autentificarea cu Facebook pentru a permite utilizatorilor să se autentifice folosind conturile lor de pe această platformă.
10. Autentificarea pentru Client și Șoferi: S-au dezvoltat funcționalități separate pentru autentificarea clienților și a șoferilor, astfel încât fiecare categorie de utilizatori să aibă acces la funcționalități specifice.
11. Structura Generală a Panoului de Control: S-a stabilit o structură generală pentru panoul de control care permite administratorilor să gestioneze restaurantele, meniurile și comenzile.
12. Design Avansat pentru Pagina de Restaurant: S-au dezvoltat design-uri avansate pentru paginile de prezentare a restaurantelor, oferind informații detaliate și imagini atractive.
13. Design Avansat pentru Paginile de Autentificare: S-au creat design-uri avansate pentru paginile de autentificare, oferind o experiență vizuală plăcută utilizatorilor.
14. Integrarea Bootstrap în Paginile de Autentificare: S-a implementat framework-ul Bootstrap pentru stilizarea paginilor de autentificare, asigurând o prezentare coerentă și atrăgătoare.
15. Formular Bootstrap: S-a dezvoltat un formular Bootstrap care facilitează introducerea datelor și oferă o experiență de utilizare fluidă.
16. Bootstrap pentru Panoul de Control: Framework-ul Bootstrap a fost integrat în panoul de control, facilitând design-ul și interacțiunea utilizatorului.
17. Stilizare Personalizată pentru Panoul de Control: S-a aplicat un stil personalizat pentru panoul de control pentru a reflecta identitatea vizuală a proiectului.
18. Pagina de Cont: S-a dezvoltat o pagină de cont pentru utilizatori, unde aceștia pot gestiona informațiile personale.
19. Pagina Meniurilor: S-a creat o pagină dedicată meniurilor restaurantelor, unde utilizatorii pot explora opțiunile disponibile.
20. Adăugarea Funcționalității de Adăugare a Mâncărurilor: S-a implementat funcționalitatea care permite restaurantelor să adauge noi mâncăruri în meniurile lor.
21. Listarea Meniurilor: S-a dezvoltat o pagină care afișează meniurile restaurantelor disponibile pentru comandă.
22. Pagina de Editare a Mâncării: S-a creat o pagină de editare a detaliilor despre mâncare, unde restaurantele pot actualiza informațiile despre produsele lor.



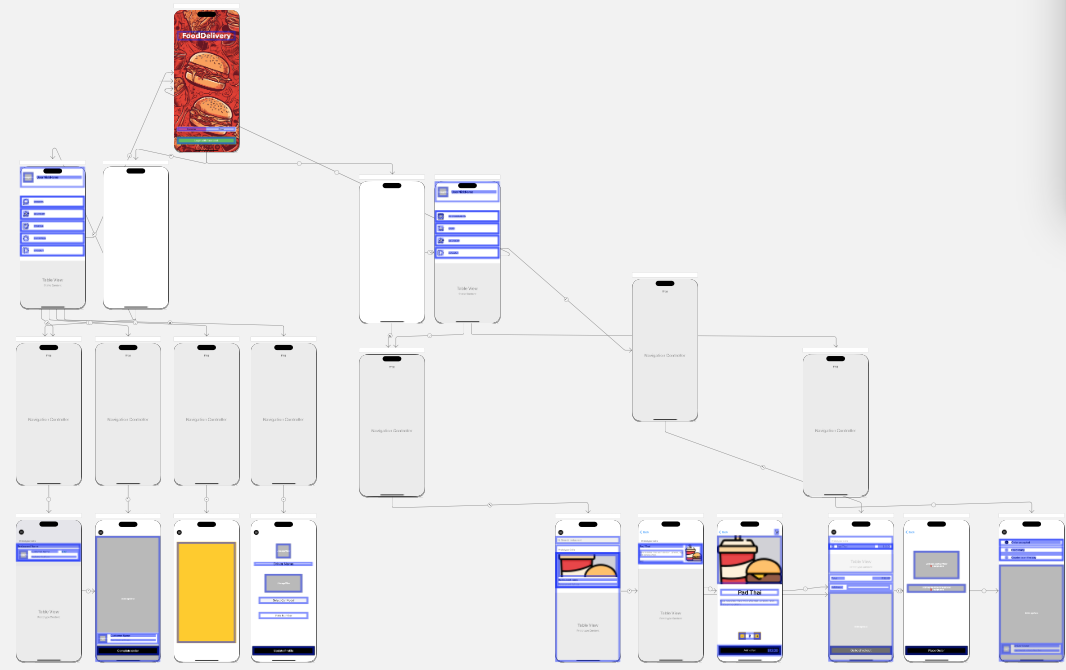
**Figura 4.1.2.** Formularul de înregistrare a restaurantului

Aceste funcționalități și etape ale proiectului "FoodDelivery" reprezintă baza dezvoltării ulterioare a aplicației și a panoului de control. Proiectul se concentrează pe oferirea unei experiențe complete și eficiente pentru utilizatori și administratori.

# 4.2 ELABORAREA APLICATIEI IN XCODE ( SWIFT )

Capitolul anterior a marcat finalizarea dezvoltării platformei web robuste, care permite restaurantelor să-și gestioneze afacerile și clienților să comande cu ușurință mâncarea preferată. Acum, povestea noastră continuă pe drumul dezvoltării mobile.

În acest nou capitol, se dezvăluie o altă fațetă a FoodDelivery, explorând crearea unei aplicații iOS moderne și atractive, destinată atât clienților, cât și restaurantelor partenere. Prin intermediul limbajului Swift, dezvoltatorii se angajează să aducă toate caracteristicile platformei web la vârful degetelor utilizatorilor de dispozitive mobile.



**Figura 4.2.1.** Pagina principala a proiectului “FoodDelivery”

Aplicația “FoodDelivery” este un proiect captivant care folosește o serie de componente și biblioteci pentru a crea o experiență interactivă și utilă pentru utilizatori.

UIView și UIViewController: Aceste două clase sunt fundamentale în dezvoltarea aplicațiilor iOS. UIView este folosit pentru a crea și gestiona elemente de interfață (UI elements) cum ar fi butoane, etichete și imagini, în timp ce UIViewController este utilizat pentru a gestiona logica de afișare și interacțiune a unei anumite ecrane.

MapKit: Aplicația utilizează MapKit, o bibliotecă puternică pentru integrarea hărților și locației în aplicațiile iOS. Cu ajutorul MapKit, se pot afișa hărți interactive, marca puncte de interes și oferi navigare bazată pe locație.

Biblioteci suplimentare:

* “**Stripe**” - Bibliotecă pentru procesarea plăților;
* “**lottie-ios**” - Animații complexe și interactive în aplicație;
* “**FBSDKCoreKit**” - Funcționalități de bază pentru integrarea autentificării și partajării cu conturile Facebook;
* “**Alamofire**” – Bibliotecă pentru efectuarea cererilor HTTP în aplicații iOS, facilitând comunicarea cu serverele web și preluarea de date;
* “**SwiftyJSON**” - Procesarea și manipularea datelor JSON în limbajul Swift;
* “**SkeletonView**” - Experiență mai bună a utilizatorului în timpul încărcării;
* “**DropDown**” - Simplifică crearea și gestionarea meniurilor derulante (dropdown menus);
* “**Charts**” - Bibliotecă pentru crearea de grafice și diagrame interactive.

În contextul proiectului FoodDelivery, conceptele de sync world și async world pot fi aplicate în diferite părți ale aplicației pentru a îmbunătăți performanța și experiența utilizatorului.

# Sync World în FoodDelivery: Partea de interfață a aplicației, cum ar fi afișarea meniurilor restaurantelor sau procesul de comandă, poate folosi o abordare sincronă. Acest lucru asigură o experiență fluidă pentru utilizatori, deoarece operațiunile sunt secvențiale și previzibile.

# Unele operațiuni de procesare a datelor, cum ar fi calcularea prețului total al unei comenzi, pot fi efectuate sincron. Acest lucru poate simplifica logica și gestionarea fluxului de date.

# Async World în FoodDelivery:

# parte importantă a aplicației FoodDelivery este interacțiunea cu serverele pentru a obține informații despre restaurante, meniuri sau starea comenzilor. Utilizarea cererilor de rețea asincrone permite ca aplicația să rămână receptivă în timpul așteptării răspunsurilor de la server.

# Evenimentele precum actualizările de starea comenzilor sau notificările de la driveri pot fi gestionate asincron pentru a nu bloca interfața utilizatorului. Acest lucru asigură faptul că utilizatorii pot continua să interacționeze cu aplicația în timp ce așteaptă evenimente sau actualizări.

# În cazul unor operațiuni lungi sau costisitoare, cum ar fi procesarea plăților sau urmărirea driverilor în timp real, abordarea asincronă poate îmbunătăți semnificativ performanța și reactivitatea aplicației.

# În general, FoodDelivery poate beneficia de o combinație inteligentă a lumii sincrone și asincrone pentru a oferi o experiență de utilizare plăcută și pentru a gestiona eficient operațiunile de fundal. Astfel, utilizatorii pot comanda mâncare, interacționa cu aplicația și urmări procesul de livrare în timp real fără a întâmpina întârzieri sau blocări nedorite.

# Funcția `requestServer` este folosită pentru a efectua cereri către server folosind librăria Alamofire. Această funcție primește câteva argumente importante:

# `method`: Metoda HTTP pentru cerere, precum GET, POST, PUT sau DELETE.

# `path`: Calea către resursa API către care se face cererea.

# `params`: Parametrii pentru cererea API, sub formă de dicționar de cheie-valoare.

# `encoding`: Tipul de codificare folosit pentru parametrii cererii, cum ar fi JSONEncoding.default sau URLEncoding.default.

* **`completionHandler`:** Handlerul de completare este o închidere (closure) care va fi apelată atunci când cererea către server se încheie.Aceasta permite aplicației să reacționeze la răspunsurile primite și să efectueze acțiuni corespunzătoare, cum ar fi actualizarea interfeței utilizatorului sau prelucrarea datelor primite.

# Linia `let url = baseURL?.appendingPathComponent(path)` este folosită pentru a construi URL-ul complet al cererii API prin combinarea URL-ului de bază (`baseURL`) cu calea specificată (`path`).

# După ce URL-ul este construit, funcția continuă să efectueze cererea folosind Alamofire pentru a comunica cu serverul și să trateze răspunsul primit, pe care îl transformă într-un obiect JSON. Această funcție este esențială pentru a face cereri la server și a gestiona răspunsurile primite.

# 4.2.1. PARTEA CLIENTULUI

**LoginViewController** - Ecranul de autentificare al aplicației care se ocupă de gestionarea autentificării utilizatorilor cu Facebook și redirecționarea lor către ecranul corect în funcție de tipul utilizatorului (client sau șofer).

* Ecranul are un buton de autentificare cu Facebook care utilizează biblioteca `FBSDKLoginKit` pentru gestionarea autentificării. Dacă utilizatorul este deja autentificat, butonul va fi etichetat cu "Continue as [Nume utilizator]".
* Experiența de autentificare se face în două etape:

1. Dacă utilizatorul este deja autentificat cu Facebook, se face o cerere API către serverul FoodDelivery pentru a efectua autentificarea și redirecționarea către ecranul corespunzător.

2. Dacă utilizatorul nu este autentificat cu Facebook, se va deschide fereastra de autentificare Facebook (`FBManager.shared.logIn`) pentru a obține permisiunile necesare și pentru a obține datele utilizatorului. Apoi se va efectua cererea API pentru autentificarea pe server.

* Există și un segment de comutare (`UISegmentedControl`) pentru a permite utilizatorilor să selecteze între tipurile de utilizatori (client sau șofer).



**Figura 4.2.1.1.** Ecranul de autentificare

**CustomerMenuTVC** - Meniul pentru clienți în aplicația FoodDelivery.

* + Există două elemente de interfață utilizate în acest `CustomerMenuTVC`:

1. „imageAvatar”: o imagine pentru avatarul utilizatorului.

2. „labelName”: un câmp pentru afișarea numelui utilizatorului.

* + Există și o funcție care se ocupă de gestionarea deconectării utilizatorului. Dacă utilizatorul dorește să se deconecteze, apelul API „**logout**” este efectuat pentru a șterge sesiunea de pe server. După deconectare, utilizatorul este redirecționat către ecranul de autentificare („**LoginViewController**”).
  + De asemenea sunt reprezentate icon-urile pentru ecranul „**RestaurantViewController**” , „**CartViewController**”, „**DeliveryViewController**”.

**RestaurantViewController** - Afișează lista de restaurante și permite utilizatorilor să caute restaurantele dorite.

* Se utilizează biblioteca `**SkeletonView**` pentru a afișa o animație de încărcare a restaurantelor.
* De asemenea este creată o functie ”**fetchRestaurants**()”, efectuează o cerere API pentru a obține lista de restaurante și se încarcă în tabel după obținerea datelor.
* În extensia `**UISearchBarDelegate**`, este gestionată căutarea și filtrarea restaurantelor.

**MealTableViewController** - Este un ecran , care permite utilizatorilor să vizualizeze și să selecteze poziții disponibile într-un restaurant specific. De asemenea, oferă o modalitate rapidă de a accesa coșul de cumpărături pentru a gestiona comenzile.

* Funcția `**fetchMeals**()` este folosită pentru a efectua o cerere API pentru a obține lista de mese pentru restaurantul specificat.
* Este implementat un "floating button" pentru a permite utilizatorilor să acceseze coșul de cumpărături (`**View Cart**`) și să vadă câte produse au în coș.

**MealDetailsViewController** - Ecranul de afișare a detaliilor unei poziții dintr-un restaurant.

* Funcția `**fetchMeal**()` afișează informațiile despre masa curentă, inclusiv imaginea, numele, descrierea scurtă și prețul.
* Funcția `**goBack**()` este utilizată pentru a reveni înapoi la ecranul anterior.
* Este implementată o funcționalitate de "**badge**" care afișează numărul de produse din coșul de cumpărături pe butonul de coș.
* Funcția `**decreaseQty**(\_:)` și `**increaseQty**(\_:)` permit utilizatorului să crească sau să descrească cantitatea de mese.
* Funcția `**addToCart**(\_:)` este apelată atunci când utilizatorul adaugă o poziție în coșul de cumpărături. Aceasta gestionează diferite scenarii, cum ar fi adăugarea de mai multe poziții identice în coș sau curățarea coșului și adăugarea poziției actuale atunci când utilizatorul comandă mese din restaurante diferite.

**CartViewController** - Acest ecran permite utilizatorului să revizuiască și să finalizeze comanda, precum și să selecteze adresa de livrare pe hartă.

* Funcția `**goToCheckout()**` este apelată atunci când utilizatorul dorește să efectueze comanda și să treacă la ecranul de finalizare a comenzii. Aceasta verifică dacă adresa de livrare este completată și afișează un mesaj de eroare dacă nu este.
* Existența unor elemente din UI, cum ar fi tabela și butonul de finalizare a comenzii, este controlată în funcție de conținutul coșului de cumpărături. Dacă coșul este gol, se afișează un mesaj de avertizare.
* Este implementată funcționalitatea de căutare a adresei, care folosește serviciul de geocodificare pentru a găsi coordonatele adresei introduse de utilizator și pentru a afișa acea locație pe hartă.
* Coordonatele găsite sunt folosite pentru a crea un pin pe hartă care indică locația adresei de livrare.

**PaymentViewController** – Ecranul responsabil pentru procesul de plată în aplicație care gestionează procesul de plată, confirmă plățile cu Stripe și creează comenzi noi atunci când plățile sunt finalizate cu succes

* Variabila `**cardTextField**` este un câmp de text specializat pentru introducerea datelor cardului de credit, furnizat de Stripe.
* Animația Lottie este afișată pentru a oferi feedback vizual utilizatorului în timp ce se așteaptă la procesarea plății.
* Funcția `**startCheckout**()` este responsabilă pentru inițierea procesului de plată. Ea creează un intent de plată folosind API-ul FoodDelivery și primește un `**client\_secret**` pentru intentul de plată, care este necesar pentru a finaliza plata.
* Funcția `**placeOrder()**` este apelată atunci când utilizatorul dorește să plaseze comanda. Aceasta verifică dacă utilizatorul are deja o comandă în curs sau dacă restaurantul este gol. Dacă totul este în regulă, se inițiază procesul de plată și se creează o comandă nouă.
* Protocolul `**STPAuthenticationContext**` este implementat pentru a specifica controlul de autentificare utilizat pentru confirmarea plății cu Stripe.

**DeliveryViewController** – Ecranul ce asigură monitorizarea și afișarea informațiilor despre starea comenzii și locația șoferului pentru o experiență de livrare în timp real

* Se utilizează biblioteca **MapKit** pentru a afișa harta și a calcula rutele dintre diferite locații.
* Funcția `**getLatestOrder**` este responsabilă pentru obținerea celei mai recente comenzi și a detaliilor acesteia. Dacă comanda este în stadiul "Pe drum" (On the way), se obțin informații despre șofer și se inițiază urmărirea locației acestuia.
* Funcția `**getLatestOrderStatus**` este apelată periodic pentru a verifica starea comenzii și pentru a actualiza starea vizuală a ecranului în funcție de stadiul comenzii (cum ar fi "Gata" sau "Pe drum").
* Funcția `**getDriverLocation**` este apelată periodic pentru a obține locația șoferului și pentru a actualiza pin-ul reprezentând șoferul pe hartă.
* Funcția `**showEmptyView**` este apelată atunci când nu există comenzi în curs sau comanda este finalizată. Afișează un mesaj corespunzător pe ecran.
* Funcțiile `**getLocation**` și `**getDirection**` sunt responsabile pentru obținerea și afișarea direcțiilor pe hartă între diferite locații, cum ar fi restaurantul și adresa clientului.
* Funcția `**showRoute**` afișează ruta dintre locații pe hartă și asigură că harta este centrată și afișată corect.
* Funcția `**mapView**()` personalizează aspectul pin-urilor de pe hartă în funcție de tipul lor (restaurant, client sau șofer).

În cadrul aplicației FoodDelivery aceste ecrane lucrează împreună pentru a oferi o experiență completă de comandă și livrare de mâncare , aceste ecrane creează o experiență fluidă și captivantă pentru utilizatori, permițându-le să navigheze prin restaurante, să comande mâncare, să efectueze plăți și să urmărească livrarea comenzii lor în timp real. Această arhitectură modulară și integrarea cu serviciul de plată facilitează procesul de comandă și livrare de mâncare pentru utilizatori în cadrul aplicației FoodDelivery.

# 4.2.2. PARTEA ȘOFERULUI

**DriverMenuTVC** - Ecranul pentru șofer, unde acesta poate vedea informații despre contul său și imaginea de profil

* Există două obiecte de interfață utilizate în acest ecran: `**imageAvatar**` și `**labelName**`. `**imageAvatar**` este o imagine care arată avatarul șoferului, iar `**labelName**` este o etichetă de text care afișează numele șoferului.
* În funcția `**viewDidLoad**`, imaginii `**imageAvatar**` i se atribuie o imagine dintr-o URL, probabil imaginea de profil a șoferului. De asemenea, numele șoferului este afișat în eticheta `**labelName**`. Se aplică o serie de proprietăți de aspect pentru a forma imaginea avatarului.

“**DriverProfileViewController**” – Ecran destinat șoferilor pentru a-și gestiona și actualiza profilurile lor, inclusiv detaliile despre mașină ( model , numere de înmatriculare).

* Există o serie de elemente de interfață, inclusiv o imagine pentru avatarul șoferului (`**imageAvatar**`), un etichetă pentru numele șoferului (`**labelDriverName**`), o imagine pentru mașină (`**imageCar**`), un buton pentru a deschide lista cu modele de mașini (`**buttonDropDown**`), un câmp text pentru numărul de înmatriculare al mașinii (`**textFieldPlateNumber**`) și un buton pentru a actualiza profilul șoferului (`**buttonUpdateProfile**`).
* În funcția `**viewDidLoad**`, se inițializează meniul cu modele de mașini (`**dropDown**`) și se încarcă profilul șoferului, inclusiv numele și imaginea de avatar.
* La apăsarea butonului "**Select Car Model**", se afișează o listă derulantă cu modele de mașini disponibile pentru alegere. Alegerea unui model de mașină actualizează imaginea mașinii afișate.
* În funcția `**updateProfile**`, se colectează informațiile introduse, cum ar fi modelul de mașină și numărul de înmatriculare al mașinii, și se trimit către server pentru actualizarea profilului șoferului. După actualizare, se afișează o alertă de notificare pentru a confirma actualizarea profilului.

**DriverOrdersTVC** - Ecran destinat șoferilor pentru a vizualiza și prelua comenzile disponibile pentru livrare. Acesta oferă o interfață pentru gestionarea comenzilor și comunicarea cu serverul pentru preluarea acestora

* În funcția `**viewDidLoad**`, se inițializează meniul lateral și se încarcă comenzile pregătite (ready orders) pentru șofer.
* Funcția `**loadReadyOrders**` utilizează API-ul pentru a obține comenzile disponibile pentru șofer și le încarcă în vectorul `orders`.
* Funcția `**pickOrder**` este folosită pentru a prelua o comandă specifică. Aceasta trimite o solicitare către API pentru a prelua comanda și afișează o alertă în funcție de rezultatul solicitării.
* În implementarea `**tableView(\_:numberOfRowsInSection:)**`, se returnează numărul de rânduri din tabel, care este egal cu numărul de comenzi din vectorul `orders`.
* În implementarea `**tableView(\_:cellForRowAt:)**`, se complectează celulele tabelului cu informații despre comenzile disponibile. Aceasta include numele restaurantului, numele și adresa clientului, și suma totală a comenzii. De asemenea, se încarcă și imaginea de avatar a clientului.
* Atunci când utilizatorul selectează o comandă din tabel **(`tableView(\_:didSelectRowAt:)`)**, se utilizează funcția `**pickOrder**` pentru a prelua comanda și a afișa o alertă corespunzătoare.

**DriverDeliveryViewController** - Acest ecran oferă șoferilor toate informațiile și instrumentele necesare pentru a efectua cu succes o livrare, inclusiv detalii despre comandă, harta cu direcții și opțiunea de a finaliza livrarea.

* Elementele de interfață, cum ar fi `**imageCustomerAvatar**`, `**labelCustomerName**`, `**labelCustomerAddress**`, `**map**`, și `**buttonComplete**`, sunt utilizate pentru a afișa informații despre comanda curentă și pentru a permite șoferului să finalizeze livrarea.
* Clasa utilizează `**CLLocationManager**` pentru a obține locația actuală a șoferului și afișează această locație pe hartă. De asemenea, actualizează în mod regulat locația șoferului pe server pentru urmărirea în timp real.
* Funcția `**loadData**` este utilizată pentru a obține detaliile comenzii curente de la server, inclusiv informații despre client și adresele restaurantului și ale clientului.
* Funcția `**showEmptyView**` este apelată când nu există nicio comandă de livrat și ascunde toate elementele de interfață relevante, afișând un mesaj corespunzător.
* Funcția `**updateLocation**` actualizează în mod regulat locația șoferului pe server.
* Funcția `**completeOrder**` este utilizată pentru a finaliza comanda curentă și afișează o alertă pentru confirmarea acestei acțiuni.
* Controlerul utilizează protocolul `**CLLocationManagerDelegate**` și protocolul `**MKMapViewDelegate**` pentru a gestiona actualizările locației și afișarea direcțiilor și rutei pe hartă.

**DriverEarningsViewController** - acest ecran oferă șoferilor o vedere vizuală a câștigurilor lor în funcție de zilele săptămânii, permițându-le să urmărească și să evalueze mai ușor performanța lor financiară.

* Clasa utilizează biblioteca `**Charts**` pentru a afișa un grafic de bare care va reprezenta câștigurile șoferului în funcție de zilele săptămânii.
* Funcția `**initializeChart**` este utilizată pentru a inițializa configurația graficului de bare. Ea stabilește aspectul și proprietățile graficului, cum ar fi animația, limitele axelor, formatarea etichetelor și altele.
* Funcția `**loadDataToChart**` este utilizată pentru a încărca datele câștigurilor șoferului în grafic. Ea face o solicitare la server pentru a obține datele și le prelucrează apoi pentru a le adăuga în graficul de bare.
* Datele sunt obținute în funcția `**getDriverRevenue**`, iar apoi sunt adăugate într-un obiect `**BarChartDataEntry**`, care reprezintă punctele de pe grafic. Aceste date sunt organizate în funcție de zilele săptămânii.
* Clasa configurează graficul pentru a afișa zilele săptămânii pe axa x și câștigurile pe axa y. De asemenea, stabilește culorile barelor și afișează datele în grafic.

Ecranele destinate șoferilor în această aplicație reprezintă o suită bine dezvoltată de funcționalități care facilitează gestionarea și efectuarea livrărilor de comenzi. Aceste ecrane oferă o experiență intuitivă și eficientă pentru șoferi, permițându-le să îndeplinească cu succes comenzile de livrare.

# CONCLUZIE

Proiectul FoodDelivery este o aplicație complexă și cuprinzătoare, care îmbină funcționalități pentru clienți și șoferi de livrare pentru a facilita comandarea și livrarea de mâncare. Iată o concluzie generală pentru acest proiect:

FoodDelivery oferă o soluție comprehensivă pentru industria serviciilor de livrare de mâncare, reunind clienții care doresc să comande mâncare de la restaurantele locale și șoferii de livrare care asigură transportul sigur și rapid al comenzilor. Cu o interfață prietenoasă pentru utilizatori, atât pe platformele web cât și pe dispozitive mobile.

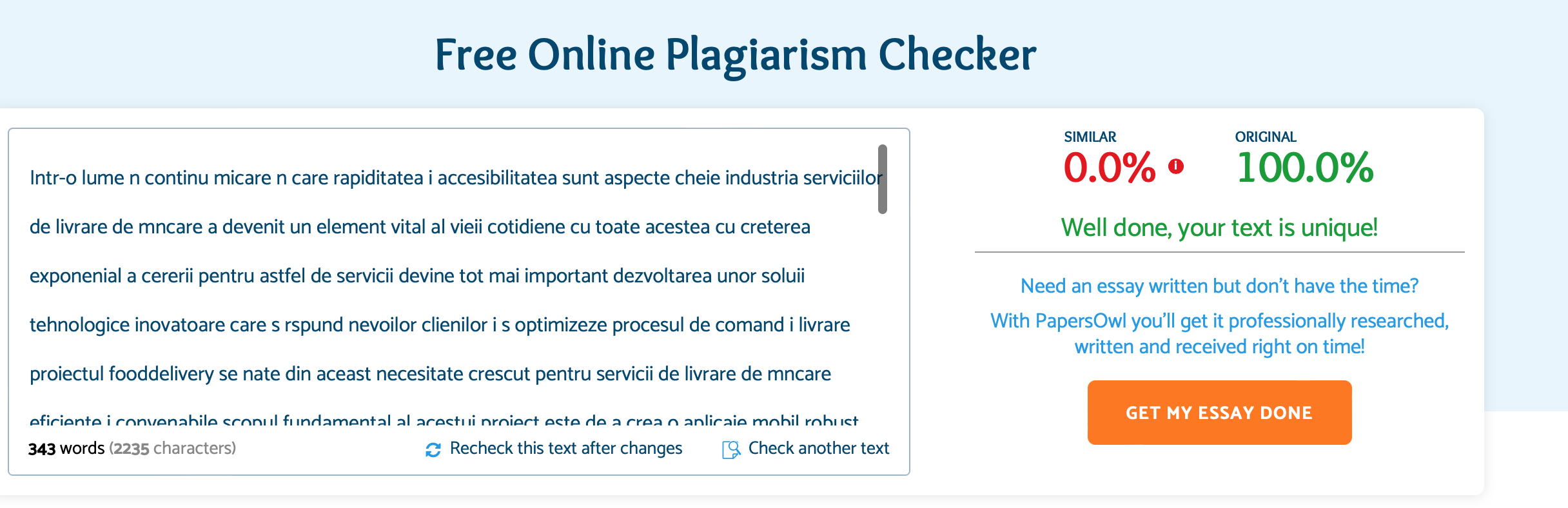
În ansamblu, FoodDelivery demonstrează abilități tehnice avansate în dezvoltarea aplicațiilor mobile și web, în gestionarea bazelor de date, în securitatea tranzacțiilor online și în crearea unui mediu interactiv și funcțional pentru utilizatori. Este o soluție versatilă pentru industria de livrare de mâncare și are potențialul de a revoluționa modul în care oamenii comandă și livrează mâncare în viitor.

# BIBLOGRAFIE

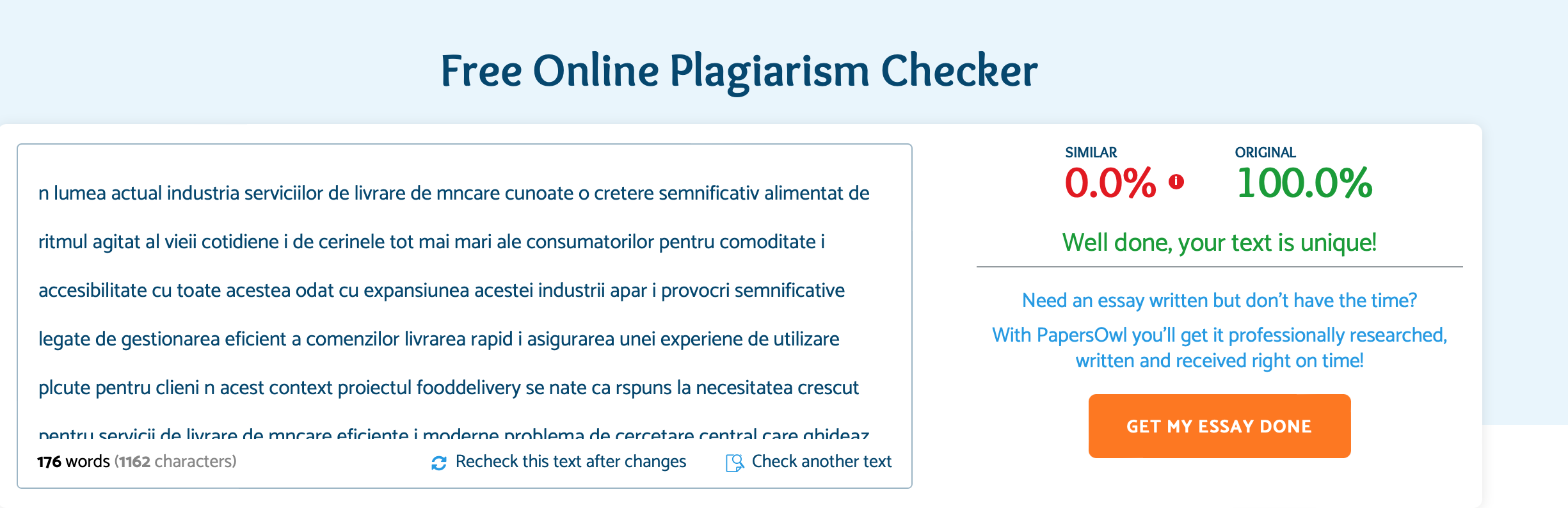
1. Swift Programming Language Documentation. Disponibil la: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/ (accesat pe 04.09.2023 – 29.09.2023)
2. Facebook for Developers Documentation. Disponibil la: https://developers.facebook.com/docs/ (accesat pe 14.09.2023).
3. Stripe API Documentation - Create Payment Intents. Disponibil la:https://stripe.com/docs/api/payment\_intents/create (accesat pe 21.09.2023).
4. SkeletonView GitHub Repository. Disponibil la: https://github.com/Juanpe/SkeletonView#-miscellaneous (accesat pe 20.09.2023).
5. Cloudinary Documentation. Disponibil la: [https://cloudinary.com/documentation (accesat pe 06.09.2023).

# ANEXE

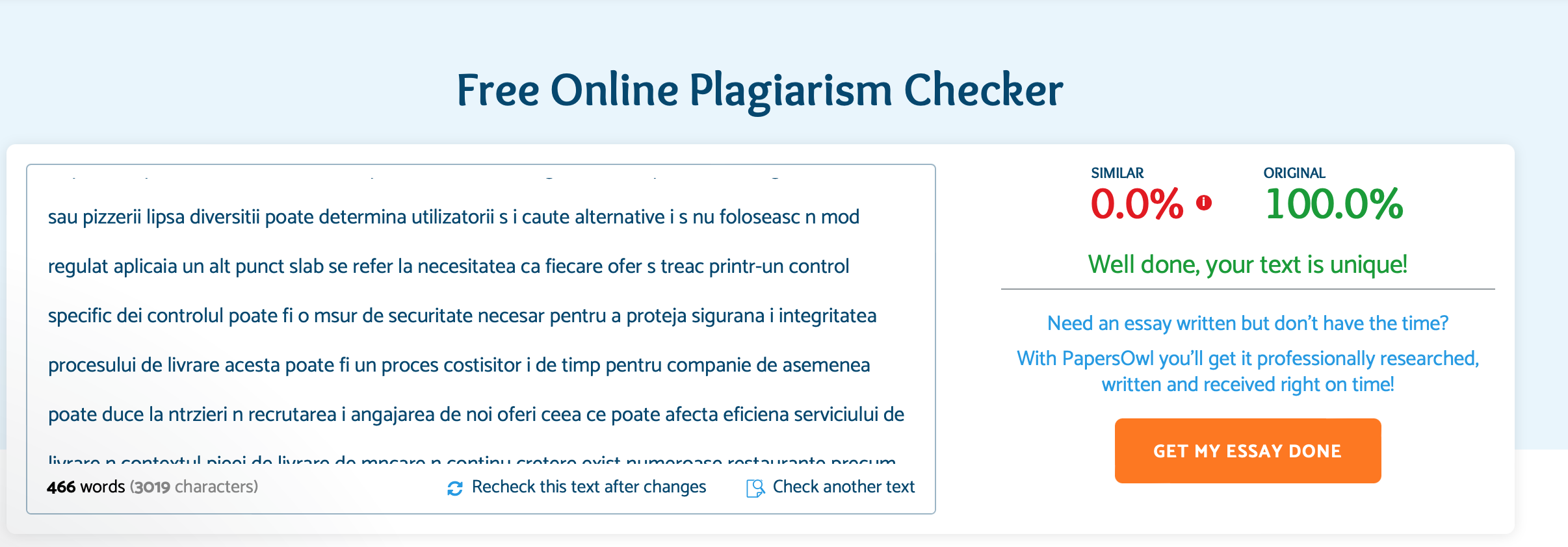
Verificarea antiplagiatului: Capitolul 1



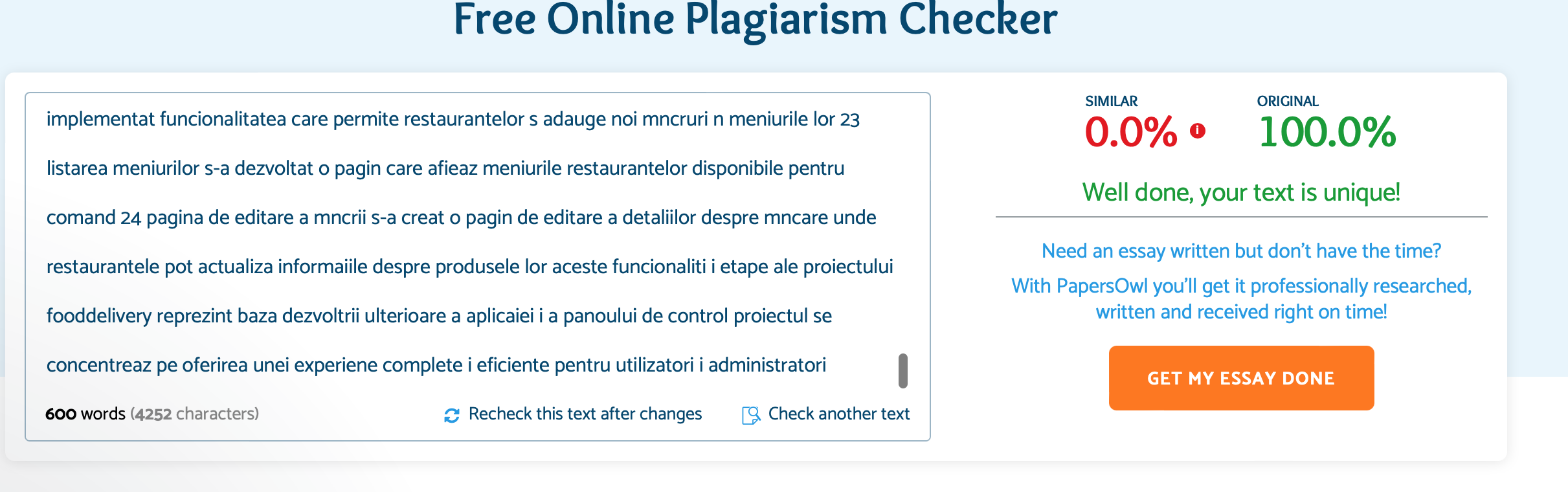
Verificarea antiplagiatului: Capitolul 2



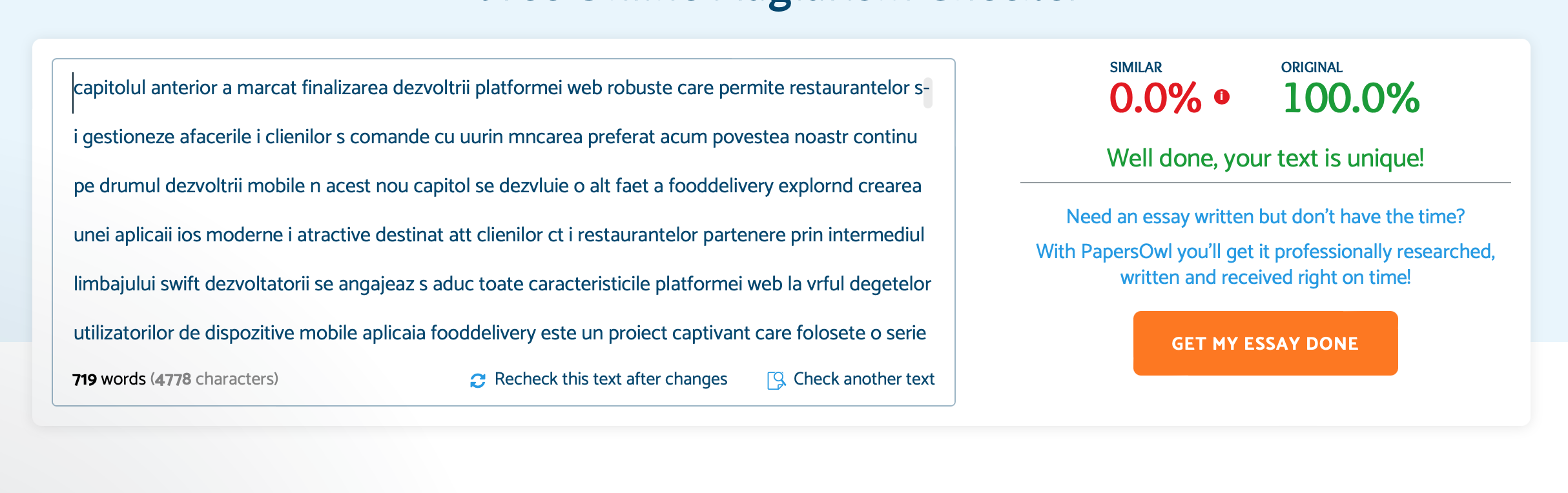
Verificarea antiplagiatului: Capitolul 3



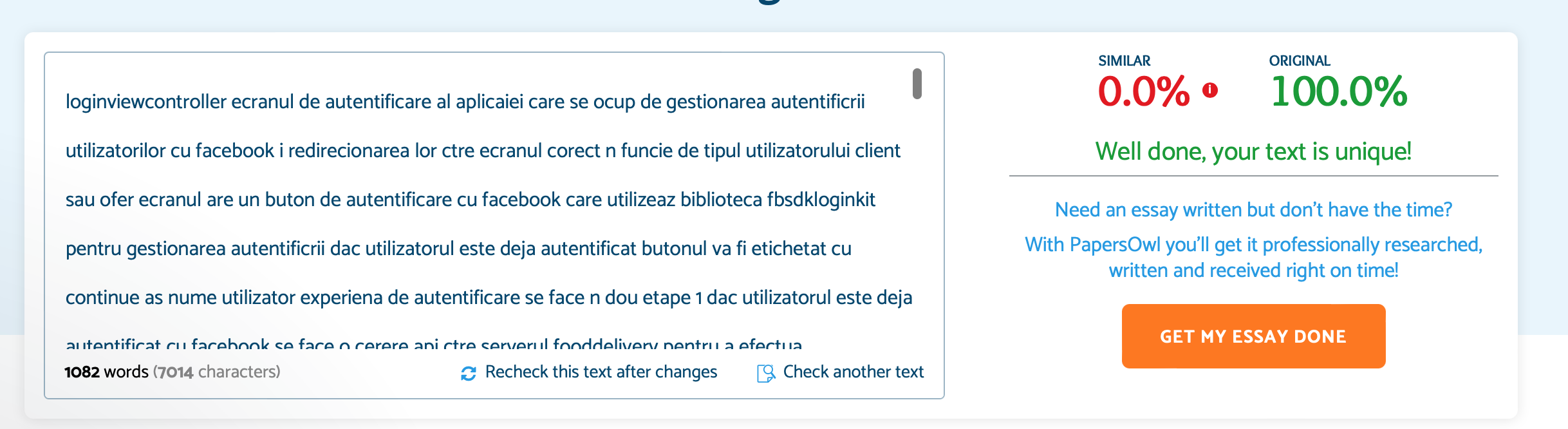
Verificarea antiplagiatului: Subcapitolul 4.1



Verificarea antiplagiatului: Subcapitolul 4.2



Verificarea antiplagiatului: Subcapitolul 4.2.1



Verificarea antiplagiatului: Subcapitolul 4.2.2

