# Aplicație pentru fidelizarea clienților unei cafenele

**Mihai-Cezar Viziteu**

**ACADEMIA DE STUDII ECONOMICE DIN BUCUREȘTI**
**Facultatea de Contabilitate și Informatică de Gestiune**
**Coordonator științific: Asist. univ. dr. Laura-Eugenia-Lavinia Barna**

**Rezumat**: *Pe o piață din ce în ce mai competitivă, cafenelele trebuie să găsească modalități de atragere și păstrare a clienților. Una dintre modalități este oferirea de promoții sau discount-uri în funcție de puncte. Prezenta lucrare arată o modalitate de implementare a unui sistem de fidelizare și retenție a clienților – compus din componenta casierului și aplicația clientului. Fiecare client primește un cod, pe care îl arată casierului când cumpără cafea. Clientul poate să verifice câte puncte are disponibile în aplicație și să folosească un număr de puncte în schimbul unei cafele gratis. Componenta casierului este scrisă în limbajul Typescript, folosind NextJs, iar aplicația clientului folosește Flutter, un kit de dezvoltare pentru aplicații multiplatformă. Aplicația își propune să atragă și să fidelizeze clienții unei cafenele.*

**Cuvinte-cheie:** cafenea, fidelizare, baze de date, multiplatformă, recompense

## 1. Introducere

Inspirația pentru această lucrare provine de la ideea de a oferi cadouri de fidelizare a clienților (oferirea gratuită a unei cafele), în același mod în care McDonald's oferă puncte în aplicație pentru fiecare comandă. Dintre cafenelele prezente în zona Piața Romană, doar Ted's Coffee Co. oferă un program de fidelitate. Astfel, mi-am pus întrebarea – cum ar arăta un asemenea sistem și cât de dificil ar fi de implementat. Această lucrare își propune să prezinte un răspuns la această întrebare.

Cel puțin în orizontul local se deschid din ce în ce mai multe cafenele. Nu toate reușesc pe o piață concurențială, dominată de giganți precum 5 to go, Starbucks, Ted's Coffee Co. Orice cafenea nou apărută pe piață este interesată să obțină notorietate pe piață în rândul clienților care consumă cafea de la concurență, încercând să obțină astfel o bucată din cota de piață. Totuși, o afacere nu beneficiază prea mult de pe urma clienților care cumpără o singură dată, astfel apărând necesitatea fidelizării și retenției acestora.

Fidelizarea este rezultatul experienței pozitive a clienților și contribuie la crearea încrederii în marcă (brand). Clienții fideli sunt dispuși să cumpere în mod repetat de la noi și nu de la concurenți, doar dacă compania pune la dispoziția clienților anumite oferte sau beneficii (Oracle, 2024).

Ca un program de fidelizare să aibă efect, clienții trebuie să considere că acesta le aduce valoare suplimentară, pe deasupra produselor/serviciilor cumpărate. Tipul de recompensă obținută, cât și durata de timp în care aceasta este obținută influențează drastic percepția consumatorului asupra mărcii, respectiv asupra programului de fidelizare (Başgöze et al., 2021). Un program de fidelizare include atât componenta de recompensă, cât și cea de comunicare, prin diferite modalități precum reclame pe social media, mail-uri, acestea având rolul să crească recunoașterea mărcii și să aducă mai mulți clienți în programul de fidelitate (Heesup et al., 2018).

Clienții nu sunt fidelizați doar de aplicație (și programul de fidelizare aferent), ci necesită și experiențe pozitive legate de customer service, calitatea produselor și altele. Totuși,

această lucrare tratează doar problema fidelizării clienților prin intermediul unei aplicații. De asemenea, aceste concepte se aplică oricărei afaceri de tip restaurant/cafenea.

Având ca sursă de inspirație aplicația celor de la Ted's Coffee Co., cât și cele de la McDonald's, KFC, Lidl și Lukoil, au fost identificate o serie de caracteristici comune ale aplicațiilor de fidelitate:

- fiecare utilizator are un cont propriu;
- fiecare utilizator are un cod (format din cifre și litere) care îl identifică, fiind unic (cardul de fidelitate);
- în anumite cazuri, utilizatorii pot avea un număr de puncte strânse în timp, pe care le pot da în schimbul unui produs/discount;
- în anumite cazuri, utilizatorii pot accesa direct cupoane de reducere, necondiționate de un număr de puncte;
- în anumite cazuri, punctele din unele aplicații expiră după un anumit timp.

Astfel, se diferențiază două tipuri de abordări asupra fidelității:

- puncte la schimb cu produse sau
- cupoane de reducere cu valabilitate limitată.

Un produs poate acorda unul sau mai multe puncte atașate cardului de fidelitate. Se remarcă două tipuri de carduri de fidelitate predominante: cardul „portofel" și cardul „talon". Cardul „portofel" stochează o mulțime de puncte (sute) care pot fi date la schimbul oricărui produs din promoție (ex. McDonald's), în timp ce cardul „talon" permite colectarea a mai puține puncte pentru un singur tip de produs.

Toate aceste abordări au același scop final, și anume să încurajeze clienții să cumpere mai mult și să devină clienți fideli brandului, fiind recompensați pentru asta.

Un alt motiv pentru implementarea unei aplicații de fidelitate (în locul cupoanelor/taloanelor tipărite) este scăderea cheltuielilor cu imprimarea și a impactului asupra mediului (în locul unei hârtiuțe care se aruncă la gunoi există aplicația, care nu produce deșeuri).

Având în vedere cele prezentate mai sus, aplicația propusă și implementată conține două componente principale, diferențiate de informația care poate fi accesată: prima parte este reprezentată de aplicația mobilă în care utilizatorul acumulează puncte pe un card de tip „talon", pe care le poate da în schimbul unor produse gratuite, iar a doua parte este reprezentată de aplicația casierului care gestionează cafeneaua.

McDonald's, Lidl și Lukoil au partea de fidelitate integrată în sistemul de comenzi sau al casei de marcat. Neavând acces la niciunul dintre acestea, am implementat strictul necesar pentru scanarea codului utilizatorului și adăugarea de puncte.

În scopuri de prezentare a fost implementată o promoție în cadrul căreia un client care cumpără 9 cafele (sau produse pe bază de cafea) o primește pe a 10-a gratuit. În cazul în care clientul are 7 puncte și cumpără 3 cafele, va avea în cont 10 puncte după tranzacție (deci va putea obține o cafea gratis și îi va rămâne un punct care se va raporta pentru următorul nivel de puncte care vor fi acumulate la achizițiile ulterioare).

## 2. Componenta casierului

### 2.1. Tehnologii utilizate

Componenta casierului utilizează framework-ul NextJs împreună cu limbajul TypeScript pentru Backend și Frontend, respectiv MongoDB pentru baza de date. Proiectul componentei casierului poate fi în Visual Studio Code sau orice alt IDE. Baza de date poate fi accesată folosind MongoDB Compass.

Deoarece JavaScript nu utilizează tipuri de date clar definite în mod explicit (eng. *static typing*), am decis să folosesc TypeScript, un limbaj de programare open-source dezvoltat de

Microsoft. Acesta este *„transpiled”*, adică transformat în JavaScript de către compilator, după un set de verificări asupra tipurilor de date. Printre avantajele acestei alegeri se numără eliminarea anumitor probleme cauzate de operații nedefinite pe variabile cu tipuri de date care se schimbă la runtime (de exemplu, adunare între un șir de caractere și un număr) și obținerea unei mai bune integrări cu mediul de dezvoltare.

Backend-ul este componenta aplicației care se ocupă cu menținerea unui REST API, gestionarea structurii modelului de date și servirea către utilizator a frontend-ului.

Un REST API se bazează pe conceptul de resurse (date specifice puse la dispoziția altor aplicații sau servicii web), care sunt accesate prin intermediul unor adrese unice (URI – Uniform Resource Identifier). Utilizarea acestuia se face prin trimiterea de cereri (eng. requests) HTTP, care pot fi de tipul GET (pentru a prelua date), POST (pentru a adăuga date), PUT (pentru a actualiza date) sau DELETE (pentru a șterge date). Aceste metode corespund operațiilor CRUD – Create, Read, Update, Delete. Toate aceste operații pot întoarce un răspuns în format JSON.

Aplicația prevede URI pentru fiecare tip de resursă sau acțiune importantă (client, scanare, cafenele, meniu, tranzacții). Acestea sunt accesibile pe ruta „[domeniu]/api/[nume resursă în engleză]” (exemplu: GET http://localhost:3000/api/menu -> returnează conținutul meniului în format JSON).

Frontend-ul este interfața grafică cu care interacționează utilizatorul. Aceasta este construită folosind React. Proiectul folosește Mantine pentru componente (elemente pre-definite și cu un design comun). Frontend-ul comunică cu backend-ul prin intermediul REST API-ului menționat mai sus. Aplicația mobilă descrisă mai jos este o altă implementare a frontend-ului (dar cu scop diferit). Frontend-ul este organizat sub forma unui SPA (Single Page Application). Datorită faptului că datele se pot schimba foarte repede, acestea sunt încărcate pe clientside, folosind strategia SWR (Stale While Revalidate). SWR menține un cache al datelor încărcate (care poate fi proaspăt – *fresh* sau vechi – *stale*) și descarcă din backend informații noi (*revalidate*) în urma acțiunilor care modifică date sau la reluarea focusului pe pagină, când se reia conexiunea la internet etc.

Putem considera că fiecare grupare de rută din backend cu pagina din frontend alcătuiesc un modul. Astfel, se definesc următoarele module (Tabelul 1):

Tabel 1. Conținutul modulelor

| **Modulul** | **Rută/rute** | **Conținut** |
|---|---|---|
| Dashboard | / | Prezintă statistici |
| Clienți | /clients<br>/api/clients | Arată o listă cu toți clienții |
| Scanare | /scan | Permite scanarea cardului de fidelitate, adăugarea punctelor, activarea promoțiilor |
| Cafenele | /cafes<br>/api/cafes | Arată o listă cu toate cafenelele înregistrate |
| Meniu | /menu<br>/api/menu | Arată o listă cu toate produsele înregistrate |
| Tranzacții | /transactions<br>/api/transactions | Arată o listă cu toate tranzacțiile efectuate |
| Client Autentificat | /api/private | Endpoint aplicație mobilă |

*Sursa:* Prelucrarea autorului

Există o singură rută diferită (/api/private), care răspunde doar la metoda HTTP GET. Aceasta reprezintă endpoint-ul la care se conectează aplicația mobilă. Primește ca date de intrare un request care conține un header de autorizare cu Bearer Token (MDN Web Docs,

2024). Acest token este verificat cu endpoint-ul "User Info" din auth0, care trimite ca răspuns și datele clientului dacă acesta este valid.

Datele clientului conțin un câmp numit "sub" (de la "subject"), care identifică unic un client atât în Auth0, cât și în baza de date a sistemului.

La fel ca celelalte rute, și ruta "/api/private" întoarce ca răspuns datele clientului serializate în format JSON.

### 2.2. Autentificare

Autentificarea aplicației casierului este gestionată de Auth0 (mai multe informații fiind prezentate în secțiunea despre Auth0). Conturile casierilor sunt separate de cele ale clienților cafenelei prin roluri. Panoul de autentificare al aplicației casierului este configurat să accepte doar conturi care au rolul de casier.

### 2.3.Baza de date

Baza de date a sistemului este o bază de date nonrelațională (MongoDB), bazată pe documente. Documentele pot conține, pe lângă tipuri de date simple (șiruri de caractere, numere, valori de tip boolean) și vectori, obiecte și referințe către alte documente. Mai multe documente formează o colecție, iar fiecare document respectă o schemă, care este parte din modelul de date definit de backend. Documentele folosesc pe post de cheie primară un număr de identificare. Acesta este un șir hexadecimal generat în funcție de timpul la care a fost adăugat documentul și este garantat să fie unic.

Baza de date conține următoarele colecții: clients, coffeeshops, products, transactions.

Tabel 2. Conținutul colecțiilor din baza de date

| Colecție | Conținuturi |
|---|---|
| clients | Toti clienții, identificați după „sub” (subject), împreună cu conținutul cardului de fidelitate |
| coffeeshops | Toate cafenelele, cu adresa și lista de email-uri pentru fiecare casier |
| products | Meniul cafenelei |
| transactions | Un istoric de tranzacții cu fiecare adăugare sau scadere de puncte |

*Sursa:* Prelucrarea autorului

Cea mai importantă colecție din baza de date este cea a clienților, care stochează informații despre clienți și conținutul cardului de fidelitate.

```
/** Represents a client, with identifying information and an object to store their loyalty points */
class Client {
    /** The client's unique identifier */
    @prop({ required: true })
    public sub?: string;

    /** The client's name */
    @prop()
    public name?: string;

    /** The client's email */
    @prop()
    public email?: string;

    /** The client's loyalty card */
    @prop({ default: new LoyaltyCard() })
    public loyaltyCard?: LoyaltyCard;
}

export default getModelForClass(Client);
```

Figura 1. Structura documentului care reprezintă un client

*Sursa:* Prelucrarea autorului

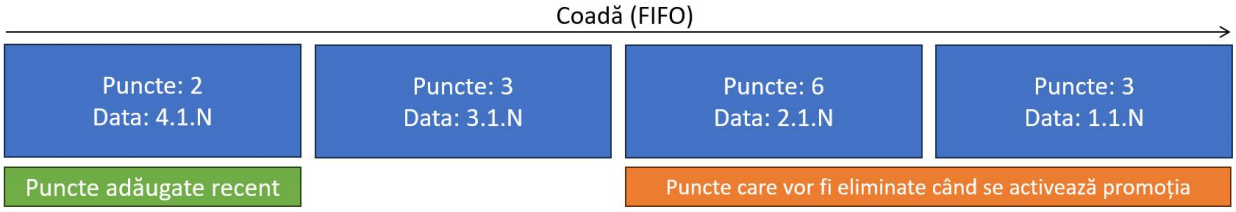
În implementarea inițială, aplicația utiliza e-mail-ul utilizatorului pe post de cheie primară. În teorie, acest lucru ar fi permis unificarea conturilor cu același email, indiferent de

modalitatea de autentificare (de exemplu, dacă utilizatorul folosea același e-mail și la contul de Google și la cel de Facebook, indiferent de metoda de conectare aleasă ar fi avut același cont cu același număr de puncte).

Conturile de Google și Apple au un email asociat, dar pentru conturile de Facebook acesta nu este obligatoriu. Deși am specificat pe lista de scopes că am nevoie și de e-mail (vezi secțiunea despre OAuth), acesta nu este garantat întotdeauna să existe.

Auth0 utilizează un id garantat să fie unic (și deci un candidat perfect să fie cheie primară), numit sub (subject), format din numele provider-ului și un cod numeric, mai multe informații fiind prezentate în secțiunea Auth0.

Cardul de fidelitate conține două câmpuri, unul reprezentând numărul de puncte, iar celălalt o coadă cu mai multe informații despre fiecare tranșă de puncte. Dat fiind faptul că punctele pot expira, sistemul le stochează sub formă unei cozi (metoda FIFO – First In, First Out). Punctele noi sunt adăugate ultimele, iar în cazul activării unei promoții, sunt utilizate punctele cele mai vechi (adică primele din coadă). Înainte de fiecare operațiune cu puncte se verifică valabilitatea acestora (pentru a ne asigura că nu utilizăm puncte expirate).




Cardul de fidelitate al clientului (*Client.loyaltyCard*)

Figura 2. Colecția de puncte asociată unui card de fidelitate
*Sursa:* Prelucrarea autorului

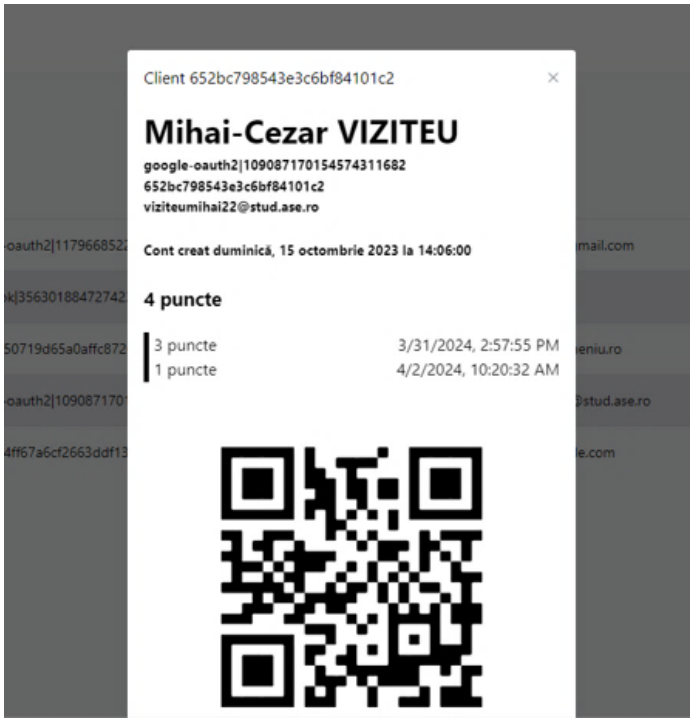



Figura 3. Lista de puncte din perspectiva casierului
*Sursa:* Prelucrarea autorului

## 3. Componenta clientului

### 3.1.Tehnologii utilizate

Componenta clientului este o aplicație mobilă, dezvoltată folosind framework-ul Flutter în limbajul Dart. Proiectul aplicației poate fi deschis în Android Studio.

Limbajul Dart este un limbaj de programare compilat, type-safe și null-safe, capabil să implementeze algoritmi complecși cu ajutorul operațiilor asincrone (*eng. async*) și al tipurilor de date nullable și non-nullable. Compilatorul Dart permite rularea de cod pe platforme native (*de exemplu pentru dispozitive mobile*) sau pe web (prin "traducerea" codului în JavaScript) (Dart, 2024). Codul nativ poate fi rulat JIT (*eng. just in time*) în timpul procesului de dezvoltare, apoi AOT (*eng. ahead of time*) pentru cod utilizat în producție. Modalitățile de rulare diferite oferă o experiență de dezvoltare foarte fluidă, respectiv o aplicație rapidă.

Flutter este un framework pentru dezvoltarea aplicațiilor multiplatformă, deci putem avea un singur codebase, din care rezultă aplicații funcționale pentru orice target (Flutter, 2024). Aplicația țintește în mod special Android și IOS. Componenta de bază a unei interfețe grafice dezvoltate în Flutter se numește *"widget"*. Acesta descrie logica, interacțiunile și design-ul unui element de interfață grafică. Modalitatea lor de funcționare este similară cu cea a componentelor din React, cu diferența că în Flutter, atât logica, cât și descrierea interfeței sunt scrise în Dart (spre deosebire de JavaScript + Html).

În limbajul Dart, operațiile pe fișiere și cele în rețea sunt asincrone, deoarece nu știm cu exactitate dacă și în cât timp se va termina execuția. O funcție asincronă returnează o valoare de tip Future<T>, care se transformă într-o valoare de tip T când operațiunea este finalizată.

Interfața grafică este alcătuită din multiple elemente care sunt afișate condițional în funcție de starea aplicației. Modulul de autentificare este sub forma unui Provider, care permite menținerea unei stări globale. Daca utilizatorul nu este autentificat, se afișează pagina de conectare (*login page*), altfel se interoghează serverul casierului și se obțin punctele (folosind datele din Provider).

Dat fiind că aplicația folosește cod dependent de rețea sau de sistemul de fișiere, interfața utilizează mai multe Widget-uri numite FutureBuilder. Acestea necesită doi parametri: *future* și *builder*. *Future* reprezintă operația asincronă care urmează să fie executată, iar *builder* este o funcție care are ca parametri context și snapshot. *Snapshot* este un obiect de tip AsyncSnapshot<T>, care are ca proprietăți statusul (waiting, active, done, none), datele și eventualele erori returnate de future.

```
return FutureBuilder<bool>(
      future: authProvider.isUserLoggedIn(),
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
          return const Scaffold(body: LoadingScreen(loadingText: "Fetching Login State..."));
        } else if (snapshot.hasError) {
          return Scaffold(body: ErrorScreen( friendlyText: "Error (while fetching login state):",  errorObject:
snapshot.error ));
        } else {
          final bool isUserLoggedIn = snapshot.data!;
          if (!isUserLoggedIn) {
            return const LoginPage();
          }

          //Rest of the app if the user is logged in
          return Scaffold(
```

Figura 4. Utilizarea FutureBuilder în codul aplicației
*Sursa:* Prelucrarea autorului

### 3.2.Interfața grafică prezentată utilizatorului autentificat

Pagina prezentată utilizatorului autentificat este compusă din trei panouri dispuse vertical. Primul panou conține numele, adresa de email, numărul de puncte, ID-ul unic al clientului împărțit în grupe de 4 caractere separate prin linii, reprezentarea ID-ului sub forma unui cod QR și un buton de refresh. Al doilea panou este o reprezentare digitală a cardului de tip "talon". Acesta are 9 pătrate care pot fi completate sau necompletate, în funcție de numărul de puncte deținute de utilizator. Dacă acesta are 9 sau mai multe puncte, se afișează un text care îl anunță că are suficiente puncte pentru o cafea gratis. Ultimul panou conține o listă cu punctele utilizatorului, sortate în funcție de data obținerii (respectiv data expirării).

Deși ID-ul clientului poate fi reprezentat de mai multe tipuri de coduri de bare, atât 1D (codul de bare de pe produse) cât și 2D (cod QR, aztec, datamatrix etc.), am ales să folosesc coduri QR deoarece nu necesită echipament special pentru scanare (orice dispozitiv cu o cameră video poate să scaneze un cod QR) și au o acuratețe foarte ridicată chiar și la distanțe mari (50 cm – 1 m). Un cod QR conține o serie de modele (eng. *patterns*) care ajută scannerul să îl găsească și scaneze, indiferent dacă acesta este rotit sau înclinat. Aceste coduri includ și un nivel de corectare a erorilor (eng. *error correction level*) (Denso Wave, 1994).

### 3.3.Autentificare și autorizare

Aplicația permite atât utilizarea unui cont cu e-mail și parolă, cât și login cu Google, Apple și Facebook. Pentru autentificare și autorizare am decis să utilizez o soluție de autentificare și autorizare terță chiar dacă implementarea uneia este relativ simplă, fiind vorba de stocarea datelor clienților, siguranța acestora este mai importantă.

Auth0 este o platformă de identitate și gestiune a accesului (IAM – *identity and access management*). Aceasta oferă un SDK (software development kit) pentru a permite dezvoltarea ușoară a aplicațiilor care folosesc această platformă.

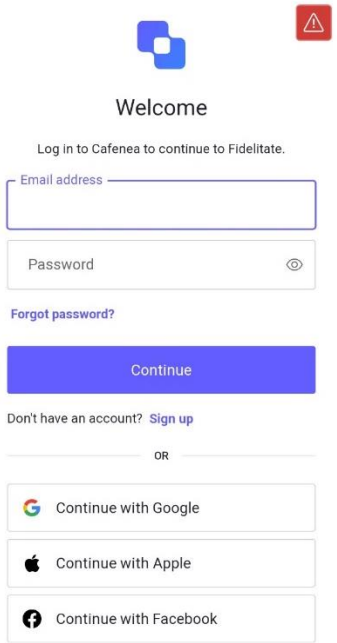



Figura 5. Ecranul de Login din aplicație
*Sursa:* Prelucrarea autorului

Referitor la clienții care nu dețin un telefon performant, soluția ar fi imprimarea unui cod QR pe un card. Deoarece codul cardului de fidelitate corespunde cu câmpul ID din baza de date (deci nu se schimbă vreodată), acesta poate fi tipărit și scanat ulterior. Clienții care nu

dispun de un telefon sau nu doresc să utilizeze aplicația pot să dețină un card și să îl prezinte casierului, cu diferența că nu își vor putea verifica punctele decât din interiorul unei cafenele.


Figura 6. Concept card de fidelitate tipărit
*Sursa:* Prelucrarea autorului

### 3.3.1. OAuth2.0

OAuth2.0 (*eng. Open Authorization 2*) este un standard de autorizare care permite utilizatorilor să ofere acces la resursele de pe un site către alt site, fără să partajeze datele de conectare. Este utilizat de butoanele de tip "Log In with Google" (Auth0, 2024).

OAuth definește *scopes* ca fiind un mecanism de limitare a accesului unei aplicații la contul unui utilizator. O aplicație poate solicita unul sau mai multe scopes, informație care este prezentată utilizatorului. Tokenul de acces returnat va fi limitat doar la accesul oferit de scopes (OAuth 2.0, 2024).

Cele 3 tipuri de conturi ale unor aplicații terțe (Google, Apple, Facebook) sunt folosite pe post de *identity providers* – furnizează aplicației informații despre utilizator. De asemenea, reduce numărul de conturi pe care utilizatorul trebuie sa le țină minte (un singur cont de Google/Apple/Facebook versus câte un cont pentru fiecare aplicație). Utilizarea autentificării prin OAuth permite aplicației să nu stocheze parole în baza de date, reducând astfel și riscul asociat unei posibile scurgeri de date.

### 3.3.2. **PKCE** (*eng. Proof Key for Code Exchange*)

PKCE este o procedură de autorizare. Aceasta are rolul să prevină CSRF (*eng. Cross Site Request Forgery* – o metodă de a forța un utilizator autentificat să facă o acțiune nedorită) și alte tehnici similare.

Prin secret se înțelege o valoare numerică sau alfa-numerică grea sau imposibilă de ghicit, generată de funcții sigure din punct de vedere criptografic.

Fiind vorba de o aplicație mobilă, aceasta nu poate utiliza un secret universal deoarece acesta ar putea fi aflat prin decompilare. PKCE presupune ca clientul să creeze un secret la fiecare cerere de autorizare, apoi să utilizeze acel secret din nou când face schimbul între cod de autorizare și access token. În acest fel, dacă codul de autorizare este interceptat, nu va fi util deoarece cererea pentru token necesită secretul inițial (OAuth (okta), 2024).

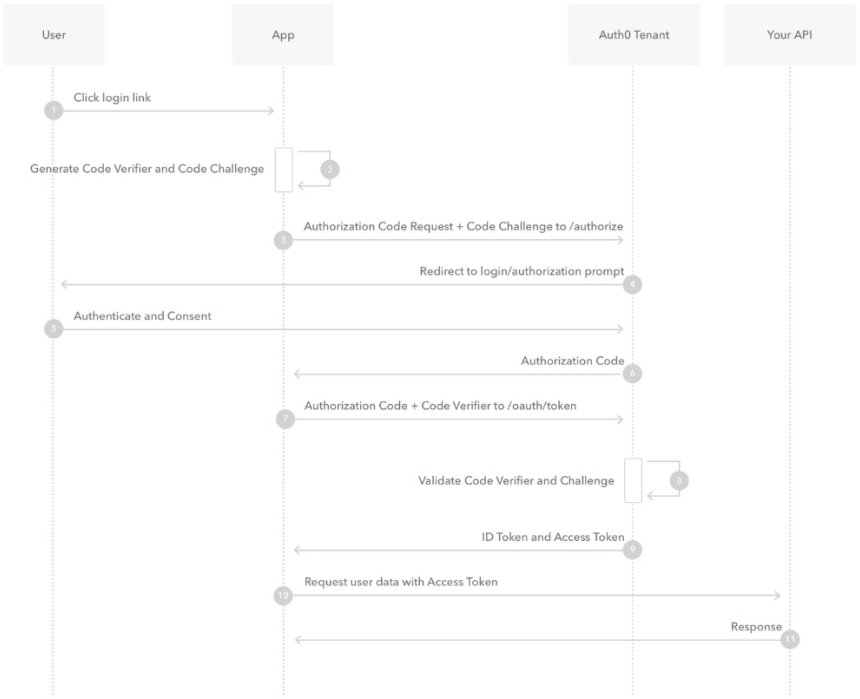



Figura 7. Diagrama (code flow) autentificării cu PKCE
*Sursa: Auth0, "Authorization Code Flow with Proof Key for Code Exchange (PKCE)", https://auth0.com/docs/get-started/authentication-and-authorization-flow/authorization-code-flow-with-pkce*

Când utilizatorul apasă butonul de Login din aplicație, SDK-ul Auth0 generează un set de valori numite code_verifier, respectiv code_challenge.

Code_verifier este un șir de caractere generat aleator, limitat la un set de caractere acceptate (alfa-numeric, minus, punct, underscore, tilde). Standardul RFC 7636 recomandă o secvență de cel puțin 32 octeți (Sakimura et al., 2015).

Code_challenge este generat în funcție de code_verifier. Există două metode de a obține un code_challenge: "plain" – ambele valori sunt egale, și "S256", unde code_verifier este *hashed* cu algoritmul sha256, apoi transformat în base64.

Auth0 redirecționează utilizatorul către serverul de autentificare (ruta */authorize*) și transmite code_challenge. Serverul redirecționează utilizatorul către pagina de login. Utilizatorul alege una din metodele permise (email, google etc) și eventual își dă acordul pentru partajarea informațiilor. Serverul Auth0 stochează code_challenge și redirecționează utilizatorul înapoi în aplicație cu un cod de autorizare, care funcționează o singură dată. Codul este trimis împreună cu code_verifier către ruta */oauth/token* a serverului, care verifică dacă toate valorile primite corespund, apoi răspunde cu un token de acces (eng. *access token*) și posibil un token de refresh (eng. *refresh token*).

Tabel 3. Exemplu de variabile folosite în procesul de autentificare

| **Variabilă** | **Exemplu** |
|---|---|
| code_verifier | QINoRVeQo7M4eqEOndSKw6FUKdpW9gbwZeE8QP9Dd7wb |
| code_challenge | ViALjAC6TeXEo2tChBVdU1AWcfsxMj9vEzkTSv427rE |
| Access token (JWT) | eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c |

*Sursa:* Prelucrarea autorului

La finalul acestui proces, utilizatorul este autentificat și poate utiliza access token-ul pentru a accesa informații.

În cazul aplicației, aceasta trimite un request către backend, pe ruta /api/private, împreună cu acel access token. Backend-ul verifică validitatea tokenului și trimite datele corespunzătoare utilizatorului în format JSON daca acesta (tokenul) este valid.

Aplicația deserializează răspunsul primit la request și afișează datele. Pentru a face interfața mai responsabilă (*eng. responsive*) și a permite aplicației să funcționeze și fără acces la internet, aceasta implementează un sistem de caching. Astfel, este suficient să se conecteze o singură dată la internet și să descarce datele care nu se schimbă niciodată (*de exemplu codul de client*).

Pagina prezentată utilizatorului pentru autentificare este deschisă în webview-ul implicit al dispozitivului și are aceleași cookies.

### 4. Scenariu de utilizare

În stadiul curent, aplicația este concepută să fie utilizată împreună cu o casă de marcat deja existentă. Componenta web poate rula pe un telefon mobil sau pe același computer cu casa de marcat. Cu ajutorul unui webcam (sau a camerei telefonului), casierul poate scana codurile clienților, adăuga sau elimina puncte etc.

Interfața grafică a aplicației casierului este concepută să fie ușor de utilizat și previzibilă, astfel încât casierul să nu piardă prea mult timp încercând să adauge puncte. Interfața grafică a aplicației clientului este simplă și aerisită, încercând să imite designul unui card de tip "talon".


Figura 8. Captură de ecran din aplicație
*Sursa: Prelucrarea autorului*

## 5. Introducere, aplicare (deployment) și îmbunătățiri posibile

În stadiul curent, aplicația ar trebui să poată fi utilizată într-un scenariu real. Fiind dezvoltată cu framework-ul NextJS, aplicația casierului poate fi găzduită direct pe platforma Vercel sau pe un VPS (eng. *Virtual Private Server*).

Aplicația mobilă poate fi compilată și adăugată la Play Store, App Store etc. Fiecare dintre aceștia percepe o taxă per cont sau per an.

Aplicația, respectiv baza de date sunt scalabile atât vertical cât și orizontal, dar nu anticipez ca sistemul să utilizeze prea multe resurse per total, fiecare client făcând câte o solicitare (*eng. request*) când intră în aplicație (ca să vizualizeze punctele).

Implementarea unui asemenea sistem de fidelitate nu ar trebui să fie o investiție foarte costisitoare pentru o cafenea/lanț de cafenele.

Posibilitățile viitoare de dezvoltare a aplicației includ:

- Integrarea cu casa de marcat / POS a cafenelei
- Adăugarea unor oferte (Monday Coffee – oferă o reducere de 10% unei categorii de cafea, Happy Hour – la o cafea comandata, primești o brioșă gratuit etc.)

## Concluzii

Orice afacere nou apărută pe piață trebuie să se remarce și să obțină o parte din cota de piață pentru a putea concura cu marii jucători de pe piață. O posibilă modalitate de a se remarca față de clienți este crearea unui program de fidelizare. Deși nu este suficientă, fidelizarea clienților este o parte importantă a activității de marketing a unei afaceri de tip restaurant/ cafenea. Lucrarea de față și-a propus să analizeze modul în care poate fi realizată fidelizarea clienților prin utilizarea unei aplicații informatice care să permită acest lucru.

Au fost identificate caracteristici și abordări comune în cazul programelor de fidelizare ale liderilor de piață. Indiferent de tehnologiile utilizate, aceste metode pot fi aplicate și în cazul unei afaceri aflate la început.

La nivel macro, sistemul este format din componenta casierului și cea a clientului. Componenta clientului are rolul să stocheze la nivelul bazei de date cardul de fidelitate al clientului, precum și datele despre clienți.

Sistemele de autentificare și gestionare a accesului protejează aceste date. Clientul folosește aplicația ca sa acceseze cardul de fidelitate și îl prezintă casierului. Astfel, un număr de puncte din contul clientului pot fi schimbate gratuit cu un produs.

Aplicația are rolul să digitalizeze sistemele existente de fidelizare bazate pe carduri fizice, să reducă cheltuielile cu imprimarea acestora, respectiv să reducă impactul negativ asupra mediului.

În concluzie, lucrarea a prezentat modul de realizare a unei aplicații pentru fidelizarea clienților unei cafenele, îmbinând atât aspectele teoretice, cât și cele practice (prin prezentarea tehnologiilor utilizate pentru implementarea acesteia), punând la dispoziția cititorilor acestei lucrări informații utile precum modalitatea de valorificare a activității unei cafenele, dar și principalele beneficii pe care le pot obține atât clienții, cât și managerii unei cafenele.

## Bibliografie


1. Auth0 (2024). *Protocols*. [Online]. Disponibil la: <https://auth0.com/docs/authenticate/protocols> [Accesat 25 Martie 2024]
2. Başgöze, P., Atay, Y., Metin Camgöz, S., & Hanks, L. (2021). The mediating effects of program loyalty in loyalty rewards programs: an experimental design in coffee shops. *Journal of Service Theory and Practice*, 31(6), 932-949. https://doi.org/10.1108/JSTP-01-2021-0020

3. Dart (2024). *Dart Overview*. [Online]. Disponibil la: <https://dart.dev/overview> [Accesat 25 Martie 2024]
4. Denso Wave Inc (1994). *Error Correction Feature*. [Online]. Disponibil la: <https://www.qrcode.com/en/about/error_correction.html> [Accesat 27 Martie 2024]
5. Flutter (2024). *Flutter Documentation*. [Online]. Disponibil la: <https://docs.flutter.dev/> [Accesat 23 Martie 2024]
6. Heesup, H., Hong, N.N., Hakjun, S., Bee-Lia, C., Sanghyeop, L., & Wansoo, K. (2018). Drivers of brand loyalty in the chain coffee shop industry. *International Journal of Hospitality Management*, 72, 86-97.
7. MDN Web Docs (2024). *HTTP authentication*. [Online]. Disponibil la: < https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication#authentication_schemes> [Accesat 23 Martie 2024]
8. OAuth 2.0 (2024). *OAuth Scopes*. [Online]. Disponibil la: < https://oauth.net/2/scope/> [Accesat 23 Martie 2024]
9. OAuth (okta) (2024). *Protecting Apps with PKCE*. [Online]. Disponibil la: <https://www.oauth.com/oauth2-servers/pkce/> [Accesat 24 Martie 2024]
10. Oracle (2024). *Ce este fidelitatea clienților?* [Online]. Disponibil la: <https://www.oracle.com/ro/cx/marketing/customer-loyalty/what-is-customer-loyalty/> [Accesat 15 Martie 2024]
11. Sakimura, N., Bradley, J., & Agarwal, N. (2015). *Proof Key for Code Exchange by OAuth Public Clients.* [Online] Internet Engineering Task Force (IETF). Disponibil la: <https://datatracker.ietf.org/doc/html/rfc7636> [Accesat 23 Martie 2024]