Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCA

Android aplikácia pre študentské zľavy s využitím polohy



2026

Bc. Ján Hyben

Vedúci práce:
Mgr. Markéta Trnečková, Ph.D.

Študijný program: Aplikovaná informatika, Specializace: Vývoj software

Bibliografické údaje

Autor:	Bc. Ján Hyben
Názov práce:	Android aplikácia pre študentské zľavy s využitím polohy
Typ práce:	diplomová práca
Pracovisko:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2026
Študijný program:	Aplikovaná informatika, Specializace: Vývoj software
Vedúci práce:	Mgr. Markéta Trnečková, Ph.D.
Počet strán:	91
Prílohy:	elektronické dáta v online úložisku katedry
Jazyk práce:	slovenský

Bibliographic info

Author:	Bc. Ján Hyben
Title:	Android application for student discounts using location
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2026
Study program:	Applied Computer Science, Specialization: Software Development
Supervisor:	Mgr. Markéta Trnečková, Ph.D.
Page count:	91
Supplements:	electronic data in the department's online repository
Thesis language:	Slovak

Anotácia

Diplomová práca sa zaoberá návrhom a implementáciou mobilnej aplikácie PerkAround, ktorej cieľom je sprístupniť študentom prehľad aktuálnych zliav dostupných v ich bezprostrednom okolí. Aplikácia využíva GPS polohu zariadenia na zobrazenie blízkych prevádzok s aktívnymi ponukami a na zasielanie notifikácií pri vstupe do ich blízkosti. Mobilný klient je vyvinutý pomocou frameworku .NET MAUI pre platformu Android, serverová časť je implementovaná ako REST API v prostredí ASP.NET Core s databázou PostgreSQL. Dáta o prevádzkach a zlavách sú automatizovane získavané z portálu ISIC prostredníctvom samostatnej služby na zber dát (tzv. web scraping) postavenej na knižnici Playwright. Autentifikácia je riešená pomocou JWT tokenov v kombinácii s refresh tokenmi. Práca sa venuje návrhu systému, implementačným rozhodnutiam, testovaniu a nasadeniu aplikácie vrátane súladu so zákonnou úpravou ochrany osobných údajov podľa nariadenia GDPR.

Synopsis

Master thesis presents the design and implementation of a mobile application called PerkAround, aimed at helping students discover current discounts available in their immediate vicinity. The application uses the device's GPS location to display nearby venues with active offers and to send notifications when the user approaches them. The mobile client is developed using the .NET MAUI framework targeting the Android platform, while the server-side component is implemented as a REST API built on ASP.NET Core with a PostgreSQL database. Data about venues and discounts is automatically collected from the ISIC portal by a dedicated web scraping service based on the Playwright library. Authentication is handled using JWT access tokens in combination with refresh tokens. The thesis covers system design, implementation decisions, testing, and deployment, including compliance with personal data protection requirements under the GDPR regulation.

Kľúčové slová: Android aplikácia; geolokácia; študentské zľavy; REST API; .NET MAUI; JWT autentifikácia; GDPR

Keywords: Android application; geolocation; student discounts; REST API; .NET MAUI; JWT authentication; GDPR

Ďakujem vedúcej mojej práce Mgr. Markéte Trnečkovej, PhD. za odborné rady a usmernenia pri procese vývoja aplikácie a písaní záverečnej práce. Moje vďaka patrí aj mojej rodine, priateľke a priateľom za ich podporu a povzbudenie počas celého môjho štúdia.

Odovzdaním tohto textu jeho autor/ka čestne vyhlasuje, že celú prácu vrátane príloh vypracoval/a samostatne a za použitia iba zdrojov spomínaných v texte práce a uvedených v zozname literatúry.

Obsah

1 Úvod **10**
1.1 Ciele práce . . . 11

2 Analýza problematiky **12**
2.1 Mobilné aplikácie založené na polohe . . . 12
2.1.1 Location-based services (LBS) . . . 12
2.1.2 GPS, presnosť, obmedzenia . . . 13
2.2 Študentské zľavy a digitálne platformy . . . 14
2.3 Existujúce riešenia . . . 14
2.3.1 Webová stránka ISIC . . . 15
2.3.2 Alive App . . . 16
2.4 Autentifikácia . . . 17
2.4.1 JSON Web Token (JWT) . . . 17
2.4.2 Tokeny . . . 18
2.4.3 Bezpečnosť . . . 19
2.5 REST architektúra . . . 19
2.5.1 REST princípy . . . 20
2.5.2 HTTP metódy . . . 20
2.5.3 Stavovosť vs. bezstavovosť . . . 21
2.6 Výber technológií . . . 21
2.6.1 Mobilný klient: .NET MAUI . . . 21
2.6.2 Serverová platforma a databáza: ASP.NET Core a PostgreSQL . . . 22
2.6.3 Automatizovaný zber dát: Playwright . . . 22
2.6.4 Prístup k notifikáciám . . . 23
2.6.5 Mapové podklady: Leaflet a OpenStreetMap . . . 23

3 Návrh systému **24**
3.1 Funkčné požiadavky . . . 24
3.2 Nefunkčné požiadavky . . . 25
3.2.1 Správa používateľských údajov podľa GDPR . . . 26
3.3 Architektúra systému . . . 26
3.3.1 Klient-server model . . . 26
3.4 Návrh databázového modelu . . . 27
3.4.1 ER diagram . . . 27
3.4.2 Prehľad tabuliek . . . 28
3.5 Návrh autentifikačného mechanizmu . . . 29
3.6 Návrh používateľského rozhrania . . . 29
3.6.1 Wireframy . . . 30
3.7 Vizuálna identita aplikácie . . . 32
3.7.1 Logo a ikona aplikácie . . . 32

3.7.2 Farebná schéma . . . 33
3.7.3 Typografia . . . 34
3.7.4 Tvorba grafických podkladov . . . 34

4 Implementácia riešenia 35
4.1 Implementácia backendu . . . 35
4.1.1 Konfigurácia . . . 35
4.1.2 Štruktúra projektu . . . 35
4.1.3 Dátové modely a databázový kontext . . . 36
4.1.4 Kontroléry . . . 37
4.1.5 Služby . . . 38
4.1.6 Automatizovaný zber dát . . . 38
4.2 Implementácia autentifikácie . . . 40
4.2.1 JWT konfigurácia . . . 40
4.2.2 TokenService . . . 40
4.2.3 Registrácia a prihlásenie . . . 40
4.2.4 Obnovenie tokenov, rotácia a odhlásenie . . . 41
4.2.5 Obnovenie zabudnutého hesla . . . 41
4.2.6 Ochrana endpointov . . . 42
4.3 Implementácia mobilnej aplikácie . . . 43
4.3.1 .NET MAUI . . . 43
4.3.2 Organizácia klienta . . . 43
4.3.3 Štýly a témy . . . 43
4.3.4 Prihlásenie a registrácia . . . 45
4.3.5 Obnovenie zabudnutého hesla . . . 46
4.3.6 Navigácia . . . 47
4.3.7 MapView . . . 49
4.3.8 FavView a OffersView . . . 50
4.3.9 Detail ponuky . . . 54
4.3.10 SettingsView . . . 55
4.3.11 ProfileView . . . 56
4.3.12 GDPR . . . 57
4.4 Komunikácia klient-server . . . 59
4.4.1 HttpClient . . . 59
4.4.2 JSON serializácia . . . 59
4.4.3 Ošetrenie chýb . . . 60
4.5 Geolokácia . . . 61
4.5.1 Získavanie polohy . . . 61
4.5.2 Výpočet vzdialenosti . . . 61
4.5.3 Filtrovanie podľa vzdialenosti . . . 62
4.6 Notifikačný systém . . . 63
4.6.1 Architektúra . . . 63
4.6.2 Preferencie a rozsah notifikácií . . . 63
4.6.3 Spúšťanie notifikácií a obmedzovanie frekvencie . . . 63

4.6.4 Vyhodnotenie a zostavenie notifikácie . . . 64
4.6.5 Zobrazenie a beh na pozadí . . . 64
4.7 E-maily a newsletter . . . 65
4.7.1 Odosielanie e-mailov . . . 65
4.7.2 Plánovanie e-mailov . . . 65
4.7.3 Personalizovaný obsah . . . 66
4.7.4 Odhlásenie z odberu . . . 66
4.8 Nastavenia a personalizácia . . . 67
4.8.1 Dvojvrstvová perzistencia . . . 67
4.8.2 Vplyv personalizácie na správanie . . . 67
4.9 Odolnosť a offline režim . . . 68
4.9.1 Indikácia stavu pripojenia . . . 68
4.9.2 Offline dostupnosť obľúbených . . . 68
4.9.3 Ošetrenie sieťových chýb . . . 69
4.10 Aktualizácia aplikácie . . . 69
4.10.1 Zverejnenie verzie na serveri . . . 69
4.10.2 Kontrola dostupnosti aktualizácie . . . 70
4.10.3 Stiahnutie a inštalácia . . . 70

5 Testovanie **71**
5.1 Vývojové a testovacie nástroje . . . 71
5.1.1 Swagger . . . 71
5.1.2 Docker . . . 71
5.1.3 pgAdmin . . . 72
5.1.4 Zariadenie s operačným systémom Android . . . 72
5.1.5 Fake GPS . . . 73
5.2 Testovanie mobilnej aplikácie . . . 73
5.2.1 Metóda testovania . . . 73
5.2.2 Testovacie prostredie a distribúcia . . . 73
5.2.3 Testované scenáre . . . 74
5.2.4 Nájdené chyby a ich riešenie . . . 74
5.2.5 Spätná väzba a použiteľnosť . . . 75
5.2.6 Zhrnutie . . . 75
5.3 Testovanie bezpečnosti . . . 76
5.3.1 Expirácia tokenu . . . 76
5.3.2 Neplatný token . . . 76
5.3.3 Neoprávnený prístup . . . 77

6 Nasadenie systému **78**
6.1 Serverové prostredie . . . 78
6.2 Kontajnerizácia . . . 79
6.3 Reverzný proxy server a zabezpečenie HTTPS . . . 80
6.4 Konfigurácia a tajné údaje . . . 81
6.5 Distribúcia aplikácie . . . 82

7 Výsledky práce **83**
7.1 Splnenie cieľov 83
7.2 Limity riešenia 84
7.3 Možné rozšírenia 85

Záver **86**

Conclusions **87**

A Obsah elektronických dát **88**

Literatúra **89**

Zoznam obrázkov

1 Aplikácia Alive App: úvodná stránka, nástenka s digitálnym preukazom ISIC (vľavo), katalóg zliav s vyhľadávaním, kategóriami a obľúbenými (v strede) a mapa zliav v okolí (vpravo). 16
2 Konceptuálny ER model databázy systému PerkAround. Podčiarknuté sú primárne kľúče, kurzívou vyznačené atribúty predstavujú cudzie kľúče vyjadrujúce daný vzťah. 27
3 Wireframy hlavných obsahových kariet: mapa s okolím (vľavo), zoznam ponúk (v strede) a obľúbené prevádzky (vpravo). 30
4 Wireframy vyhľadávania a detailu: zadávanie hľadaného výrazu (vľavo), výsledky vyhľadávania (v strede) a detail prevádzky (vpravo). 31
5 Wireframy účtu a nastavení: nastavenia (vľavo), profil používateľa (v strede) a rozbaľovacia ponuka (vpravo). 32
6 Logo aplikácie PerkAround. 33
7 Ikona aplikácie PerkAround. 33
8 Farebná paleta aplikácie PerkAround. 34
9 Farby pozadia a textu vo svetlom a tmavom režime. 44
10 Prihlasovacia obrazovka (vľavo) a registračný formulár s pravidlami hesla a povinným súhlasom so spracúvaním osobných údajov (vpravo). 45
11 Obnova hesla: zadanie e-mailu na odoslanie kódu (vľavo) a zadanie šesťciferného kódu spolu s novým heslom (vpravo). 46
12 Úvodná obrazovka s mapou (*MapView*) po spustení aplikácie (vľavo) a rozbalené *burger* menu s e-mailom prihláseného používateľa a ďalšími sekundárnymi akciami (vpravo). 48
13 Mapové zobrazenie prevádzok v okolí (vľavo) a vyskakovacie okno s názvom prevádzky, vzdialenosťou a tlačidlom „View more“ po klepnutí na fialový marker (vpravo). 50
14 Karta prevádzky v troch stavoch: štandardná ponuka v okolí s uvedenou vzdialenosťou (hore), online poskytovateľ bez fyzickej pobočky (v strede) a prevádzka mimo nastaveného okruhu vyhľadávania (dole). 51
15 Obrazovka `OffersView` s tlačidlami „Show all offers“ a „Search offers“, pruhom filtrov podľa kategórií (aktívny je „Gastronomy & Food“) a zoznamom ponúk v okolí zoradeným podľa vzdialenosti. 52
16 Obrazovka `FavView` (Favorites) so zoznamom obľúbených poskytovateľov, pri každej je uvedená vzdialenosť k najbližšej pobočke. . 53
17 Detail ponuky (`ProviderDetailPopup`) otvorený z mapy. . . . 55
18 Ukážka e-mailov generovaných serverom: personalizovaný newsletter s ponukami v okolí (vľavo) a narodeninový pozdrav (vpravo). 66

1 Úvod

Študentské obdobie je, popri akademických povinnostiach, často spojené s obmedzenými finančnými prostriedkami. Mnohí študenti sú odkázaní na finančnú podporu rodiny, štipendiá alebo príležitostné brigády, pričom zároveň čelia rôznym výdavkom súvisiacim so štúdiom a každodenným životom. Z tohto dôvodu sa snažia hospodáriť so svojimi financiami čo najefektívnejšie a vyhľadávať možnosti, ako ušetriť pri nákupe tovarov a služieb.

Jedným z najrozšírenejších nástrojov na získavanie a uplatňovanie študentských zliav je medzinárodný študentský preukaz ISIC (*International Student Identity Card*). Tento preukaz umožňuje študentom využívať rôzne zľavy poskytované obchodníkmi, kultúrnymi inštitúciami, dopravnými spoločnosťami alebo poskytovateľmi iných služieb. Držiteľom preukazu ISIC môže byť v Českej republike osoba mladšia ako 26 rokov,[1] ktorá je zapísaná na štúdium v prezenčnej forme a má tak status študenta[2]. Zľavy môžu zahŕňať napríklad znížené vstupné do kultúrnych zariadení, lacnejšie stravovanie v reštauráciách, výhodnejšie cestovné alebo špeciálne ponuky v internetových obchodoch. [1]

Preukaz ISIC vydáva medzinárodná organizácia *ISIC Association*, ktorej cieľom je podporovať mobilitu študentov a poskytovať im prístup k rôznym výhodám a zľavám v oblastiach dopravy, kultúry, vzdelávania, stravovania či voľnočasových aktivít. V mnohých krajinách zároveň slúži aj ako oficiálny identifikačný preukaz študenta v rámci školských systémov. Aj na našej univerzite ISIC slúži okrem iného ako prístupová čipová karta, ktorá oprávneným študentom umožňuje prístup do odborných učební a na príslušnú katedru. Vďaka širokej akceptácii umožňuje držiteľom využívať tisíce zliav po celom svete, pričom ponuky poskytujú nielen veľké medzinárodné spoločnosti, ale aj lokálne podniky. [1]

Napriek veľkému množstvu dostupných zliav sú informácie o nich často roztrúsené medzi rôzne webové stránky, mobilné aplikácie alebo marketingové kampane jednotlivých podnikov. Študenti tak často nemajú prehľad o všetkých možnostiach, ktoré môžu využiť, a o mnohých zľavách sa dozvedia až náhodne alebo vôbec. V praxi to znamená, že potenciálne výhody spojené so študentským statusom zostávajú často nevyužité. Motiváciou pre vznik tejto práce je preto návrh riešenia, ktoré by študentom umožnilo jednoduchšie objavovať a využívať zľavy dostupné v ich okolí. Cieľom je navrhnúť a implementovať mobilnú aplikáciu, ktorá bude agregovať informácie o študentských zľavách a zároveň používateľov upozorňovať na relevantné ponuky na základe ich aktuálnej polohy. Takýto prístup môže prispieť k lepšej dostupnosti informácií o zľavách a zároveň podporiť efektívnejšie využívanie výhod vyplývajúcich zo študentského statusu.

[1] Preukaz ISIC sa zvykne označovať aj ako ISIC<26, pričom <26 značí vek menší ako 26 rokov.

[2] Podmienky sa môžu líšiť podľa krajiny a typu štúdia.

1.1 Ciele práce

Hlavným cieľom tejto práce je navrhnúť a implementovať funkčnú mobilnú aplikáciu pre platformu Android, ktorá študentom sprístupní prehľad aktuálnych zliav dostupných v ich bezprostrednom okolí a na základe ich aktuálnej polohy ich upozorní na relevantné ponuky. Aplikácia má na jednom mieste agregovať informácie o študentských zľavách, ktoré sú v súčasnosti roztrúsené medzi rôzne zdroje, a prostredníctvom geolokácie zvýšiť pravdepodobnosť, že používateľ dostupnú zľavu skutočne využije.

Riešenie nie je koncipované len ako samotná mobilná aplikácia, ale ako ucelený systém pozostávajúci z mobilného klienta, serverového rozhrania a mechanizmu automatizovaného získavania dát o zľavách. Na dosiahnutie hlavného cieľa boli stanovené nasledujúce čiastkové ciele:

- Analyzovať problematiku mobilných aplikácií založených na polohe, oblasť študentských zliav a existujúcich digitálnych platforiem, ako aj princípy autentifikácie používateľov a architektúry REST.
- Na základe analýzy definovať funkčné a nefunkčné požiadavky na systém a navrhnúť jeho architektúru vrátane databázového modelu a autentifikačného mechanizmu.
- Implementovať serverovú časť ako REST API v prostredí ASP.NET Core s databázou PostgreSQL vrátane efektívneho vyhľadávania prevádzok podľa geografickej polohy.
- Navrhnúť a implementovať službu na automatizovaný zber údajov o prevádzkach a zľavách z portálu ISIC a ich synchronizáciu do databázy systému.
- Implementovať mobilného klienta pomocou frameworku .NET MAUI, ktorý používateľovi umožní zobrazovať ponuky na mape aj v zozname, vyhľadávať a filtrovať ich, spravovať obľúbené prevádzky a svoj používateľský účet.
- Zabezpečiť bezpečnú autentifikáciu používateľov pomocou tokenov JWT v kombinácii s obnovovacími tokenmi a bezpečné ukladanie citlivých údajov na strane klienta.
- Zabezpečiť spracúvanie osobných údajov v súlade s požiadavkami nariadenia GDPR.
- Otestovať výsledné riešenie a nasadiť ho do produkčného prostredia.

Splnenie jednotlivých cieľov je vyhodnotené v závere práce.

2 Analýza problematiky

Pred samotným návrhom a implementáciou je potrebné analyzovať technológie a koncepty, na ktorých navrhovaná mobilná aplikácia stavia. V rámci kapitoly sú predstavené základné princípy mobilných aplikácií založených na určovaní polohy používateľa, ktoré umožňujú poskytovať služby a informácie v závislosti od aktuálnej geografickej polohy zariadenia. Ďalej je pozornosť venovaná problematike študentských zliav a existujúcim digitálnym platformám, ktoré sprostredkúvajú zľavové ponuky pre študentov. Súčasťou analýzy je tiež oblasť autentifikácie používateľov, ktorá je nevyhnutná pre bezpečný prístup k aplikáciám a webovým službám. V závere kapitoly sú predstavené základné princípy architektúry REST, ktorá predstavuje bežný spôsob komunikácie medzi mobilnými aplikáciami a ich serverovými rozhraniami.

2.1 Mobilné aplikácie založené na polohe

Mobilné aplikácie založené na polohe predstavujú kategóriu aplikácií, ktoré využívajú geografickú polohu používateľa na poskytovanie relevantných informácií alebo služieb. Tento prístup je známy ako *location-based services* (skr. LBS) a umožňuje prispôsobovať obsah aplikácie podľa aktuálnej polohy mobilného zariadenia. Vďaka rozšíreniu smartfónov vybavených technológiami na určovanie polohy, ako sú globálne navigačné satelitné systémy, mobilné siete alebo Wi-Fi siete, sa tieto služby stali bežnou súčasťou moderných mobilných aplikácií.

Mobilné aplikácie sú pre implementáciu služieb založených na polohe obzvlášť vhodné, keďže mobilné zariadenia sú väčšinou neustále prítomné pri používateľovi a dokážu priebežne získavať informácie o jeho aktuálnej polohe. To umožňuje aplikáciám poskytovať používateľom kontextovo relevantné informácie v reálnom čase a reagovať na zmeny prostredia alebo polohy zariadenia.

Služby založené na polohe sa dnes využívajú v mnohých oblastiach. Typickým príkladom sú navigačné aplikácie, ktoré poskytujú používateľom informácie o trase a aktuálnej polohe. Ďalším príkladom sú aplikácie umožňujúce vyhľadávanie podnikov alebo služieb v okolí používateľa, napríklad reštaurácií, obchodov alebo dopravných spojení. V marketingu sa technológie založené na polohe využívajú na zobrazovanie personalizovaných ponúk alebo reklám v závislosti od polohy používateľa. Podobný princíp využívajú aj aplikácie zamerané na zľavy, ktoré môžu používateľom zobrazovať aktuálne zľavové ponuky dostupné v ich bezprostrednom okolí.

2.1.1 Location-based services (LBS)

Location-based services (LBS) predstavujú služby, ktoré využívajú geografickú polohu mobilného zariadenia na poskytovanie informácií alebo funkcií relevantných pre používateľa. Základnou myšlienkou LBS je prispôsobenie obsahu alebo služieb na základe aktuálnej polohy zariadenia, čím je možné používateľovi poskytovať kontextovo relevantné informácie v reálnom čase. Tento prístup sa uplat-

ňuje v mnohých oblastiach, napríklad v navigačných aplikáciách, pri vyhľadávaní podnikov v okolí alebo pri zobrazovaní lokalizovaných marketingových ponúk [2].

Služby založené na polohe sú zvyčajne tvorené viacerými základnými komponentmi. Prvým z nich je mobilné zariadenie používateľa, ktoré slúži ako klientská platforma a poskytuje údaje o polohe. Druhým komponentom je lokalizačná technológia, ktorá umožňuje určiť geografickú polohu zariadenia, napríklad prostredníctvom satelitných navigačných systémov, mobilných sietí alebo bezdrôtových Wi-Fi sietí. Ďalšou súčasťou je serverová infraštruktúra, ktorá spracováva požiadavky klientských aplikácií, uchováva dáta a poskytuje relevantné informácie na základe získanej polohy. Posledným komponentom je aplikačná vrstva, ktorá zabezpečuje samotnú funkcionalitu služby a prezentuje výsledky používateľovi prostredníctvom používateľského rozhrania.

Z hľadiska spôsobu interakcie používateľa so službou je možné LBS rozdeliť na dva základné typy. Prvým typom sú takzvané *pull* služby, pri ktorých používateľ aktívne vyhľadáva informácie na základe svojej polohy, napríklad pri vyhľadávaní reštaurácií alebo obchodov v okolí. Druhým typom sú *push* služby, pri ktorých systém automaticky poskytuje používateľovi informácie alebo upozornenia bez potreby jeho aktívnej požiadavky. V tomto prípade môže aplikácia napríklad informovať používateľa o dostupnej zľave v blízkosti jeho aktuálnej polohy [2].

S push službami úzko súvisí koncept tzv. *geofencingu*, ktorý umožňuje definovať virtuálne geografické oblasti a reagovať na situácie, keď používateľ do takejto oblasti vstúpi alebo ju opustí. Geofencing sa často využíva práve v marketingových alebo zľavových aplikáciách, kde môže systém používateľa automaticky upozorniť na dostupné ponuky v okolí jeho aktuálnej polohy.

2.1.2 GPS, presnosť, obmedzenia

Jednou z najpoužívanejších technológií na určovanie polohy mobilných zariadení je globálny navigačný satelitný systém GPS (angl. *Global Positioning System*). Ide o satelitný systém, ktorý umožňuje určiť geografickú polohu zariadenia na základe signálov prijímaných zo siete satelitov obiehajúcich okolo Zeme. Mobilné zariadenie prijíma signály z viacerých satelitov a na základe času potrebného na prenos signálu vypočítava svoju približnú polohu. Tento proces je založený na princípe trilaterácie, pri ktorom sa poloha zariadenia určuje na základe vzdialeností od viacerých satelitov. Pri priaznivých podmienkach môže GPS dosahovať presnosť približne 5 až 10 metrov. [3]

Napriek pomerne vysokej presnosti má GPS aj určité obmedzenia. Presnosť určovania polohy môže byť výrazne znížená v mestskom prostredí s vysokými budovami, ktoré môžu blokovať alebo odrážať satelitné signály. Podobný problém nastáva aj v interiéroch budov, kde je príjem satelitného signálu často nedostatočný alebo úplne nemožný. Ďalším faktorom je spotreba energie, keďže neustále využívanie GPS môže výrazne ovplyvniť výdrž batérie mobilného zariadenia. Okrem toho môže dochádzať aj k miernemu oneskoreniu pri získavaní presnej polohy, najmä pri prvotnom určovaní polohy zariadenia.

Z týchto dôvodov mobilné zariadenia často využívajú aj alternatívne metódy určovania polohy. Jednou z nich je lokalizácia pomocou Wi-Fi sietí, pri ktorej sa poloha zariadenia odhaduje na základe dostupných bezdrôtových sietí v okolí. Ďalšou možnosťou je určovanie polohy pomocou mobilných základňových staníc, teda vysielačov mobilnej siete. Tieto metódy síce zvyčajne nedosahujú takú presnosť ako GPS, avšak môžu byť efektívnejšie v prostrediach, kde je satelitný signál nedostupný.

Moderné mobilné operačné systémy preto pri určovaní polohy často kombinujú viacero zdrojov lokalizačných údajov. Kombináciou údajov z GPS, Wi-Fi sietí a mobilných základňových staníc je možné dosiahnuť rýchlejšie a spoľahlivejšie určenie polohy zariadenia v rôznych typoch prostredia.

2.2 Študentské zľavy a digitálne platformy

Študentské zľavy sú formou cenovej diferenciácie, ktorou podniky oslovujú mladšiu klientelu v odvetviach ako doprava, gastronómia, kultúra či maloobchod. Reagujú tým na obmedzené finančné možnosti študentov, ktorí zároveň predstavujú perspektívnu skupinu zákazníkov s dlhodobým spotrebiteľským potenciálom: zvýhodnená cena zvyšuje šancu, že si študent vyberie danú službu, a môže prispieť k jeho neskoršej lojalite. Pre študenta je zľava naopak možnosťou znížiť bežné výdavky a sprístupniť si služby, ktoré by inak boli finančne menej dostupné, čím sa stáva vnímaným benefitom spojeným so statusom študenta.

S rozvojom digitálnych technológií sa zľavové systémy postupne presúvajú z papierových kupónov a vernostných kariet do digitálnej podoby dostupnej cez webové platformy a mobilné aplikácie. Tie zjednodušujú správu ponúk na strane poskytovateľov a používateľom poskytujú rýchly prístup k aktuálnym informáciám o dostupných zľavách.

Digitálna podoba zliav zároveň otvára priestor pre personalizáciu a zobrazovanie ponúk podľa aktuálnej polohy používateľa. Práve mobilné aplikácie sú na sprostredkovanie takýchto ponúk vhodnou platformou, keďže spájajú okamžitú dostupnosť informácií, jednoduché používateľské rozhranie a prepojenie s ďalšími technológiami mobilných zariadení.

2.3 Existujúce riešenia

V súčasnosti existuje viacero digitálnych platforiem, ktoré poskytujú študentom prístup k zľavám a výhodám, či už prostredníctvom webových stránok, alebo mobilných aplikácií. Tieto riešenia majú za cieľ centralizovať zľavové ponuky a sprístupniť ich používateľom na jednom mieste, líšia sa však spôsobom fungovania a mierou, do akej pracujú s polohou používateľa.

Keďže navrhovaná aplikácia je zameraná na študentské zľavy viazané na preukaz ISIC, táto časť sa podrobnejšie venuje dvom najrelevantnejším oficiálnym riešeniam: webovej stránke ISIC a mobilnej aplikácii Alive App.

2.3.1 Webová stránka ISIC

Základnou platformou na vyhľadávanie študentských zliav v Českej republike je oficiálna webová stránka `isic.cz`, ktorú prevádzkuje organizácia zastrešujúca preukazy ISIC. Web slúži ako centrálny rozcestník, ktorý okrem predaja samotných preukazov (ISIC, ISIC Scholar, ITIC, IYTC a AliveID) sústreďuje na jednom mieste prehľad partnerských zliav a ďalšie služby pre študentov, ako je sprostredkovanie brigád, cestovné a zdravotné poistenie či aktuality zo študentského prostredia.

Jadrom stránky je z pohľadu tejto práce sekcia so zľavami, ktorá funguje ako katalóg partnerských ponúk. Používateľ v nej môže zľavy vyhľadávať alebo filtrovať podľa kategórií. Ponuky sú rozdelené do jednotlivých kategórií. Zoznam zliav je možné zoraďovať podľa viacerých kritérií, konkrétne podľa odporúčania, abecedy, vzdialenosti, novosti alebo blížiaceho sa konca platnosti.

Stránka rozlišuje medzi zľavami uplatniteľnými online a zľavami platnými v kamenných prevádzkach a umožňuje medzi nimi filtrovať pomocou samostatných prepínačov. Pre zľavy viazané na konkrétne miesto poskytuje aj geografické zobrazenie – ponuky je možné prezerať buď ako zoznam, alebo na mape, čo umožňuje vyhľadať prevádzky poskytujúce zľavu v danej lokalite. Každá zľava má zároveň vlastnú detailnú stránku s informáciami o ponuke a spôsobe jej uplatnenia. [4]

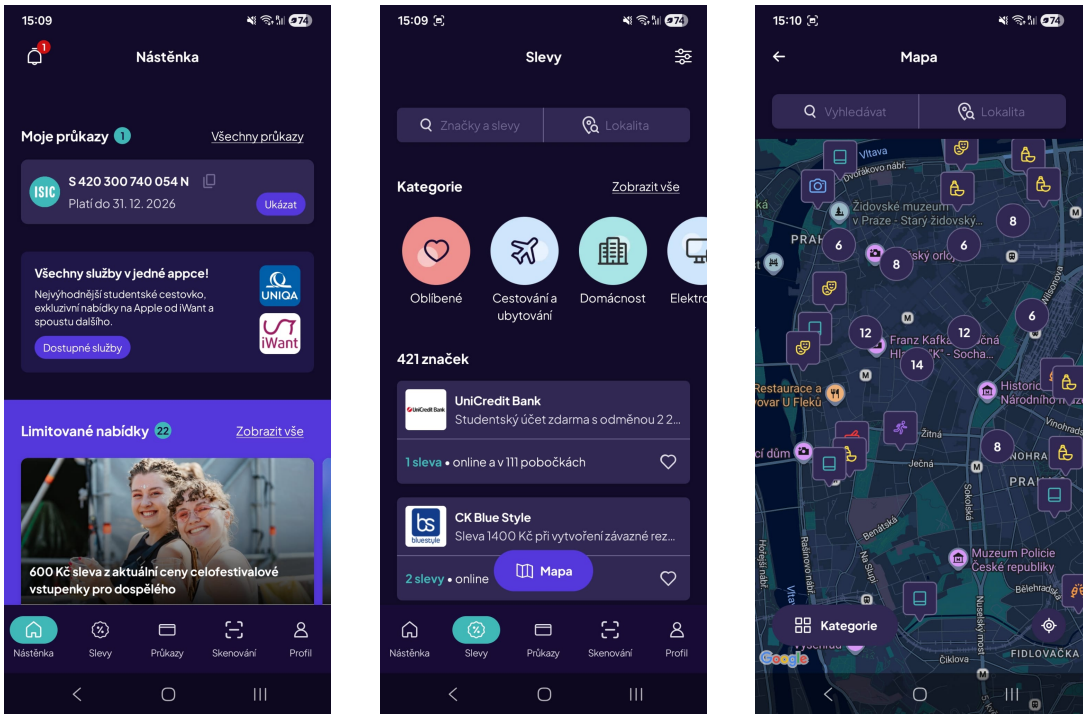
Z pohľadu študenta, ktorý chce takýmto spôsobom vyhľadať zľavu, má webová stránka ISIC viacero výhod. Predstavuje oficiálny a dôveryhodný zdroj, ktorý na jednom mieste sústreďuje ucelený prehľad partnerských zliav. Vďaka fulltextovému vyhľadávaniu a filtrovaniu podľa kategórií je jednoduché nájsť konkrétnu ponuku v prípade, že študent vopred vie, čo hľadá. Stránka je navyše dostupná z ľubovoľného zariadenia s webovým prehliadačom bez nutnosti inštalácie a rozlišuje medzi zľavami uplatniteľnými online a zľavami v kamenných prevádzkach.

Tento spôsob vyhľadávania má však aj podstatné nedostatky. Vyplývajú predovšetkým z toho, že ide o webovú stránku fungujúcu na princípe aktívneho vyhľadávania, teda o *pull* službu (viď podkapitolu 2.1.1). Študent musí stránku sám z vlastnej iniciatívy otvoriť a zľavu aktívne vyhľadať, stránka ho na dostupné ponuky sama neupozorní. V praxi to znamená, že o zľave v prevádzke, okolo ktorej práve prechádza, sa nedozvie, pokiaľ mu nenapadne ponuku vyhľadať. Hoci stránka obsahuje mapu a zoraďovanie podľa vzdialenosti, tieto funkcie vyžadujú manuálnu interakciu a nepracujú s priebežne aktualizovanou polohou zariadenia v reálnom čase. Chýba tak akékoľvek proaktívne a kontextové upozorňovanie na zľavy v bezprostrednom okolí používateľa.

Práve tieto nedostatky rieši navrhovaná aplikácia. Na rozdiel od webovej stránky funguje ako natívna mobilná aplikácia, ktorá priebežne sleduje polohu zariadenia a na princípe *push* služby aktívne upozorňuje používateľa na zľavy dostupné v jeho bezprostrednom okolí, a to aj v prípade, že ich sám aktívne nevyhľadáva. Tým sa zvyšuje pravdepodobnosť, že sa študent o dostupnej zľave vôbec dozvie a skutočne ju využije.

2.3.2 Alive App

Alive App je oficiálna mobilná aplikácia spojená s preukazom ISIC, ktorú si používatelia inštalujú na svoje mobilné zariadenie. Na rozdiel od webovej stránky ide o natívnu mobilnú aplikáciu, ktorá okrem samotného digitálneho preukazu poskytuje aj prehľad partnerských zliav vrátane ich zobrazenia na mape, a z pohľadu tejto práce preto predstavuje najbližšie existujúce riešenie. Hlavné obrazovky aplikácie sú znázornené na obrázku 1.



Obr. 1: Aplikácia Alive App: úvodná stránka, nástenka s digitálnym preukazom ISIC (vľavo), katalóg zliav s vyhľadávaním, kategóriami a obľúbenými (v strede) a mapa zliav v okolí (vpravo).

Úvodná obrazovka (nástenka) obsahuje odkaz na digitálny preukaz ISIC s jeho číslom a platnosťou, ktorý je možné priamo zobraziť a použiť na preukázanie nároku na zľavu. Okrem preukazu sa tu nachádza prehľad ďalších služieb, napríklad reklamné spolupráce s poskytovateľmi zliav a blok s časovo obmedzenými ponukami.

Zľavy sú dostupné v samostatnej sekcii, ktorá funguje ako katalóg partnerských značiek. V čase spracovania obsahoval 421 značiek. Ponuky je možné vyhľadávať podľa názvu značky, filtrovať podľa kategórií a označovať ako obľúbené. Pri každej značke je uvedené, či zľava platí online alebo v kamenných prevádzkach. Aplikácia zároveň umožňuje vyhľadávanie podľa lokality a zobrazenie zliav na mape, na ktorej sú prevádzky s ponukami vyznačené značkami podľa kategórie a zoskupené podľa počtu v danej oblasti.

Alive App tak predstavuje funkčne najbohatšie existujúce riešenie a priamo sa prekrýva s viacerými funkciami navrhovanej aplikácie – ponúka natívne mobilné rozhranie, katalóg zliav, obľúbené položky aj mapu. Jej ťažiskom však zostáva správa digitálneho preukazu a prehliadanie zliav na vyžiadanie. Vyhľadávanie podľa polohy aj mapa fungujú na princípe aktívneho vyhľadávania, teda *pull* služby – používateľ musí aplikáciu sám otvoriť a o zľavu sa aktívne zaujímať. Alive App síce zasiela aj notifikácie, tie sa však týkajú exkluzívnych ponúk a súťaží od obľúbených značiek, nie zliav v aktuálnom okolí používateľa. Aplikácia teda priebežne nevyhodnocuje polohu zariadenia a sama neupozorňuje na zľavy v bezprostrednom okolí používateľa. [5]

Práve v tomto spočíva odlíšenie navrhovaného riešenia. Zatiaľ čo notifikácie aplikácie Alive App upozorňujú na ponuky a súťaže vybraných značiek, navrhovaná aplikácia je zameraná primárne na proaktívne, polohovo podmienené upozorňovanie na princípe *push* služby: priebežne vyhodnocuje polohu používateľa a aktívne ho informuje o zľavách v jeho bezprostrednom okolí bez toho, aby ich musel sám vyhľadávať. Navrhovaná aplikácia tak nekonkuruje Alive App v úlohe digitálneho preukazu, ale dopĺňa medzeru v aktívnom objavovaní zliav podľa aktuálnej polohy používateľa.

2.4 Autentifikácia

Autentifikácia predstavuje proces overenia identity používateľa alebo systému, ktorý sa pokúša získať prístup k chráneným či súkromným zdrojom alebo službám. V kontexte webových a mobilných aplikácií ide o mechanizmus, ktorý zaručuje, že k danému rozhraniu alebo dátam pristupuje skutočne ten subjekt, ktorý má na to oprávnenie. Autentifikácia je základným predpokladom zabezpečeného prístupu a tvorí dôležitú súčasť každého systému pracujúceho s používateľskými účtami alebo citlivými informáciami.

V tradičných webových aplikáciách sa autentifikácia realizovala prevažne prostredníctvom relácií (angl. *sessions*), pri ktorých server po úspešnom prihlásení vytvoril a uložil reláciu na svojej strane a klientovi odovzdal identifikátor relácie uložený v cookie. Tento prístup je však pre moderné distribuované aplikácie a rozhrania REST menej vhodný, keďže si vyžaduje udržiavanie stavu na strane servera. Alternatívou, ktorá sa v súčasnosti bežne používa v kombinácii s bezstavovými REST rozhraniami, je autentifikácia pomocou tokenov. Token je kompaktný, samostatne stojaci objekt nesúci informácie o identite a oprávneniach používateľa, ktorý môže server overiť bez potreby uchovávať stav relácie.

2.4.1 JSON Web Token (JWT)

JSON Web Token (JWT) je otvorený štandard definovaný v špecifikácii RFC 7519 [6], ktorý opisuje kompaktný a bezpečný formát na prenos informácií (tzv. *claims*) medzi dvoma stranami vo forme JSON objektu. Tieto informácie môžu byť digitálne podpísané, čo umožňuje overiť ich autenticitu a integritu. JWT tokeny sú podpisované buď symetrickým algoritmom HMAC (napríklad HS256),

alebo asymetrickým algoritmom RSA či ECDSA. V prípade asymetrického podpisu je token podpísaný súkromným kľúčom a overiteľný verejným kľúčom, čo umožňuje jeho overenie aj tretími stranami bez potreby zdieľania tajného kľúča.

Štruktúra JWT sa skladá z troch častí oddelených bodkou a kódovaných pomocou Base64Url: hlavičky (*header*) s typom tokenu a podpisovým algoritmom, obsahu (*payload*) s tvrdeniami (*claims*) o používateľovi a podpisu (*signature*), ktorý zaručuje, že obsah nebol po vydaní zmenený. Tvrdenia sa delia na registrované (štandardizované, napríklad `iss`, `sub`, `aud`, `exp` a `iat`), verejné a súkromné (dohodnuté medzi konkrétnym vydavateľom a príjemcom, napríklad rola či oprávnenia používateľa).

Výhodou JWT tokenov v kontexte REST rozhraní je ich bezstavovosť – server nemusí udržiavať databázu aktívnych relácií, pretože všetky potrebné informácie sú obsiahnuté priamo v tokene a ich platnosť je možné overiť kryptograficky. Token môže byť prenášaný v HTTP hlavičke `Authorization` s prefixom `Bearer`, čo je v súlade s OAuth 2.0 štandardom [7].

2.4.2 Tokeny

V moderných autentifikačných systémoch postavených na JWT sa bežne pracuje s dvomi typmi tokenov: prístupovým tokenom (*access token*) a obnovovacím tokenom (*refresh token*). Ich kombinácia umožňuje dosiahnuť rovnováhu medzi bezpečnosťou a používateľským komfortom.

Prístupový token slúži na autorizáciu konkrétnych požiadaviek klienta voči chráneným endpointom API. Má krátku dobu platnosti, spravidla v rozsahu niekoľkých minút až hodín, čím sa minimalizuje potenciálna škoda v prípade jeho odcudzenia. Klient prikladá prístupový token ku každej požiadavke na chránenú URL adresu, server token overí a na základe informácií v jeho obsahu rozhodne, či je požiadavka oprávnená. Keďže JWT je bezstavový, overenie prebieha čisto na základe kryptografického podpisu a informácií v tokene bez nutnosti konzultovať databázu.

Obnovovací token plní odlišnú funkciu. Jeho úlohou je umožniť klientovi získať nový prístupový token po vypršaní jeho platnosti bez toho, aby sa používateľ musel znovu prihlasovať. Obnovovací token má dlhšiu dobu platnosti, typicky v rozsahu dní až týždňov, avšak musí byť uložený bezpečne a je spájaný s konkrétnym používateľom v databáze servera. Po predložení platného obnovovacieho tokenu server vydá nový prístupový token a voliteľne aj nový obnovovací token, čím dochádza k rotácii tokenov.

Rotácia obnovovacích tokenov je odporúčanou bezpečnostnou praktikou, pri ktorej sa každý obnovovací token po použití zneplatní a vydá sa nový. Tento mechanizmus znižuje riziko zneužitia odcudzeného obnovovacieho tokenu, pretože útočník by mohol token použiť len raz. Okrem toho je možné detekovať potenciálne zneužitie: ak server zaznamená pokus o použitie tokenu, ktorý bol už predtým zneplatnený, môže to indikovať únos tokenu a systém môže reagovať napríklad zneplatnením všetkých aktívnych tokenov daného používateľa.

Dôležitým aspektom správy tokenov je ich bezpečné ukladanie na strane klienta, tokeny by nemali byť uchovávané v otvorenej podobe. Na platforme Android sa na tento účel využíva *Android Keystore* [8], systémové, hardvérovo izolované úložisko kryptografických kľúčov. Samotné tokeny sa spravidla neukladajú priamo doň, ale sa šifrujú kľúčom uloženým v Android Keystore a v zašifrovanej podobe sa uchovávajú v súkromnom úložisku aplikácie, ktoré nie je prístupné iným aplikáciám.

2.4.3 Bezpečnosť

Autentifikácia na báze JWT si vyžaduje zohľadniť viacero bezpečnostných rizík. Najzávažnejším je odcudzenie tokenu – útočník s platným prístupovým tokenom môže konať v mene používateľa po celú dobu jeho platnosti. Toto riziko obmedzuje krátka doba platnosti tokenu a výhradné použitie šifrovaného protokolu HTTPS, ktoré bráni odpočúvaniu a útokom typu *man-in-the-middle*. Heslá sa v databáze nikdy neukladajú v čitateľnej podobe, ale pomocou hašovacích funkcií prispôsobených na heslá, ako je BCrypt [9], ktorých výpočtová náročnosť sťažuje útoky hrubou silou a slovníkové útoky pri prípadnom úniku databázy.

Server musí pri každom spracovaní tokenu dôsledne overovať podpis aj tvrdenia, inak hrozí akceptovanie cudzích alebo už expirovaných tokenov. Keďže obsah JWT je podpísaný, no nie šifrovaný a ľahko dekódovateľný, nesmie obsahovať citlivé údaje a mal by sa obmedziť len na identifikačné a autorizačné informácie. Bezstavové JWT navyše nemožno priamo zneplatniť pred expiráciou. Bežným riešením je viesť na serveri evidenciu vydaných obnovovacích tokenov (tzv. *allowlist*, resp. *blacklist* zneplatnených tokenov), pričom sa v databáze neukladá samotný token, ale iba jeho kryptografický haš, a pri odhlásení alebo detekcii zneužitia príslušný token zneplatniť, čím sa zabráni vydávaniu nových prístupových tokenov. Komplexný prehľad odporúčaní pri implementácii JWT poskytuje organizácia OWASP [10].

2.5 REST architektúra

REST (angl. *Representational State Transfer*) je architektonický štýl pre návrh distribuovaných systémov, ktorý definuje sadu obmedzení a princípov pre komunikáciu medzi klientom a serverom. Pojem REST zaviedol Roy Fielding vo svojej dizertačnej práci z roku 2000 [11], kde ho opísal ako abstrakciu architektúry webu. REST nie je protokol ani štandard, ale sada architektonických princípov, ktoré pri ich dodržiavaní vedú k škálovateľným, spoľahlivým a ľahko udržiavateľným systémom. Rozhrania navrhnuté v súlade s princípmi REST sa označujú ako RESTful API alebo jednoducho REST API.

RESTful API sa stali hlavným prístupom pri návrhu webových rozhraní pre mobilné aplikácie a webové služby. Ich popularita vyplýva z jednoduchosti, prehľadnosti a priamočiareho mapovania na štandard HTTP, čo uľahčuje ich implementáciu aj konzumáciu. V kontexte navrhovaného systému tvorí RESTful API komunikačnú vrstvu medzi mobilnou aplikáciou a serverovým backendom.

2.5.1 REST princípy

Architektonický štýl REST je definovaný šiestimi obmedzeniami (angl. *constraints*), ktorých dodržiavanie zaručuje vlastnosti spájané s REST architektúrou:

- *Jednotné rozhranie* (angl. *Uniform Interface*) – ústredný princíp REST: zdroje sú jednoznačne identifikované pomocou URI (napríklad `/api/users/42`) a klient s nimi pracuje prostredníctvom ich reprezentácií, najčastejšie vo formáte JSON.
- *Klient-server separácia* – klient (používateľské rozhranie) a server (uchovávanie a spracovanie dát) sa vyvíjajú a nasadzujú nezávisle, čo zvyšuje prenositeľnosť aj škálovateľnosť.
- *Bezstavovosť* (angl. *Statelessness*) – každá požiadavka nesie všetky informácie potrebné na svoje spracovanie a server neuchováva kontext medzi volaniami, čo zásadne uľahčuje škálovanie.
- *Ukladanie do vyrovnávacej pamäte* (angl. *Cacheability*) – odpovede sú explicitne označené ako (ne)ukladateľné do cache, čím sa znižuje záťaž servera aj doba odozvy.
- *Vrstvený systém* (angl. *Layered System*) – medzi klientom a serverom môžu byť sprostredkovatelia (proxy, load balancer, brána API) bez toho, aby o nich klient vedel.
- *Kód na vyžiadanie* (angl. *Code on Demand*) – server môže voliteľne rozšíriť klienta o spustiteľný kód, ide o jediné voliteľné obmedzenie, v praxi využívané zriedka.

2.5.2 HTTP metódy

REST API využívajú na komunikáciu protokol HTTP a štandardné HTTP metódy na vyjadrenie sémantiky operácií so zdrojmi. Správne použitie HTTP metód je kľúčové pre intuitívnosť a predvídateľnosť rozhrania.

Jednotlivé metódy vyjadrujú sémantiku operácií so zdrojmi: `GET` načíta reprezentáciu zdroja (bezpečná a idempotentná operácia), `POST` vytvára nový zdroj alebo vykoná akciu (nie je idempotentná), `PUT` nahradí zdroj a `DELETE` zdroj odstráni (idempotentná). Vhodné použitie metód nad prehľadnými URI vedie k intuitívnemu rozhraniu. Napríklad súbor operácií `GET /api/offers`, `GET /api/offers/{id}`, `POST /api/offers` a `DELETE /api/offers/{id}` tvorí prehľadné CRUD rozhranie pre správu zľavových ponúk.

HTTP protokol tiež definuje sadu stavových kódov, ktoré server používa na informovanie klienta o výsledku spracovania požiadavky. Stavové kódy sú rozdelené do tried: 2xx označuje úspech (napr. `200 OK`, `201 Created`, `204 No Content`), 4xx označuje chybu na strane klienta (napr. `400 Bad`

`Request`, `401 Unauthorized`, `404 Not Found`, `409 Conflict`) a 5xx označuje chybu na strane servera (napr. `500 Internal Server Error`). Správne používanie a rozlíšovanie stavových kódov je dôležité pre spoľahlivú komunikáciu medzi klientom a serverom.

2.5.3 Stavovosť vs. bezstavovosť

Kľúčovým rozhodnutím pri návrhu serverového rozhrania je voľba medzi stavovým a bezstavovým prístupom (*stateful* alebo *stateless*). V stavovom prístupe si server udržiava kontext komunikácie s klientom, typicky formou serverových relácií (*sessions*) identifikovaných pomocou cookie. To však komplikuje horizontálne škálovanie, ktoré si potom vyžaduje smerovanie klienta na tú istú inštanciu (*sticky sessions*) alebo zdieľané úložisko relácií. Bezstavový prístup, jeden zo základných princípov REST, naopak vyžaduje, aby každá požiadavka niesla všetky informácie potrebné na svoje spracovanie – v prípade autentifikácie teda platný token. JWT tokeny sú preň obzvlášť vhodné, keďže potrebné údaje nesú priamo vo svojom obsahu a ich overenie nevyžaduje externé úložisko. Bezstavovosť uľahčuje horizontálne škálovanie aj nasadenie v kontajneroch a zvyšuje odolnosť voči výpadkom, za cenu mierne vyšších požiadaviek, zložitejšej prípravy a správy životného cyklu tokenov na strane klienta.

2.6 Výber technológií

Voľba technológií pre navrhovaný systém vychádzala z jednej zastrešujúcej úvahy, a to zjednotiť celé riešenie okolo platformy .NET a jazyka C#. Keďže systém vyvíjal jeden autor a pozostáva z troch samostatných častí (mobilný klient, serverové API a služba na zber dát), spoločný jazyk a ekosystém výrazne znižujú réžiu vývoja: umožňujú zdieľať dátové modely aj pomocný kód, používať jednotné vývojové nástroje a sústrediť sa na jednu technologickú rodinu namiesto viacerých. Nasledujúce časti zdôvodňujú konkrétne voľby a porovnávajú ich s bežnými alternatívami.

2.6.1 Mobilný klient: .NET MAUI

Pre mobilného klienta bol zvolený multiplatformový framework .NET MAUI [12], nástupca Xamarin.Forms, ktorý umožňuje vyvíjať aplikácie pre Android aj iOS z jednej kódovej základne v jazyku C#. Alternatívou bol natívny vývoj (Kotlin či Java pre Android a Swift pre iOS), ktorý ponúka najlepší výkon, no za cenu samostatnej implementácie pre každý systém. Zvažované boli aj frameworky Flutter (Dart) a React Native (JavaScript), tie by však do projektu vniesli ďalší jazyk a ekosystém. Rozhodla jednotnosť s ostatnými časťami systému: rovnaký jazyk a zdieľanie dátových modelov so serverom, pričom .NET MAUI sprístupňuje natívne funkcie zariadenia (GPS, notifikácie, bezpečné úložisko) cez platformovo špecifický kód, čo je pre aplikáciu založenú na polohe nevyhnutné.

2.6.2 Serverová platforma a databáza: ASP.NET Core a PostgreSQL

Serverová časť je postavená na frameworku ASP.NET Core, ktorý je výkonný, multiplatformový (beží aj na operačnom systéme Linux, čo umožňuje jeho nasadenie do kontajnera) a má prvotriednu podporu pre tvorbu REST API, autentifikáciu tokenmi JWT aj prácu s databázou. Oproti alternatívam ako Node.js s frameworkom Express (JavaScript), Spring Boot (Java) či Django alebo FastAPI (Python) opäť rozhodla konzistentnosť s jazykom C# používaným naprieč celým systémom. [13]

Na perzistentné ukladanie dát bola zvolená relačná databáza PostgreSQL [14]. Dátový model systému je relačný – používatelia, poskytovatelia, ponuky, pobočky a ich vzťahy vrátane väzby M:N pre obľúbené položky – takže relačná databáza je preň vhodnejšia než dokumentové (NoSQL) riešenia typu MongoDB. Spomedzi relačných databáz bola PostgreSQL uprednostnená pred databázami MySQL/MariaDB a komerčným SQL Serverom pre jej otvorenosť a bezplatnosť, robustnosť, bohatú funkcionalitu (vrátane priestorových dotazov cez rozšírenie PostGIS, využiteľných pri prípadnej budúcej optimalizácii geolokácie) a kvalitnú podporu v prostredí .NET prostredníctvom poskytovateľa Npgsql [15] pre Entity Framework Core.

2.6.3 Automatizovaný zber dát: Playwright

Na automatizovaný zber dát z portálu ISIC bola zvolená knižnica Playwright [16], ktorá ovláda skutočný prehliadač (Chromium) v bezobrazovkovom režime. Toto rozhodnutie vyplynulo z povahy zdrojovej stránky: jej obsah so zľavami sa načítava dynamicky pomocou JavaScriptu, vrátane postupného dopĺňania záznamov cez tlačidlo „*Load more*“, takže jednoduché stiahnutie a parsovanie statického HTML by potrebné dáta vôbec nezískalo. Oproti staršiemu nástroju Selenium ponúka Playwright modernejšie a spoľahlivejšie automatické čakanie na načítanie prvkov aj jednoduchšie nastavenie, a na rozdiel od knižnice Puppeteer, určenej primárne pre prostredie Node.js, má oficiálnu podporu pre .NET, čím opäť zapadá do zvyšku systému.

Zber dát prebieha výhradne na nekomerčné, akademické a demonštračné účely v rámci tejto diplomovej práce. Služba je pritom navrhnutá ohľaduplne k zdrojovému portálu. Identifikuje sa vlastným reťazcom `User-Agent`, spúšťa sa len raz týždenne a v čase mimo špičky a medzi jednotlivými požiadavkami dodržiava časové odstupy, takže portál nadmerne nezaťažuje. Automatizovaný zber dát z cudzej webovej stránky však môže podliehať podmienkam používania prevádzkovateľa portálu a právnej ochrane obsahu. Pre reálne, najmä komerčné nasadenie by preto bolo potrebné získať súhlas prevádzkovateľa alebo využiť oficiálne rozhranie na poskytovanie dát.

2.6.4 Prístup k notifikáciám

Kľúčovou vlastnosťou navrhovanej aplikácie je proaktívne upozorňovanie na zľavy v okolí, ktoré bolo možné riešiť dvoma spôsobmi. Prvým je cloudová *push* infraštruktúra, typicky Firebase Cloud Messaging (FCM), pri ktorej server odosiela notifikácie cez službu tretej strany a tie sa doručia aj vtedy, keď aplikácia práve nebeží. Druhým sú lokálne notifikácie, ktoré zobrazuje samotná aplikácia počas svojho behu. Zvolený bol druhý prístup: keďže aplikácia tak či tak priebežne sleduje polohu zariadenia (a na to musí bežať aj na pozadí prostredníctvom služby typu *foreground service*), lokálne notifikácie sú postačujúce a zároveň sa vyhýbajú závislosti od externej služby, správe push tokenov aj nutnosti registrácie v platforme Firebase. Cenou za toto zjednodušenie je, že po úplnom ukončení alebo vynútenom zastavení aplikácie sa preruší sledovanie polohy a notifikácie sa prestanú doručovať. Toto obmedzenie sa podrobnejšie rozoberá vo výsledkoch práce. Ani prechod na serverom riadenú *push* infraštruktúru by ho však úplne neodstránil, keďže bez bežiacej aplikácie sa nezbiera aktuálna poloha a server by nedokázal určiť, o čom informovať.

2.6.5 Mapové podklady: Leaflet a OpenStreetMap

Na zobrazenie prevádzok na mape v okolí používateľa bola zvolená JavaScriptová knižnica Leaflet [17] s mapovými dlaždicami od projektu OpenStreetMap [18], vložená do webového pohľadu v aplikácii. Toto riešenie nevyžaduje žiadny API kľúč ani poplatky a nemá limity na počet zobrazení, na rozdiel od komerčných alternatív ako Google Maps alebo Mapbox, ktoré vyžadujú registráciu, fakturačné údaje a podliehajú kvótam. Leaflet zároveň umožňuje jednoducho vymeniť zdroj dlaždíc a ich tému, čo bolo neskôr využité pri implementácii tmavého režimu mapy.

3 Návrh systému

Aplikácia nesie názov *PerkAround*, ktorý je tvorený spojením dvoch anglických slov: *perk* (výhoda, benefit, zľava) a *around* (okolo, v okolí). Tento názov výstižne vyjadruje hlavnú myšlienku systému – sprístupniť študentom prehľad výhod a zliav dostupných v ich bezprostrednom okolí.

Návrh systému PerkAround vychádza z analýzy problematiky a existujúcich riešení uvedených v predchádzajúcej kapitole a tvorí základ pre následnú implementáciu. Postupne definuje funkčné a nefunkčné požiadavky, zvolenú architektúru a komunikáciu medzi jej vrstvami, databázový model, mechanizmus autentifikácie a napokon vizuálnu identitu a návrh používateľského rozhrania.

3.1 Funkčné požiadavky

Funkčné požiadavky opisujú konkrétne správanie systému – čo aplikácia musí umožniť používateľovi vykonať. Delí aplikáciu na približné sekcie, na základe ktorých sa implementovali konkrétne súčasti užívateľského rozhrania a funkcionality. Pri návrhu aplikácie PerkAround boli identifikované nasledujúce funkčné požiadavky.

Registrácia

Používateľ musí mať možnosť sa zaregistrovať pomocou e-mailovej adresy a hesla. Zvolené heslo musí spĺňať minimálne požiadavky na dĺžku a zloženie znakov. Záujemca o vytvorenie účtu musí súhlasiť s podmienkami používania aplikácie (na základe nefunkčnej požiadavky GDPR). Systém overí formát hesla a e-mailu a zabezpečí unikátnosť každého účtu. Po úspešnej registrácii je používateľ o tejto skutočnosti informovaný a je presmerovaný na prihlasovací formulár.

Prihlásenie

Prihlásenie prebieha zadaním e-mailu a hesla zadaného pri registrácii. Po úspešnom overení prihlasovacích údajov systém vydá prístupový JWT token a refresh token, ktoré sú uložené v bezpečnom úložisku zariadenia. Používateľ zostáva prihlásený až do uplynutia platnosti tokenov alebo do manuálneho odhlásenia. Systém tiež poskytuje riešenie pre prípad, kedy používateľ stratí možnosť prihlásiť sa do svojho účtu (napr. zabudne svoje heslo).

Zobrazenie ponúk podľa polohy

Domovská obrazovka aplikácie zobrazuje interaktívnu mapu so zvýraznenou aktuálnou polohou používateľa a zvýraznenými ponukami, ktoré ponúkajú študentské zľavy v jeho okolí. Okruh vyhľadávania je možné nastaviť v nastaveniach aplikácie. Každá prevádzka na mape je reprezentovaná markerom, po kliknutí naň sa zobrazia základné informácie o zľave.

Vyhľadávanie a filtrovanie ponúk

Sekcia *Ponuky* zobrazuje zoznam dostupných zliav. Používateľ má možnosť vyhľadávať podľa názvu poskytovateľa alebo filtrovať ponuky podľa určitej kategórie. Každá ponuka obsahuje detailné informácie vrátane podmienok využitia, adresy a platnosti zľavy.

Obľúbené ponuky

Používateľ môže označiť ľubovoľnú ponuku ako obľúbenú. Všetky takto označené ponuky sú dostupné v sekcii *Obľúbené*, kde má k nim rýchly prístup. Okrem toho môže používateľ v nastaveniach zvoliť, či chce dostávať notifikácie len pre obľúbené ponuky, alebo pre všetky v okolí.

Notifikácie

Systém posiela notifikácie o dostupných študentských zľavách v blízkom okolí používateľa. Notifikácie je možné personalizovať – používateľ si môže zvoliť, či chce byť upozorňovaný na zľavy zo všetkých ponúk alebo len tých, ktoré má uložené v obľúbených. Správu notifikácií (vrátane ich vypnutia) umožňuje sekcia *Nastavenia*.

Správa používateľského účtu

V sekcii *Môj profil* má používateľ prístup k správe svojho účtu. Môže tu upraviť svoje osobné údaje a preferencie alebo sa odhlásiť z aplikácie. Aplikácia taktiež umožňuje presmerovanie na aplikáciu Alive App za účelom zobrazenia digitálnej ISIC karty – používateľ tak nemusí opúšťať prostredie PerkAround pri preukazovaní nároku na zľavu.

3.2 Nefunkčné požiadavky

Nefunkčné požiadavky definujú kvalitu a obmedzenia systému – teda nie čo systém robí, ale akým spôsobom to robí. Pre aplikáciu PerkAround boli stanovené nasledujúce nefunkčné požiadavky.

Bezpečnosť

Všetka komunikácia medzi klientom a serverom musí byť šifrovaná prostredníctvom protokolu HTTPS. Používateľské heslá sú ukladané výhradne v hašovanej podobe. Autentifikácia je riešená pomocou JWT tokenov s obmedzenou dobou platnosti. Citlivé údaje uložené na zariadení (napr. tokeny) musia byť chránené mechanizmami operačného systému Android.

Výkonnosť

Vyhľadávanie zliav podľa polohy musí prebehnúť rýchlo, aby používateľ nezaznamenal výrazné oneskorenie. API endpointy by mali odpovedať do niekoľkých stoviek milisekúnd za bežných prevádzkových podmienok. Mapa musí byť responzívna a plynulá aj pri väčšom počte zobrazených markerov.

Škálovateľnosť

Backend musí byť navrhnutý tak, aby podporoval rast počtu používateľov a rozširovanie databázy prevádzok bez potreby zásadných architektonických zmien. Cloudová infraštruktúra a modulárny návrh API umožňujú horizontálne škálovanie v prípade zvýšenej záťaže.

Použiteľnosť

Aplikácia musí byť intuitívna a jednoduchá na ovládanie, vhodná pre používateľov bez hlbokých technických znalostí. Navigácia je realizovaná pomocou spodnej navigačnej lišty s piatimi kartami. Dizajn musí byť responzívny, prehľadný a konzistentný na rôznych veľkostiach obrazoviek Android zariadení.

3.2.1 Správa používateľských údajov podľa GDPR

Keďže aplikácia PerkAround spracúva osobné údaje (e-mailová adresa, poloha a voliteľné profilové údaje), musí byť v súlade s nariadením GDPR (angl. *General Data Protection Regulation*), právne záväzným v celej Európskej únii od 25. mája 2018. Z toho vyplýva požiadavka na zákonnosť a transparentnosť spracúvania, minimalizáciu zbieraných údajov, ich primerané zabezpečenie a rešpektovanie práv používateľa vrátane práva na výmaz účtu. Konkrétne kategórie údajov, právne základy spracúvania, príjemcov a práva používateľa rozoberá časť 4.3.12 venovaná implementácii. [19]

3.3 Architektúra systému

Systém PerkAround je navrhnutý ako trojvrstvová aplikácia skladajúca sa z mobilného klienta, serverového REST API a relačnej databázy. Jednotlivé vrstvy sú navzájom oddelené a komunikujú prostredníctvom jasne definovaných rozhraní, čo zabezpečuje ich nezávislosť, testovateľnosť a možnosť samostatného škálovania.

3.3.1 Klient-server model

Aplikácia využíva klasický klient-server model. Mobilný klient zasiela HTTP požiadavky na REST API, ktoré spracúva biznis logiku a pristupuje k databáze. Klient neobsahuje žiadnu priamu logiku pre prácu s databázou – všetky dáta získava výhradne prostredníctvom API volaní. Táto separácia umožňuje nezávislý vývoj oboch vrstiev a prípadnú výmenu ktorejkoľvek z nich bez nutnosti zásahu do ostatných.

Trojvrstvová architektúra sa skladá zo serverového backendu, mobilného frontendu a databázy. Backend je REST API v prostredí ASP.NET Core (.NET 8) organizované do vrstiev kontrolérov, služieb a dátového prístupu. Zodpovedá za autentifikáciu, správu prevádzok a ponúk, filtrovanie podľa GPS polohy aj notifikácie. Frontend je mobilná aplikácia postavená na frameworku .NET MAUI, cielená primárne na operačný systém Android. Logika obrazoviek je v ich code-behind

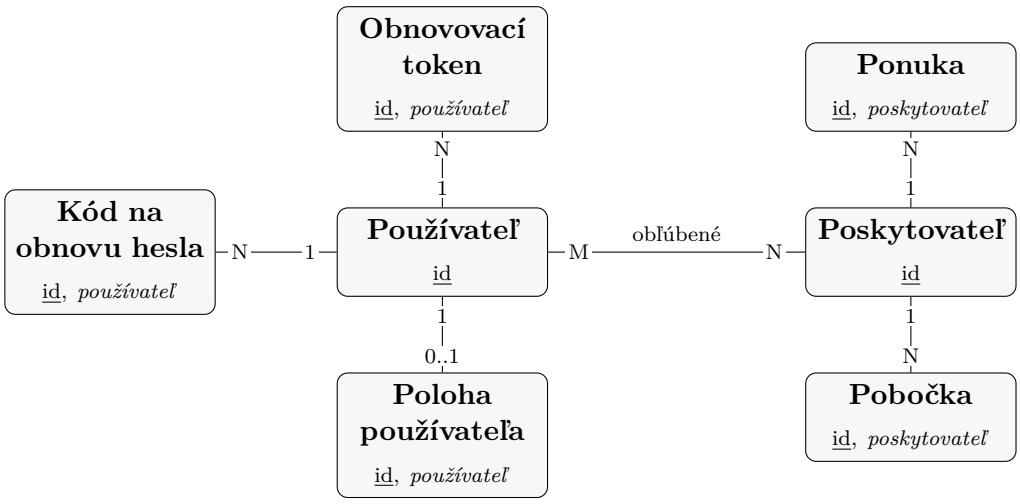
a aplikačná logika je vyčlenená do služieb sprístupnených cez vkladanie závislostí. Perzistenciu zabezpečuje relačná databáza PostgreSQL [14], ku ktorej backend pristupuje cez ORM Entity Framework Core prostredníctvom Npgsql [15]. Zdôvodnenie voľby týchto technológií je v časti 2.6, podrobnosti ich implementácie v kapitole 4.

3.4 Návrh databázového modelu

Návrh dátového modelu aplikácie zahŕňa hlavné entity, ich atribúty a vzťahy medzi nimi.

3.4.1 ER diagram

Konceptuálny dátový model tvorí niekoľko entít usporiadaných do dvoch logických vetiev, ktoré prepája vzťah obľúbených položiek. Prvou je používateľská a autentifikačná vetva sústredená okolo entity Používateľ, na ktorú sa viažu jeho obnovovacie tokeny, kódy na obnovu hesla a posledná známa poloha. Druhou je vetva obsahu okolo entity Poskytovateľ, ku ktorej patria jednotlivé zľavové ponuky a fyzické pobočky. Entity Používateľ a Poskytovateľ spája vzťah M:N vyjadrujúci označenie poskytovateľa ako obľúbeného. Model je znázornený na obrázku 2.



Obr. 2: Konceptuálny ER model databázy systému PerkAround. Podčiarknuté sú primárne kľúče, kurzívou vyznačené atribúty predstavujú cudzie kľúče vyjadrujúce daný vzťah.

Jeden používateľ môže mať viacero obnovovacích tokenov aj kódov na obnovu hesla (vzťah 1:N), pričom každému používateľovi prislúcha najviac jeden záznam o jeho poslednej polohe (1:0..1). Jeden poskytovateľ združuje viacero ponúk aj pobočiek (1:N). Vzťah medzi používateľom a poskytovateľom je M:N – používateľ môže označiť viacero poskytovateľov ako obľúbených a jeden poskytovateľ môže

byť obľúbený viacerými používateľmi. Tento vzťah M:N je v relačnom modeli realizovaný samostatnou väzobnou tabuľkou, ktorú spolu s ostatnými entitami bližšie opisuje nasledujúca časť.

3.4.2 Prehľad tabuliek

Jednotlivé entity dátového modelu sú nižšie opísané na logickej úrovni, teda aké údaje uchovávajú a ako navzájom súvisia. Každá entita má vlastný jednoznačný primárny kľúč („id“), ak nie je uvedené inak.

Používateľ

Reprezentuje registrovaného používateľa aplikácie. Uchováva prihlasovacie údaje (e-mailová adresa a hašované heslo), voliteľné profilové údaje (meno, priezvisko, dátum narodenia, profilová fotografia), rolu a preferencie notifikácií a newsletteru. Na používateľa sa viažu jeho obnovovacie tokeny, kódy na obnovu hesla, posledná známa poloha a zoznam obľúbených poskytovateľov.

Obnovovací token

Uchováva vydaný obnovovací token (na strane servera len v podobe jeho haša) spolu s časom jeho vytvorenia, expirácie a prípadného zrušenia. Každý token patrí práve jednému používateľovi (vzťah N:1) a slúži na obnovenie prihlásenia bez opätovného zadávania hesla.

Kód na obnovu hesla

Uchováva jednorazový kód na obnovu zabudnutého hesla, a to výhradne v hašovanej podobe, spolu s časom platnosti a príznakom použitia. Každý kód patrí jednému používateľovi (N:1).

Poloha používateľa

Uchováva poslednú známu polohu používateľa a čas jej aktualizácie. Každému používateľovi prislúcha najviac jeden takýto záznam (0..1:1).

Poskytovateľ

Reprezentuje obchodníka poskytujúceho študentskú zľavu. Uchováva názov, odkaz na zdrojovú stránku, logo, ilustračný obrázok a kategóriu. Jeden poskytovateľ združuje viacero ponúk a pobočiek.

Ponuka

Konkrétna zľavová ponuka patriaca práve jednému poskytovateľovi (N:1). Uchováva názov a popis ponuky.

Pobočka

Fyzická prevádzka poskytovateľa s geografickými súradnicami (zemepisná šírka a dĺžka). Každá pobočka patrí práve jednému poskytovateľovi (N:1) a práve pobočky sú predmetom vyhľadávania podľa polohy používateľa.

Obľúbený poskytovateľ

Väzobná entita realizujúca vzťah M:N medzi používateľom a poskytovateľom. Jej primárny kľúč je zložený z dvojice (používateľ, poskytovateľ), takže každú kombináciu možno uložiť najviac raz. Vyjadruje, ktorých poskytovateľov si používateľ označil ako obľúbených.

Okrem uvedených entít obsahuje databáza aj pomocnú tabuľku na evidenciu už rozoslaných dávok e-mailového newsletteru, ktorá zabezpečuje ich idempotentné rozosielanie a nie je súčasťou doménového modelu.

3.5 Návrh autentifikačného mechanizmu

Aplikácia spravuje vlastné používateľské účty (e-mail a heslo) namiesto delegovania na externého poskytovateľa identity, napríklad cez OAuth. Autentifikácia je navrhnutá ako kombinácia krátkodobých JWT *access* tokenov a dlhodobých *refresh* tokenov, ktorá vyvažuje bezpečnosť a používateľský komfort. Prístupový token má platnosť 15 minút, obnovovací token 7 dní a jeho haš sa uchováva v databáze servera, čo umožňuje jeho okamžité zneplatnenie. Obnovovacie tokeny sa pri každej obnove rotujú – predložený token sa zneplatní a nahradí novým s čerstvou sedemdňovou platnosťou, čím vzniká priebežné posúvanie expirácie (tzv. sliding expiration) a zároveň možnosť odhaliť znovupoužitie odcudzeného tokenu. Na strane klienta sú tokeny uložené v Android Keystore, na strane servera v databáze výhradne ako ich kryptografický haš. Konkrétna realizácia tohto návrhu vrátane rotácie, detekcie znovupoužitia a obnovy zabudnutého hesla je opísaná v časti 4.2.

Detekcia znovupoužitia tokenu predstavuje kompromis medzi bezpečnosťou a používateľským komfortom. Ak sa pri nestabilnom mobilnom pripojení stratí odpoveď servera a klient zopakuje požiadavku o obnovu s už zrotovaným obnovovacím tokenom, systém to vyhodnotí rovnako ako pokus o zneužitie a odhlási používateľa zo všetkých zariadení. Server totiž nedokáže spoľahlivo odlíšiť takéto neúmyselné zopakovanie požiadavky od skutočného znovupoužitia odcudzeného tokenu. Návrh preto vedome uprednostňuje bezpečnosť pred komfortom, keďže ide o zriedkavú situáciu, ktorej jediným dôsledkom je opätovné prihlásenie používateľa.

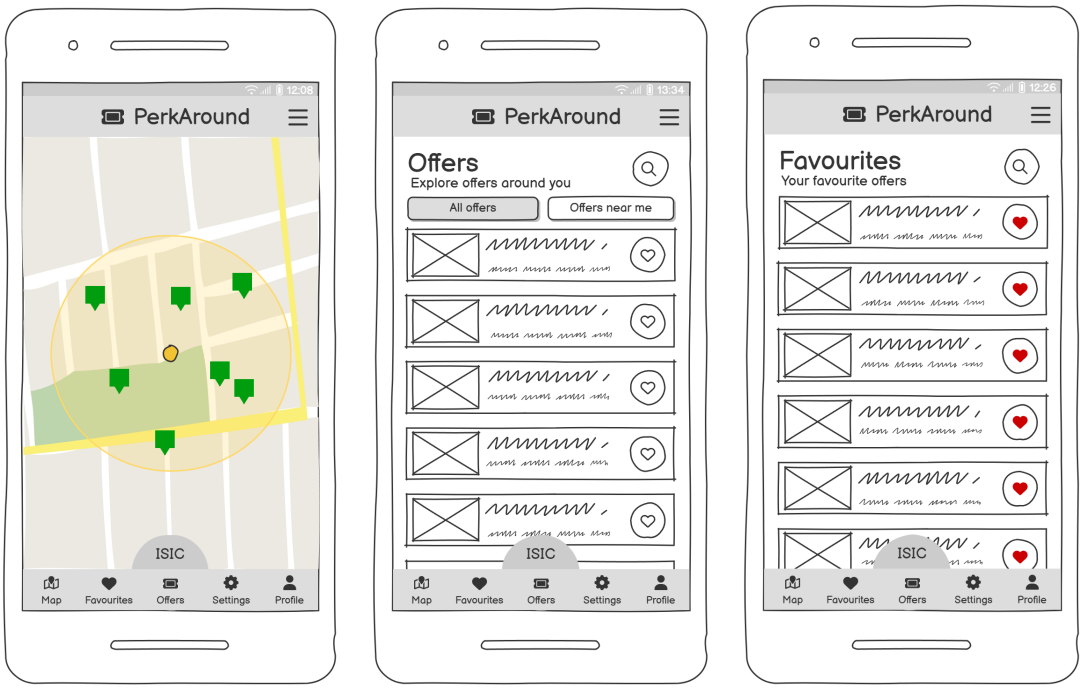
3.6 Návrh používateľského rozhrania

Pred samotnou implementáciou bolo rozhranie aplikácie navrhnuté vo forme wireframov, teda jednoduchých nízkoúrovňových náčrtov rozloženia jednotlivých obrazoviek. Návrh vychádza z definovaných funkčných požiadaviek: aplikácia je členená do piatich hlavných kariet dostupných zo spodnej navigačnej lišty (Map, Favourites, Offers, Settings, Profile), pričom v hornej lište je umiestnené logo a tlačidlo rozbaľovacej ponuky. Cieľom wireframov bolo overiť rozmiestnenie prvkov a tok obrazoviek ešte pred písaním kódu.

3.6.1 Wireframy

Wireframy boli vytvorené v nástroji Balsamiq[3], ktorý zámerne používa štýl imitujúci kresbu voľnou rukou. Táto skicová estetika je pri nízkoúrovňovom návrhu výhodná, pretože upriamuje pozornosť na rozloženie a funkciu prvkov namiesto detailov vizuálneho štýlu (farby, presné písmo), ktoré sa riešia až neskôr.

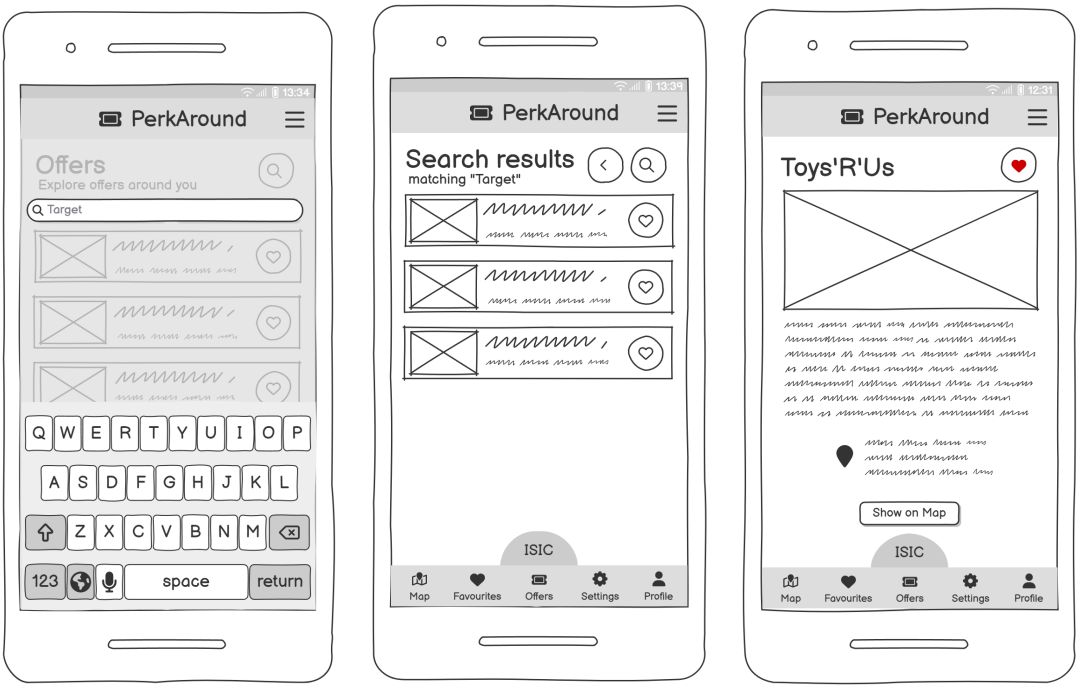
Navrhnuté wireframy boli počas tvorby predložené trom ľuďom mimo oblasti informačných technológií a na základe ich pripomienok a preferencií bol návrh postupne upravovaný tak, aby bol čo najviac univerzálne a intuitívne používateľsky prívetivý. Tento neformálny používateľský test pomohol odhaliť nejasnosti v rozložení a pomenovaní prvkov skôr, než sa premietli do implementácie.



Obr. 3: Wireframy hlavných obsahových kariet: mapa s okolím (vľavo), zoznam ponúk (v strede) a obľúbené prevádzky (vpravo).

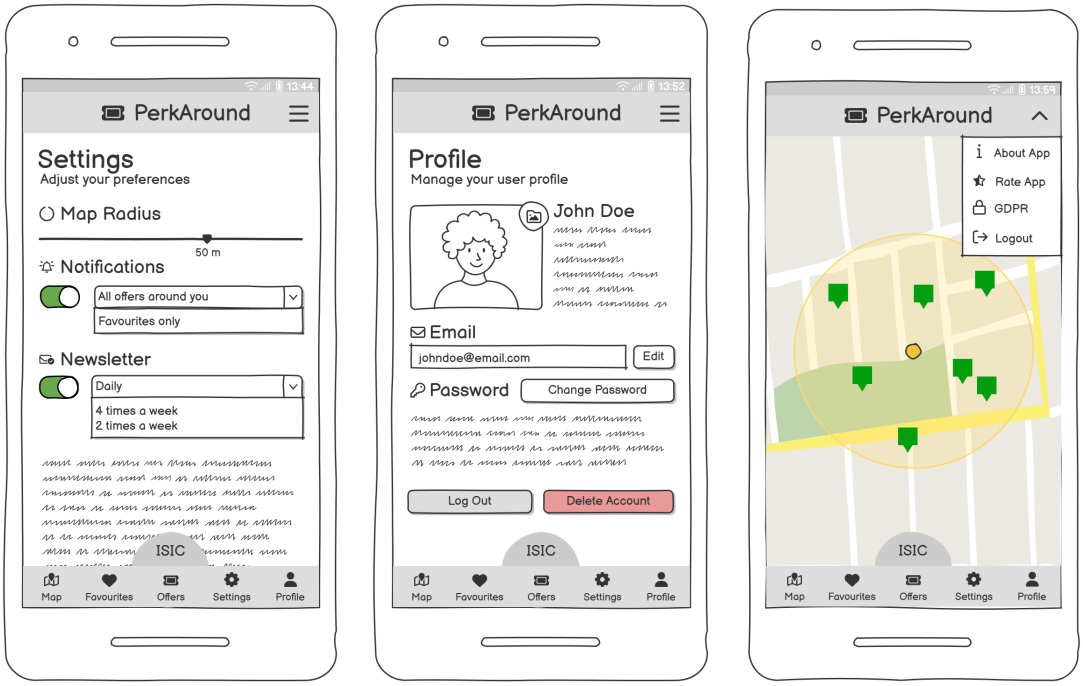
Obrázok 3 znázorňuje tri hlavné obsahové karty. Mapa (vľavo) zvýrazňuje aktuálnu polohu používateľa a okruh vyhľadávania, v ktorom sú značkami vyznačené prevádzky s ponukami. Karta ponúk (v strede) obsahuje nadpis, dvojicu prepínačov (všetky ponuky a ponuky v okolí), tlačidlo vyhľadávania a zoznam kariet, kde každá karta má náhľadový obrázok, názov, popis a ikonu srdca. Obľúbené (vpravo) majú rovnaké rozloženie kariet, pričom všetky sú označené ako obľúbené. Vo všetkých obrazovkách je prítomná horná lišta s logom a ponukou, tlačidlo ISIC vedúce k ISIC preukazu a spodná navigačná lišta.

[3]Oficiálne stránky: https://balsamiq.cloud/



Obr. 4: Wireframy vyhľadávania a detailu: zadávanie hľadaného výrazu (vľavo), výsledky vyhľadávania (v strede) a detail prevádzky (vpravo).

Obrázok 4 znázorňuje vyhľadávanie a detail ponuky. Po klepnutí na vyhľadávanie sa zobrazí vyhľadávacie pole a klávesnica (vľavo). Po zadaní výrazu sa zobrazia zodpovedajúce výsledky (v strede). Klepnutím na ktorúkoľvek prevádzku sa otvorí jej detail (vpravo) s názvom, obrázkom, popisom, adresou a možnosťou zobrazenia na mape.



Obr. 5: Wireframy účtu a nastavení: nastavenia (vľavo), profil používateľa (v strede) a rozbaľovacia ponuka (vpravo).

Obrázok 5 obsahuje nastavenia, profil a rozbaľovaciu ponuku. Nastavenia (vľavo) umožňujú upraviť okruh vyhľadávania posuvníkom a zvoliť rozsah notifikácií a frekvenciu newslettera. Profil (v strede) zobrazuje údaje účtu (fotografia, meno, e-mail) spolu s akciami ako zmena hesla, odhlásenie či zmazanie účtu. Rozbaľovacia ponuka (vpravo) sprístupňuje sekundárne akcie – informácie o aplikácii, hodnotenie, zásady ochrany osobných údajov a odhlásenie.

Výsledná implementácia z týchto wireframov vychádza. Niektoré prvky boli počas vývoja spresnené (napríklad pomenovanie prepínačov ponúk či akcie v detaile prevádzky), no celková štruktúra obrazoviek a navigácie zostala zachovaná.

3.7 Vizuálna identita aplikácie

Vizuálnu identitu aplikácie PerkAround tvorí jej logo a ikona, farebná schéma, typografia a ďalšie dizajnové rozhodnutia stojace za vizuálnou podobou výsledného projektu.

3.7.1 Logo a ikona aplikácie

Logo aplikácie sa skladá z obrazového symbolu a slovného označenia *PerkAround*. Symbol tvorí dvojica lístkov (kupónov), ktorá odkazuje na zľavy a poukazy, teda na hlavný účel aplikácie. Slovné označenie je vysádzané zaobleným bezpätkovým písmom *Coolvetica Regular*[4], ktoré pôsobí moderne a priateľsky.

[4]Zdroj: https://www.dafont.com/coolvetica.font

Názov *PerkAround* vznikol spojením anglických slov *perk* (výhoda, benefit) a *around* (v okolí), čím vystihuje hlavnú myšlienku aplikácie – sprístupniť výhodné ponuky – zľavy dostupné v bezprostrednom okolí používateľa.



Obr. 6: Logo aplikácie PerkAround.

Ikona aplikácie (spúšťacia ikona na ploche zariadenia) vychádza z rovnakého symbolu lístkov umiestneného na fialovo-modrom pozadí, čím zachováva jednotný vizuálny jazyk s logom. Samotný symbol lístkov je založený na voľne dostupnej ikone z knižnice Flaticon[5].



Obr. 7: Ikona aplikácie PerkAround.

3.7.2 Farebná schéma

Vizuálnu identitu aplikácie určuje predovšetkým jej farebná paleta. Dominantnou farbou je sýta fialová (`#8E24AA`), ktorá sa uplatňuje na hlavných prvkoch rozhrania, teda na hornej a dolnej navigačnej lište, primárnych tlačidlách a zvýrazneniach. Dopĺňajú ju tmavší (`#6A1B9A`) a svetlejší (`#CE93D8`) odtieň fialovej, ktoré sa používajú na aktívne stavy, stlačené tlačidlá a jemné akcenty, napríklad na pozadia štítkov kategórií.

Doplnkovou akcentovou farbou je tyrkysová (`#40B8B8`), ktorá je vyhradená pre prvky spojené s preukazom ISIC, napríklad pre plávajúce tlačidlo ISIC a tlačidlo „*View on ISIC*“ v detaile ponuky. Týmto farebným odlíšením sa vizuálne prepája funkcionalita naviazaná na ISIC. Prihlasovacie obrazovky používajú výrazný diagonálny prechod z fialovej (`#8E24AA`) do modrej (`#2979FF`), ktorý dodáva úvodu do aplikácie dynamický dojem. Okrem hlavných, tzv. *brandových* farieb používa aplikácia aj sémantické (stavové) farby, ktoré sa uplatňujú

[5]Zdroj: https://www.flaticon.com/free-icon/park-tickets-couple_53756

najmä vo výstražných pruhoch v hornej časti obrazovky a pri spätnej väzbe: zelenú (`#2E7D32`) pre potvrdenia a oznámenie o obnovení pripojenia, oranžovú (`#F57C00`) pre upozornenie na vypnuté sledovanie polohy a červenú (`#D32F2F`) pre stav offline a deštruktívne akcie (napríklad odhlásenie či zmazanie účtu). Značkové farby sú zhrnuté na obrázku 8. Ich technickú realizáciu vrátane farieb plôch, textu a podpory tmavého režimu opisuje časť 4.3.3 v kapitole o implementácii.



Obr. 8: Farebná paleta aplikácie PerkAround.

3.7.3 Typografia

Používateľské rozhranie aplikácie využíva bezpätkové písmo *Open Sans*, a to v reze Regular pre bežný text a Semibold pre zvýraznenia a nadpisy. Open Sans je humanistický bezpätkový font s vysokou čitateľnosťou aj pri malých veľkostiach, čo je vhodné pre mobilné rozhranie zobrazované na menších obrazovkách. Ukážka písma je nasledovná:

> Open Sans
> **Nearby offers** - Discover perks around you!

Text „PerkAround“ v logu aplikácie je vysádzané zaobleným bezpätkovým písmom *Coolvetica Regular*. Ako už bolo zdôraznené, jeho minimalistický a moderný charakter odlišuje značku od bežného textu rozhrania a dodáva logu osobitý vzhľad, pričom zároveň ladí s humanistickým tvaroslovím písma Open Sans použitého v používateľsom rozhraní.

3.7.4 Tvorba grafických podkladov

Všetky vlastné grafické podklady aplikácie – logo, ikonu aj ďalšie vizuálne prvky rozhrania – boli vytvorené v open-source grafickom editore GIMP (*GNU Image Manipulation Program*)[6]. GIMP je bezplatný rastrový grafický editor, ktorý svojimi možnosťami zodpovedá bežným profesionálnym nástrojom na úpravu obrázkov – podporuje prácu s vrstvami, masky, filtre aj export do rôznych formátov. Použitie voľne dostupného nástroja umožnilo vytvoriť konzistentnú vizuálnu identitu bez potreby zakúpenia licencie komerčného softvéru.

[6]Oficiálne stránky: `https://www.gimp.org`

4 Implementácia riešenia

Implementácia systému PerkAround prebiehala po jednotlivých častiach. Postupne sú opísané serverová časť (backend) a automatizovaný zber dát, autentifikačný mechanizmus, mobilný klient, komunikácia medzi klientom a serverom, spracovanie georgrafickej polohy a ostatné aspekty systému. Pri každej časti sú uvedené použité technológie a kľúčové implementačné rozhodnutia.

4.1 Implementácia backendu

Backend aplikácie PerkAround je implementovaný ako REST API postavené na platforme ASP.NET Core 8.0. Zodpovedá za spracovanie HTTP požiadaviek od mobilného klienta, biznis logiku, prístup k databáze a správu autentifikácie. Kód je organizovaný do viacerých vrstiev, čo zabezpečuje prehľadnosť, testovateľnosť a jednoduchú rozšíriteľnosť projektu. Backend repozitár som nazval `DiscountAPI`.

4.1.1 Konfigurácia

Aplikácia využíva ASP.NET Core 8.0 s konfiguráciou v súbore `Program.cs`. Pri štarte sa automaticky vykonajú databázové migrácie, čím sa zaručí, že schéma databázy je vždy aktuálna bez nutnosti ručného zásahu. Konfigurácia zahŕňa registráciu JWT Bearer autentifikácie s algoritmom HMAC SHA-256, napojenie na PostgreSQL databázu cez Entity Framework Core s logikou opakovaného pokusu o pripojenie (až 12 pokusov s exponenciálnym oneskorením), a Swagger rozhranie s podporou autorizácie pre potreby testovania API.

4.1.2 Štruktúra projektu

Projekt je rozdelený do nasledujúcich adresárov:

- `Controllers/` – triedy obsluhujúce HTTP požiadavky a definujúce API endpointy.
- `Services/` – biznis logika oddelená od kontrolérov, obsahuje pomocné triedy pre prácu s tokenmi a geolokáciou.
- `Models/` – doménové entity mapované na databázové tabuľky pomocou EF Core.
- `DTOs/` – dátové prenosové objekty definujúce vstupný a výstupný formát API volaní.
- `Data/` – databázový kontext `AppDbContext` s konfiguráciou vzťahov a indexov.
- `Migrations/` – automaticky generované EF Core migrácie databázovej schémy.

4.1.3 Dátové modely a databázový kontext

Perzistentná vrstva je postavená na frameworku Entity Framework Core [20]. Doménové entity sú definované ako obyčajné triedy v adresári `Models/` a databázový kontext `AppDbContext` dedí od triedy `DbContext`. Pre každú entitu vystavuje množinu `DbSet<T>` (sety `Users`, `RefreshTokens`, `Providers`, `Offers`, `Branches`, `UserFavoriteProviders`, `UserLocationCaches`, `UserBranchNotifications`, `PasswordResetCodes` a `NewsletterDispatches`). Mapovanie na tabuľky vychádza z konvencií EF Core, pričom kľúče, indexy a vzťahy sú spresnené pomocou Fluent API v metóde `OnModelCreating`. Nasledujúci prehľad dopĺňa logický model z kapitoly návrhu o konkrétne stĺpce, dátové typy a konfiguráciu.

Stĺpce entít vychádzajú z logického modelu, nižšie sú zhrnuté len tie s netriviálnym riešením, bežné metadáta (`CreatedAt`, `UpdatedAt` a pod.) sa neopakujú. Entita `User` (primárny kľúč `Id` typu `Guid`) okrem prihlasovacích (`Email` s unikátnym indexom, `PasswordHash`) a profilových údajov nesie aj preferencie notifikácií a newsletteru a navigáciu na svoje `RefreshTokens`. Entita `RefreshToken` neukladá token v otvorenej podobe, ale iba jeho `TokenHash` (SHA-256), spolu s `ExpiresAt`, `RevokedAt` a `ReplacedByTokenHash` (haš tokenu, ktorý ho pri rotácii nahradil), vypočítaná vlastnosť `IsActive` indikuje nezrušený a neexpirovaný token.

Poskytovateľa, ponuku a pobočku (`Provider`, `Offer`, `Branch`) spája vzťah s kaskádovým mazaním. `Provider` má unikátny index nad `Name` aj `DetailUrl`, zatiaľ čo `Branch` okrem GPS súradníc uchováva aj predpočítané súradnice mriežky `TileX` a `TileY` s kompozitným indexom, ktorý urýchľuje geolokačné dotazy. Väzobná tabuľka `UserFavoriteProvider` má zložený kľúč (`UserId`, `ProviderId`) a kaskádové mazanie na obe strany. Entity `UserLocationCache` a `UserBranchNotification` identifikujú používateľa reťazcovým `UserId` (hodnota tvrdenia `NameIdentifier`) a slúžia na vyrovnávaciu pamäť polohy a evidenciu už odoslaných upozornení. `PasswordResetCode` ukladá výhradne `CodeHash` (nie samotný kód pre reset hesla), zatiaľ čo `NewsletterDispatch` má nad dvojicou (`Kind`, `ScheduledDate`) unikátny index zaručujúci idempotentné rozoslanie dávky.

Konfigurácia modelu v metóde `OnModelCreating` definuje unikátne indexy (`User.Email`, `Provider.Name`, `Provider.DetailUrl` a dvojica `NewsletterDispatch.Kind` a `ScheduledDate`), kompozitný index nad `Branch.TileX` a `TileY` a vzťahy s kaskádovým mazaním (`Offer` aj `Branch` na `Provider`, `UserFavoriteProvider` na `User` aj `Provider`, `PasswordResetCode` na `User` a `UserBranchNotification` na `Branch`). Časové stĺpce `CreatedAt` a `UpdatedAt` sú inicializované na aktuálny čas UTC a výčtové typy `NotifyScope` a `NewsletterFrequency` sa do databázy ukladajú ako celé čísla.

4.1.4 Kontroléry

Kontroléry dedia od triedy `ControllerBase`, sú označené atribútom `[ApiController]` a využívajú atribútové smerovanie. Väčšina endpointov vyžaduje platný prístupový token (atribút `[Authorize]`), výnimkou sú verejné operácie ako prihlásenie alebo obnova hesla. API obsahuje osem kontrolérov:

- `AuthController` (`api/auth`) – registrácia (`POST register`), prihlásenie (`POST login`), obnovenie prístupového tokenu (`POST refresh`), odhlásenie (`POST logout`) a obnova zabudnutého hesla (`POST forgot-password` a `reset-password`). Endpointy sú verejné a heslá sa hašujú algoritmom BCrypt [9].
- `UsersController` (`api/users`) – správa profilu prihláseného používateľa: načítanie údajov (`GET me`), úprava mena, priezviska a dátumu narodenia (`PUT me`), zmena e-mailu (`PUT me/email`) a hesla (`PUT me/password`), nahranie profilovej fotografie (`POST me/avatar`), zmazanie účtu (`DELETE me`) a čítanie i uloženie preferencií notifikácií a newsletteru (`GET` a `PUT me/notifications`). Vyžaduje autorizáciu.
- `LocationController` (`api/location`) – prijíma aktuálnu polohu klienta (`POST update`), na základe geomriežky a vyrovnávacej pamäte polohy identifikuje blízke pobočky s aktívnymi ponukami a vyhodnocuje odoslanie notifikácií. Endpoint `GET all` vráti všetkých poskytovateľov spolu s ich najbližšou pobočkou. Vyžaduje autorizáciu.
- `ProvidersController` (`api/providers`) – vracia detail konkrétneho poskytovateľa vrátane jeho ponúk (`GET {id}`). Vyžaduje autorizáciu.
- `ProvidersImportController` (`api/providers/import`) – zabezpečuje hromadný import poskytovateľov, pobočiek a ponúk z webscrapingovej služby. Import je transakčný a obsahuje detekciu zmien: nové záznamy sa vložia, zmenené aktualizujú a tie, ktoré už v zdroji nie sú, sa odstránia.
- `FavoritesController` (`api/favorites`) – správa obľúbených poskytovateľov: zoznam identifikátorov na rýchlu synchronizáciu (`GET ids`), plné karty s najbližšou pobočkou (`GET`), pridanie (`POST {providerId}`) a odobranie (`DELETE {providerId}`). Vyžaduje autorizáciu.
- `AppUpdateController` (`api/app`) – verejný endpoint `GET version` vracajúci informácie o najnovšej verzii aplikácie (číslo verzie, adresa súboru APK a zoznam zmien) pre mechanizmus vnútornej aktualizácie.
- `NewsletterController` (`api/newsletter`) – verejný endpoint `GET unsubscribe` na odhlásenie z odberu newsletteru pomocou tokenu z e-mailu a chránený endpoint `POST test` na manuálne otestovanie rozoslania.

4.1.5 Služby

Biznis logika nezávislá od HTTP vrstvy je umiestnená v adresári `Services/`:

- `TokenService` – generuje prístupové JWT tokeny podpísané algoritmom HMAC SHA-256 (s tvrdeniami `sub`, `email` a `role` a predvolenou platnosťou 15 minút) a kryptograficky bezpečné obnovovacie tokeny (64-bajtový náhodný reťazec kódovaný v Base64).
- `GeoHelper` – vypočítava reálnu vzdialenosť dvoch GPS súradníc pomocou Haversínovho vzorca s polomerom Zeme 6 371 000 m.
- `GridGeoService` – mapuje GPS súradnice na diskrétne dlaždice mriežky s rozlíšením 100 m × 100 m (`GetTile`) a vyhľadáva susedné dlaždice v zadanom polomere (`GetNeighborTiles`). Tento prístup nahrádza nákladné výpočty vzdialenosti pre každú pobočku efektívnym indexovaným dotazom do databázy.
- `EmailService` – rozhranie `IEmailSender` a jeho implementácia `SmtpEmailSender`, ktorá odosiela e-maily (uvítací e-mail po registrácii, kód na obnovu hesla, newsletter) cez SMTP server.
- `EmailTemplates` – pomocná trieda na načítanie HTML šablón e-mailov z priečinka `Templates` a vloženie absolútnej adresy loga do šablóny.
- `NewsletterComposer` – zostavuje personalizovaný obsah e-mailového newsletteru (`ComposeAsync`) a narodeninového pozdravu (`ComposeBirthdayAsync`) na základe poslednej polohy a preferencií používateľa.
- `NewsletterBackgroundService` – dlhodobo bežiaca služba typu `BackgroundService`, ktorá v pravidelných intervaloch rozosiela newsletter (denný, týždenný a mesačný) a narodeninové pozdravy. Podrobnejšie je opísaná v časti venovanej newsletteru.

4.1.6 Automatizovaný zber dát

Súčasťou backendovej infraštruktúry je služba postavená ako samostatný projekt `IsicScraper`, ktorej úlohou je automatizovane zbierať aktuálne dáta (ďalej *webscraping*) o prevádzkach a zľavových ponukách z webovej stránky ISIC[7] a synchronizovať ich do databázy aplikácie PerkAround. Ide o .NET 8 Worker Service – dlhodobo bežiaca služba implementujúca rozhranie `BackgroundService`, ktorá beží nepretržite na serveri a webscraping spúšťa pravidelne každý pondelok o 03:00 hodine nadránom, kedy sa prepokladá minimálna záťaž systému. Dĺžka trvania webscrapingu je úmerne závislá na počte prevádzok umiestnených na portále ISIC. Tento čas tiež ovplyvňujú aj

[7] https://www.isic.cz/en/discounts

ručne nastavené časové odstupy medzi jednotlivými akciami, aby služba nebola stránkou chybne vyhodnotená ako *Denial of Service* útok a zablokovaná.

Využíva knižnicu Playwright [16] na automatizáciu Chromium prehliadača v bezobrazovkovom (tzv. *headless*) režime. Playwright umožňuje plnohodnotnú interakciu so stránkou vrátane klikania na prvky, čakania na dynamicky načítaný obsah a extrahovania DOM štruktúry. Pre komunikáciu s REST API backendu je použitý `HttpClient` z frameworku .NET.

Pri každom spustení prebehne nasledujúci postup:

1. Spustí sa Chromium prehliadač v headless režime s vlastným User-Agent reťazcom identifikujúcim aplikáciu ako výskumného bota pre účely akademického projektu.

2. Služba načíta zoznam prevádzok na stránke ISIC portálu a v slučke kliká na tlačidlo „*Load more*“, čím sa načítajú všetky dostupné záznamy.

3. Pre každú ponuku sa otvorí jej detailná stránka, z ktorej sa extrahujú: názov ponuky, URL titulného obrázka, URL loga, zoznam poskytovaných zliav (nadpis a popis) a GPS súradnice pobočiek.

4. Získané dáta sú voliteľne exportované do CSV súborov s časovou pečiatkou pre potreby zálohovania a auditovateľnosti.

5. Nakoniec sú dáta odoslané ako JSON cez HTTP POST na endpoint `/api/providers/import` backendu, kde sú transakčne spracované a uložené do databázy.

Projekt webscrapingu je členený do troch logických celkov: `Worker` (orchestrátor celého cyklu), `IsicCrawler` (logika webscrapingu, prechádzanie webovej stránky a parsovanie DOM) a `CsvExporter` (voliteľný export zozbieraných dát do CSV súborov). Dátové modely `ProviderDto`, `OfferItemDto` a `BranchDto` zodpovedajú štruktúre, ktorú očakáva importný REST API endpoint backendu.

Import na strane backendu (`ProvidersImportController`) je navrhnutý ako idempotentný – pri každom importe sa porovnávajú prichádzajúce dáta s existujúcimi záznamami v databáze. Nové prevádzky sa vložia, zmenené sa aktualizujú a záznamy bez zmeny zostanú nedotknuté. Tým sa predchádza duplicitám a zbytočným zápisom do databázy. Naopak, prevádzky, ktoré už zľavy neposkytujú, sú z databázy odstránené.

4.2 Implementácia autentifikácie

Autentifikácia v aplikácii PerkAround je postavená na kombinácii JWT *access* tokenov a *refresh* tokenov uložených v databáze. Celý mechanizmus je implementovaný v `AuthController` a `TokenService` projektu `DiscountAPI`.

4.2.1 JWT konfigurácia

JWT autentifikácia je nakonfigurovaná v súbore `Program.cs` pomocou `AddAuthentication` a `AddJwtBearer` [21], pričom predvolenou schémou pre autentifikáciu je `JwtBearerDefaults.AuthenticationScheme`. Parametre `TokenValidationParameters` majú zapnuté overovanie vydavateľa, príjemcu, doby platnosti aj podpisu (`ValidateIssuer`, `ValidateAudience`, `ValidateLifetime` a `ValidateIssuerSigningKey`). Očakávaným vydavateľom je „discount-api“ a príjemcom „discount-client“. Podpisovací kľúč sa načíta z konfigurácie (`Jwt:Key`) a obalí do `SymmetricSecurityKey`.

4.2.2 TokenService

Trieda `TokenService` zabezpečuje generovanie oboch typov tokenov a ich doby platnosti číta z konfigurácie (sekcia `Jwt`), takže v kóde nie sú natvrdo zapísané žiadne časové konštanty. Metóda `CreateAccessToken(User user)` vytvorí JWT podpísaný algoritmom HMAC SHA-256 s dobou platnosti 15 minút (`Jwt:AccessTokenMinutes`). Token nesie tri tvrdenia (claims): identifikátor používateľa (`sub`, GUID), e-mailovú adresu a rolu, spolu s vydavateľom a príjemcom z konfigurácie. Metóda `GenerateRefreshToken()` vytvára kryptograficky bezpečný 64-bajtový náhodný reťazec zakódovaný v Base64 pomocou `RandomNumberGenerator.GetBytes(64)`. Samotný obnovovací token sa však na server neukladá v otvorenej podobe: metóda `CreateRefreshToken(Guid userId)` vráti dvojicu (entita, surový token), pričom do databázy sa uloží iba jeho deterministický SHA-256 haš (statická metóda `HashRefreshToken`) a surovú hodnotu dostane výhradne klient. Rýchly haš SHA-256 je tu postačujúci, keďže token má vysokú entropiu (64 náhodných bajtov) a útok hrubou silou naň prakticky neprichádza do úvahy. Na rozdiel od hesiel nie je potrebný pomalý adaptívny haš ani soľ.

4.2.3 Registrácia a prihlásenie

Pri registrácii (`POST /api/auth/register`) server overí, že zadaný e-mail ešte nie je použitý (inak vráti `409 Conflict`), heslo zahašuje algoritmom BCrypt [9] a vytvorí nový účet. Následne sa používateľovi pokúsi odoslať uvítací e-mail, pričom prípadné zlyhanie odoslania sa iba zaznamená do logu a samotnú registráciu nezruší.

Pri prihlásení (`POST /api/auth/login`) server nájde používateľa podľa e-mailu a overí heslo funkciou `BCrypt.Verify`, pri nezhode vráti

401 Unauthorized bez toho, aby prezradil, či zlyhal e-mail alebo heslo. Po úspešnom overení vygeneruje prístupový token s platnosťou 15 minút a nový obnovovací token s platnosťou 7 dní, ktorého haš uloží do databázy, a oba tokeny vráti klientovi.

4.2.4 Obnovenie tokenov, rotácia a odhlásenie

Model RefreshToken uchováva haš tokenu (TokenHash), čas vytvorenia (CreatedAt), čas expirácie (ExpiresAt), čas odvolania (RevokedAt) a pri rotácii aj haš tokenu, ktorý ho nahradil (ReplacedByTokenHash). Vlastnosť IsActive obsahuje true iba vtedy, ak token nebol odvolaný a ešte nevypršal.

Po vypršaní prístupového tokenu klient zavolá POST /api/auth/refresh s obnovovacím tokenom. Server predložený token zahašuje a podľa hašu vyhľadá zodpovedajúci záznam v databáze. Ak taký neexistuje alebo už vypršal, vráti 401 Unauthorized. Ak je nájdený token platný, server vykoná rotáciu: pôvodný token zneplatní nastavením RevokedAt, vytvorí nový obnovovací token, prepojí ich cez ReplacedByTokenHash a klientovi vráti nový prístupový aj nový obnovovací token. Tento nový obnovovací token dostáva čerstvú sedemdňovú platnosť, čím dochádza k plynulému posúvaniu expirácie pri aktívnom používaní aplikácie.

Osobitne je ošetrené znovupoužitie tokenu. Ak klient predloží obnovovací token, ktorý už bol odvolaný, ide s vysokou pravdepodobnosťou o jeho zneužitie. Server v takom prípade zneplatní všetky aktívne obnovovacie tokeny daného používateľa, zaznamená upozornenie do logu a vráti 401 Unauthorized, čím ho odhlási zo všetkých zariadení. Súbežné prihlásenie používateľa na viacerých zariadeniach sa uskutoční bez problémov, keďže každá relácia má vlastný pár obnovovacích tokenov

Pri odhlásení (POST /api/auth/logout) server nájde príslušný obnovovací token (podľa jeho haša) a zneplatní ho nastavením RevokedAt na aktuálny čas (tzv. soft delete). Token ostáva v databáze, je však znefunkčnený. Ak sa token v databáze nenachádza, operácia sa napriek tomu považuje za úspešnú, čím je idempotentná.

Keďže pri rotácii pribúda s každou obnovou nový záznam, dlhodobo neaktívne tokeny odstraňuje na pozadí bežiaca služba TokenCleanup BackgroundService. Tá v šesťhodinových intervaloch zmaže z databázy tie obnovovacie tokeny, ktoré sú odvolané alebo expirované dlhšie, než určuje konfiguračná hodnota Jwt:RefreshTokenRetentionDays (predvolene 7 dní), čím zabraňuje neobmedzenému rastu tabuľky obnovovacích tokenov.

4.2.5 Obnovenie zabudnutého hesla

Obnovu zabudnutého hesla zabezpečujú dva verejné endpointy. Pri POST /api/auth/forgot-password v prípade, že účet s daným e-mailom existuje, server zmaže predchádzajúce nepoužité kódy, vygeneruje šesťciferný kód, uloží iba jeho BCrypt haš s platnosťou 15 minút a kód odošle e-mailom.

Odpoveď servera je vždy rovnaká bez ohľadu na to, či účet existuje, čím sa bráni zisťovaniu registrovaných e-mailov (tzv. *anti-enumeration*).

Pri `POST /api/auth/reset-password` server vyhľadá platný, nepoužitý a neexpirovaný kód a porovná ho s uloženým hašom. Po úspešnom overení používateľ nastaví nové heslo, kód server označí ako použitý a zároveň odvolá všetky aktívne obnovovacie tokeny používateľa, čím ho odhlási zo všetkých zariadení. Následne sa môže používateľ opätovne prihlásiť s novým heslom.

4.2.6 Ochrana endpointov

Middleware `UseAuthentication()` a `UseAuthorization()` sú v `Program.cs` zaregistrované v správnom poradí. Kontroléry pracujúce s používateľskými dátami (`UsersController`, `LocationController`, `FavoritesController` aj `ProvidersController`) sú chránené atribútom `[Authorize]` na úrovni celej triedy. Verejné operácie, ako prihlásenie, registrácia či obnovenie tokenu, autorizáciu nevyžadujú a sú v prípade potreby označené `[AllowAnonymous]`. Importný endpoint je namiesto JWT chránený vlastným atribútom `[ImportApiKey]`. Identita prihláseného používateľa je v kontroléroch dostupná cez vlastnosť `sub`, napríklad volaním `User.FindFirst(ClaimTypes.NameIdentifier)`.

4.3 Implementácia mobilnej aplikácie

Mobilná aplikácia PerkAround je vyvinutá pomocou frameworku .NET MAUI s cieľovou platformou Android. Skladá sa z niekoľkých obrazoviek prepínaných cez vlastnú navigačnú lištu, pričom komunikácia so serverom prebieha prostredníctvom HTTP klienta s automatickou správou JWT tokenov.

4.3.1 .NET MAUI

Aplikácia je postavená na platforme .NET MAUI [12] (Multi-platform App UI) vo verzii .NET 8. Projekt cieli výhradne na platformu Android (`net8.0-android`) s minimálnou podporovanou úrovňou API 21, čo zodpovedá systému Android 5.0 (*Lollipop*). Architektúra single-project umožňuje udržiavať zdrojový kód, zdroje a konfiguráciu v jednom projekte. Konfigurácia závislostí a registrácia služieb prebieha v `MauiProgram.cs` pomocou štandardného DI kontajnera `Microsoft.Extensions.DependencyInjection`.

4.3.2 Organizácia klienta

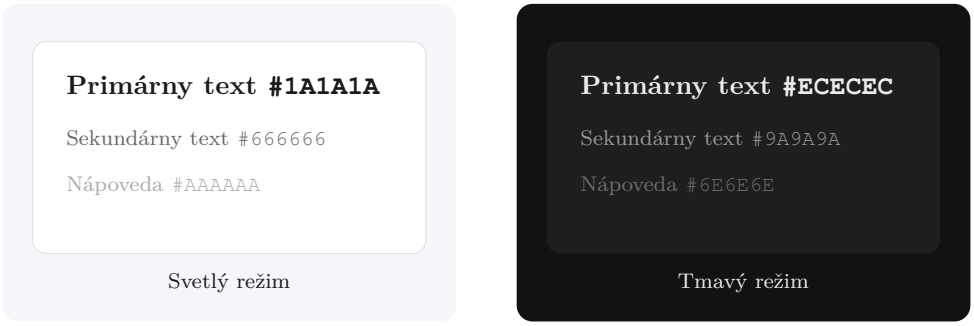
Mobilný klient udržiava logiku jednotlivých obrazoviek v ich code-behind a aplikačnú logiku centralizuje do služieb v zložke `Services/`: `AuthService` spravuje prihlásenie a reláciu, `TokenService` ukladá a načítava tokeny a `LocationTrackingService` riadi sledovanie polohy. Jednotlivé stránky si tieto služby vyžiadajú cez vkladanie závislostí (DI) a reagujú na ich udalosti (napríklad `LocationUpdated`). Pri zoznamoch ponúk sa uplatňuje dátové viazanie: položky obaľujú menšie modely implementujúce `INotifyPropertyChanged` (napríklad `NearbyBranchViewModel`), ktoré danej obrazovke sprístupňujú hodnoty a priebežne ho aktualizujú.

4.3.3 Štýly a témy

Vizuálna identita navrhnutá v časti 3.7 je v aplikácii realizovaná centralizovane pomocou slovníkov zdrojov (tzv. *resource dictionaries*). Všetky farby sú definované v súbore `Colors.xaml` a spoločné štýly ovládacích prvkov v `Styles.xaml`. Jednotlivé obrazovky sa na ne odkazujú cez `StaticResource`, čím je zaručený jednotný vzhľad a jednoduchá úprava na jednom mieste. Bezpätkové písmo Open Sans (rezy Regular a Semibold) je zaregistrované v `MauiProgram.cs` pomocou `ConfigureFonts` a nastavené ako predvolené naprieč celou aplikáciou.

Aplikácia podporuje svetlý aj tmavý režim vzhľadu. Farby plôch, textu a oddeľovačov majú v slovníku definované svetlé aj tmavé varianty a v rozhraní sa priraďujú cez `AppThemeBinding`, ktoré automaticky vyberie správnu hodnotu podľa aktuálnej témy. Vo svetlom režime tvorí pozadie svetlosivá (`#F5F6FA`) s bielymi kartami (`#FFFFFF`) a tmavým textom, v tmavom režime takmer čierne pozadie (`#121212`) s tmavosivými kartami (`#1E1E1E`) a svetlým textom. Text

je v oboch režimoch odstupňovaný do troch úrovní (primárny, sekundárny a nápoveda). Používateľ si môže tému zvoliť v nastaveniach aplikácie, pričom voľba „System“ preberá tému OS. Obidva režimy sú znázornené na obrázku 9. Okrem všeobecnej témy aplikácie si používateľ môže zvoliť tému zobrazovanej mapy zvlášť. Toto implementačné rozhodnutie bolo do aplikácie pridané po požiadavke skúšobných používateľov, ktorí aplikáciu testovali – chceli mať možnosť „Dark“ témy aplikácie spolu s „Light“ témou mapy.



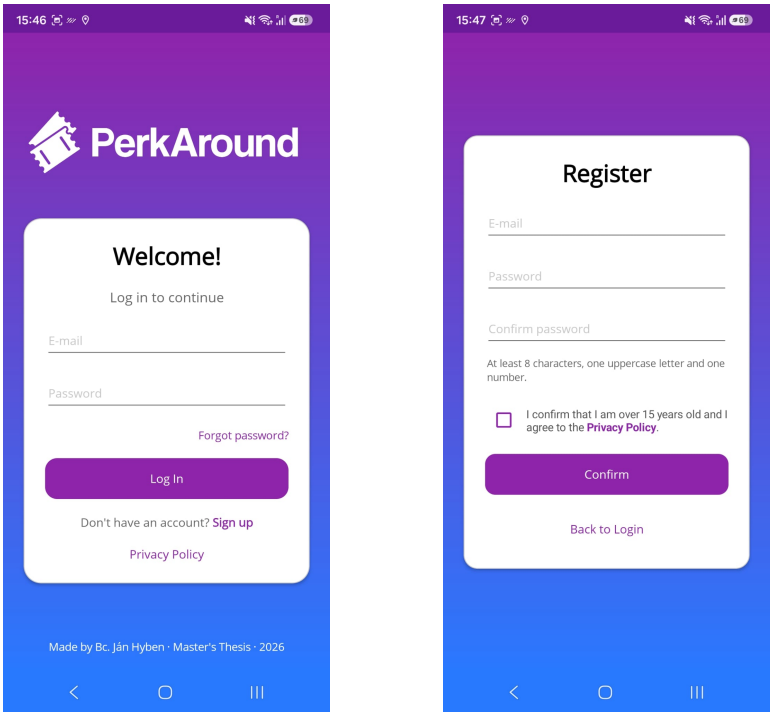
Obr. 9: Farby pozadia a textu vo svetlom a tmavom režime.

4.3.4 Prihlásenie a registrácia

Kým používateľ nie je prihlásený, aplikácia zobrazuje autentifikačné obrazovky v samostatnom navigačnom kontexte (`AuthShell`). Vstupnou obrazovkou je prihlásenie (`LoginPage`), z ktorej sa dá prejsť na registráciu alebo na obnovu zabudnutého hesla.

Na prihlasovacej obrazovke používateľ zadá e-mail a heslo a potvrdí tlačidlom „Log In“. Po základnej kontrole vyplnenia polí sa volá metóda `LoginAsync` služby `AuthService`. Pri úspechu sa uložia tokeny aj e-mail používateľa, aplikácia sa prepne do prihláseného kontextu (`MainShell`) a spustí sa poprihlasovacia inicializácia (vyžiadanie oprávnení, spustenie sledovania polohy, synchronizácia obľúbených a kontrola aktualizácií). Pri neúspechu sa zobrazí chybová hláška. Obrazovka ďalej obsahuje odkaz „Sign up“ na registráciu, odkaz „Forgot password?“ na obnovu hesla (s predvyplnením už zadaného e-mailu) a odkaz na zásady ochrany osobných údajov „Privacy Policy“.

Registračný formulár (`RegisterPage`) obsahuje polia e-mail, heslo a potvrdenie hesla. Pred odoslaním prebieha validácia na strane klienta: kontroluje sa platný tvar e-mailu, sila hesla podľa pravidiel (trieda `PasswordRules`) a zhoda oboch hesiel. Registráciu navyše podmieňuje povinné zaškrtnutie súhlasu, v ktorom používateľ potvrdzuje, že má viac ako 15 rokov a súhlasí so zásadami ochrany osobných údajov (v súlade s GDPR). Po úspešnej registrácii, ktorú spracuje metóda `RegisterAsync` (server uloží heslo len ako BCrypt haš), je používateľ informovaný a presmerovaný späť na prihlásenie.



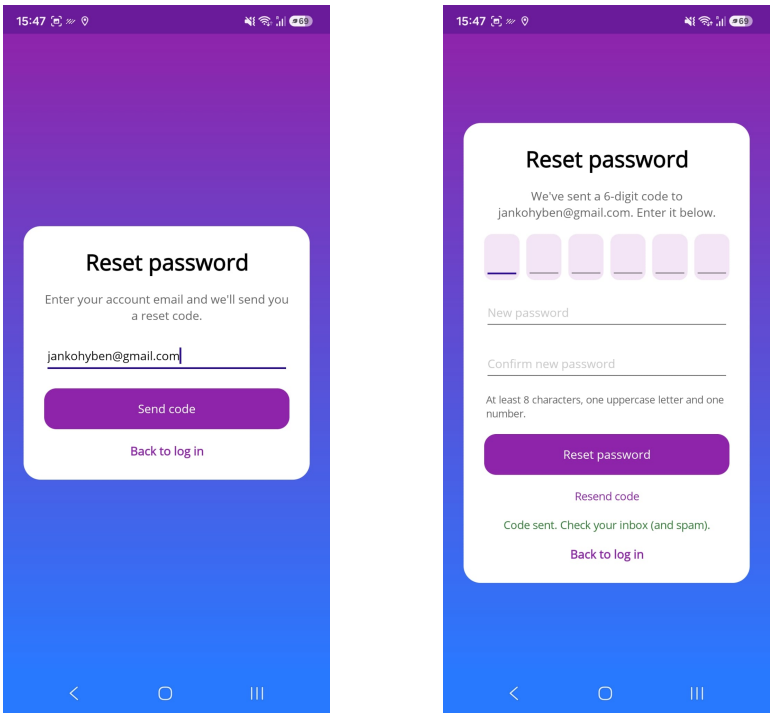
Obr. 10: Prihlasovacia obrazovka (vľavo) a registračný formulár s pravidlami hesla a povinným súhlasom so spracúvaním osobných údajov (vpravo).

4.3.5 Obnovenie zabudnutého hesla

Obnovu zabudnutého hesla zabezpečuje jediná obrazovka `ForgotPassword Page`, ktorá prechádza dvomi krokmi.

V prvom kroku používateľ zadá e-mailovú adresu svojho účtu a potvrdí tlačidlom „Send code“. Volanie `ForgotPasswordAsync` požiada server o odoslanie šesťciferného kódu na daný e-mail (na serveri sa uchováva len jeho haš s platnosťou 15 minút). Server odpovedá rovnako bez ohľadu na to, či účet s daným e-mailom existuje (ochrana proti zisťovaniu registrovaných e-mailov), preto aplikácia vždy prejde k druhému kroku.

V druhom kroku používateľ zadá prijatý kód do šiestich samostatných políčok a zvolí nové heslo spolu s jeho potvrdením, pričom nové heslo musí spĺňať rovnaké pravidlá, ako pri registrácii. Po potvrdení tlačidlom „Reset password“ sa volá `ResetPasswordAsync`, server overí kód voči uloženému hašu, nastaví nové heslo a zneplatní všetky aktívne obnovovacie tokeny, čím používateľa odhlási zo všetkých zariadení. Obrazovka umožňuje aj opätovné odoslanie kódu odkazom „Resend code“. Po úspešnej zmene je používateľ presmerovaný na prihlásenie.



Obr. 11: Obnova hesla: zadanie e-mailu na odoslanie kódu (vľavo) a zadanie šesťciferného kódu spolu s novým heslom (vpravo).

4.3.6 Navigácia

Aplikácia rozlišuje dva navigačné kontexty. Neprihlásený používateľ vidí `AuthShell` s prihlasovacou obrazovkou. Po prihlásení sa načíta `MainShell` s hlavnou obrazovkou `MainPage`, ktorá obsahuje vlastnú navigačnú komponentu `AppLayout`. Hlavná navigácia je realizovaná navigačnou lištou umiestnenou v dolnej časti obrazovky (`AppLayout.xaml`), ktorá obsahuje päť ikon prepínajúcich karty:

- Map – mapové zobrazenie prevádzok v okolí,
- Favorites – obľúbených poskytovateľov zliav používateľa,
- Offers – zoznam aktuálnych zľavových ponúk v okolí používateľa,
- Settings – nastavenia aplikácie,
- Profile – profil prihláseného používateľa.

Aktívna karta je zvýraznená tmavšou farbou (`#6A1B9A`), neaktívne karty sú zobrazené vo svetlejšej fialovej (`#8E24AA`), teda v rovnakej farbe, ako zvyšok navigačnej lišty. Prepínanie kariet obsluhuje `NavigationTab` dátového typu enum a príslušná logika v `MainPage.xaml.cs`.

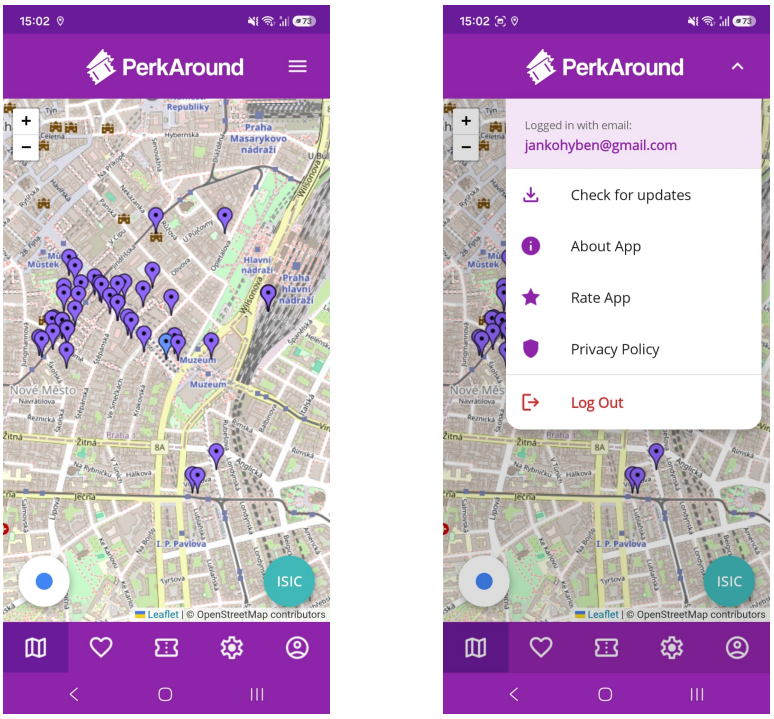
Popri spodnej navigačnej lište obsahuje `AppLayout` aj hornú lištu (tzv. *header*) s logom aplikácie a tlačidlom rozbaľovacej ponuky (tzv. *burger* menu) v pravom hornom rohu. Zatiaľ čo spodná lišta slúži na prepínanie medzi hlavnými obsahovými sekciami, burger menu združuje sekundárne, účtové a aplikačné akcie, ktoré nie sú súčasťou bežného prehliadania ponúk. Toto rozdelenie udržuje spodnú navigáciu prehľadnú a vyhradenú výlučne pre obsah, pričom menej frekventované funkcie zostávajú dostupné po jednom klepnutí bez toho, aby zaberali miesto v hlavnej lište.

Po klepnutí na tlačidlo sa jeho ikona zmení na šípku nahor a zhora sa vysunie karta (`MenuOverlay`) ukotvená pod hlavičkou pri pravom okraji obrazovky spolu s polopriehľadným stmavením zvyšku obsahu. Opätovné klepnutie na tlačidlo alebo klepnutie mimo karty (na stmavené pozadie) ponuku zatvorí. Otvorenie aj zatvorenie sprevádza krátka animácia priehľadnosti. V hlavičke karty je zobrazená e-mailová adresa prihláseného používateľa, ktorá sa načíta okamžite z lokálnych preferencií a následne sa overí zo servera. Samotná ponuka obsahuje tieto položky:

- *Check for updates* – manuálne overenie dostupnosti novšej verzie aplikácie prostredníctvom služby `UpdateService` (keďže aplikácia je distribuovaná mimo obchodu Google Play, implementuje vlastný mechanizmus aktualizácií, podrobne v časti 4.10),
- *About App* – informačné okno s verziou aplikácie a kontaktnými údajmi autora,

- *Rate App* – položka na hodnotenie aplikácie (vzhľadom na distribúciu mimo obchodu Google Play je funkcia len informatívna, umožňuje plynulé rozšírenie tejto funkcionality po potenciálnom budúcom publikovaní aplikácie v obchode Google Play),
- *Privacy Policy* – zobrazenie zásad ochrany osobných údajov (`PrivacyPolicyPage`),
- *Log Out* – odhlásenie používateľa a vrátenie aplikácie do neprihláseného kontextu (`AuthShell`).

Nezávisle od rozbaľovacej ponuky je naprieč hlavnými obrazovkami v pravom dolnom rohu umiestnené plávajúce okrúhle tlačidlo s nápisom „ISIC“ (súčasť spoločného rozloženia `AppLayout`), ktoré slúži ako rýchla skratka k digitálnemu preukazu ISIC. Tlačidlo po klepnutí otvorí oficiálnu aplikáciu Alive App: prednostne cez hĺbkový odkaz so schémou `alive-app://` smerujúci priamo na zobrazenie preukazu na celú obrazovku. Ak sa tento odkaz na zariadení nepodarí spracovať, tlačidlo otvorí Alive App bežným spôsobom, a ak aplikácia nie je nainštalovaná, presmeruje používateľa na jej stránku v obchode Google Play. Používateľ tak môže preukázať nárok na zľavu priamo z aplikácie bez zdĺhavého hľadania preukazu inde a nemusí tak opúšťať prostredie PerkAround.



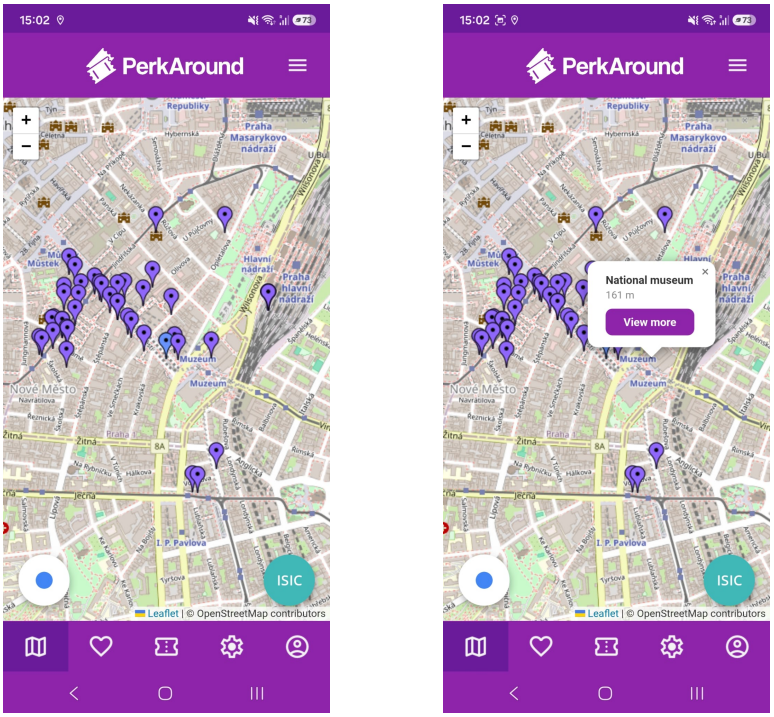
Obr. 12: Úvodná obrazovka s mapou (*MapView*) po spustení aplikácie (vľavo) a rozbalené *burger* menu s e-mailom prihláseného používateľa a ďalšími sekundárnymi akciami (vpravo).

4.3.7 MapView

Mapové zobrazenie je implementované ako `WebView` načítavajúci lokálny HTML súbor `map.html` zo zdrojov aplikácie (`android_asset/map.html`). Mapa je renderovaná pomocou JavaScriptovej knižnice Leaflet.js [17] s mapovými dlaždicami od OpenStreetMap [18]. Tento prístup nevyžaduje žiadny API kľúč a používanie máp nie je spoplatnené.

Komunikácia medzi C# kódom a JavaScriptom prebieha cez `EvaluateJavaScriptAsync()`. Aplikácia volá dve JavaScriptové funkcie: `setUserLocation(lat, lng)` nastaví polohu používateľa (modrý marker) a `addBranches(branches)` zobrazí na mape markery blízkych prevádzok (fialové markery). Počiatočná poloha mapy je nastavená na Olomouc, Česká republika, pričom táto poloha sa okamžitým načítaním mapy prepne na aktuálnu polohu používateľa. Pre ručné vycentrovanie aktuálnej polohy používateľa je možné použiť tlačidlo umiestnené v ľavom dolnom rohu mapy, ktoré implementuje práve túto funkcionalitu.

Okrem odovzdávania dát zo C# kódu do JavaScriptu je funkčný aj opačný smer toku dát, teda z mapy späť do aplikácie. Každá prevádzka je na mape vykreslená ako fialový marker, po klepnutí naň sa otvorí malé vyskakovacie okno (Leaflet popup) s názvom prevádzky, vzdialenosťou a tlačidlom „View more“. Keďže `WebView` neumožňuje priame volanie C# kódu z JavaScriptu, tlačidlo využíva most postavený na vlastnej URL schéme: jeho stlačenie presmeruje `WebView` na adresu tvaru `appcmd://viewmore/{id}`. Toto zdanlivé navigovanie zachytí obslužná metóda udalosti `Navigating` v `MapView.xaml.cs`, ktorá skutočné načítanie zruší, z adresy získa identifikátor pobočky, dohľadá príslušný záznam a otvorí detail ponuky (`ProviderDetailPopup`).



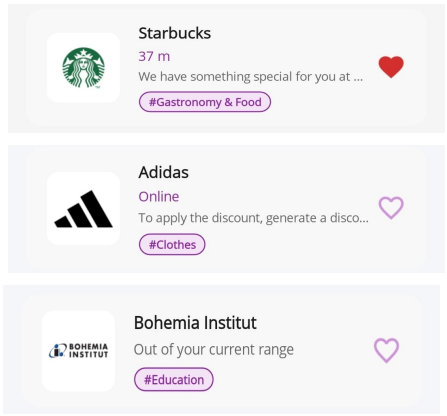
Obr. 13: Mapové zobrazenie prevádzok v okolí (vľavo) a vyskakovacie okno s názvom prevádzky, vzdialenosťou a tlačidlom „View more“ po klepnutí na fialový marker (vpravo).

4.3.8 FavView a OffersView

Obrazovky `OffersView` (zoznam ponúk v okolí) a `FavView` (obľúbení poskytovatelia) predstavujú dve z piatich hlavných kariet aplikácie. Obe sú implementované ako `ContentView`, ktoré sa vkladajú do tela komponentu `AppLayout`, a zdieľajú vizuálnu podobu kariet aj ich dátový model. Líšia sa zdrojom dát: `OffersView` zobrazuje prevádzky v aktuálnom okolí používateľa, zatiaľ čo `FavView` zobrazuje poskytovateľov, ktorých si používateľ označil ako obľúbených.

Karta prevádzky a jej ViewModel

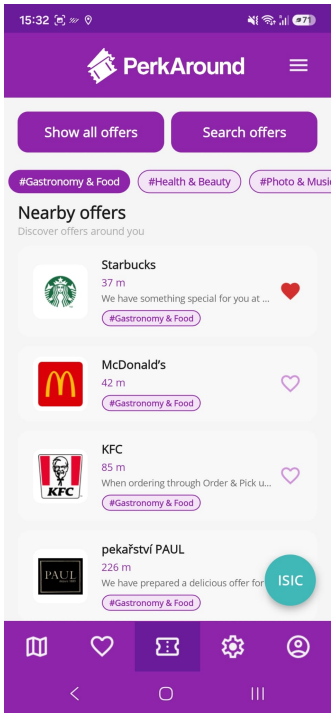
Jednotlivé položky v oboch zoznamoch vychádzajú zo spoločného modelu `NearbyBranchViewModel`, ktorý obaľuje dátový objekt pobočky (`NearbyBranch`) a poskytuje z neho hodnoty odvodené pre používateľské rozhranie. Karta prevádzky v zozname (`CollectionView`) obsahuje logo, názov prevádzky, vzdialenosť k najbližšej pobočke, krátky náhľad ponúk, štítok kategórie vo forme hashtagu (napríklad `#Sports`) a ikonu srdca vyjadrujúcu stav obľúbenosti prevádzkovateľa tejto zľavy. ViewModel formátuje vzdialenosť podľa zvolených jednotiek (metre alebo stopy) a podľa nej prispôsobuje text aj farbu, napríklad „Online“ pre ponuky bez fyzickej pobočky alebo „Out of your current range“ pre prevádzky mimo nastaveného okruhu.



Obr. 14: Karta prevádzky v troch stavoch: štandardná ponuka v okolí s uvedenou vzdialenosťou (hore), online poskytovateľ bez fyzickej pobočky (v strede) a prevádzka mimo nastaveného okruhu vyhľadávania (dole).

OffersView

Obrazovka `OffersView` získava dáta zo služby `LocationTrackingService` – pri vytvorení použije poslednú známu množinu pobočiek (`LastBranches`) a následne sa prihlási na odber udalosti `LocationUpdated`, takže sa zoznam automaticky aktualizuje pri každej zmene polohy. Kým nedorazia prvé dáta, prekrýva zoznam indikátor načítavania. Odber udalostí je naviazaný na životný cyklus komponentu (metóda `OnParentSet`) – pri opustení karty sa odber zruší a zároveň sa vynuluje filter kategórií.



Obr. 15: Obrazovka `OffersView` s tlačidlami „Show all offers“ a „Search offers“, pruhom filtrov podľa kategórií (aktívny je „Gastronomy & Food“) a zoznamom ponúk v okolí zoradeným podľa vzdialenosti.

Ponuky sú zoradené tak, že prevádzky patriace do používateľom preferovaných kategórií (nastavených v sekcii nastavení) sa zobrazujú ako prvé a v rámci každej skupiny sú zoradené od najbližšej po najvzdialenejšiu. Aby pri častých aktualizáciách polohy nedochádzalo k preblikávaniu zoznamu, obrazovky sa uchovávajú vo vyrovnávacej pamäti podľa identifikátora pobočky a pri novom dátovom cykle sa ich hodnoty len obnovia namiesto opätovného vytvorenia inštancie. To isté zabezpečuje aj rozširujúca metóda `SyncTo`, ktorá zoznam zosúladí „na mieste“ – ponechá nezmenené položky a pridá či odoberie len tie, ktoré pribudli alebo zmizli.

Filtrovanie podľa kategórií

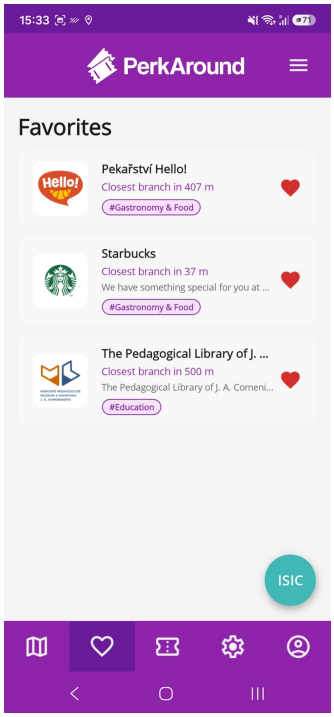
Nad zoznamom ponúk sa nachádza vodorovne rolovateľný pruh filtrovacích štítkov (tzv. *pills*) generovaný z kategórií, ktoré sa reálne vyskytujú medzi aktuálne zobrazenými ponukami. Klepnutím na štítok sa daná kategória pridá alebo odoberie z výberu (možný je aj výber viacerých kategórií naraz) a zoznam sa okamžite prefiltruje. Kategórie sú reprezentované číselnými identifikátormi zdieľanými so serverom aj so službou na zber dát, pričom ich názvy poskytuje pomocná trieda `Categories`.

Zobrazenie všetkých ponúk a vyhľadávanie

Okrem ponúk v okolí umožňuje `OffersView` pristúpiť aj k úplnému katalógu prostredníctvom dvoch tlačidiel v hlavičke – „Show all offers“ a „Search offers“. Obe otvárajú modálnu stránku `AllOffersPage`, ktorá načíta všetkých poskytovateľov (nielen tých v okolí) a pracuje v dvoch režimoch. V režime *Show all* sú prevádzky zoskupené – ak má používateľ nastavené preferované kategórie, zobrazí sa navrchu táto skupina a za ňou nasledujú abecedne zoradené ostatné podľa ich začiatočného písmena. V režime *Search* sa zobrazí vyhľadávacie pole, ktoré zoznam priebežne filtruje podľa názvu prevádzky bez ohľadu na veľkosť písmen. V oboch režimoch je súčasne k dispozícii aj filter podľa kategórií.

FavView

Obrazovka `FavView` zobrazuje zoznam obľúbených prevádzkovateľov, ktorý získava zo služby `FavoritesService`. Karty majú rovnaké rozloženie ako v zozname ponúk, ikona srdca je však vyplnená a slúži na odobranie z obľúbených – po klepnutí sa položka najprv okamžite odstráni zo zoznamu a až následne sa zmena potvrdí na serveri. Obľúbené sú riešené ako offline-first: identifikátory obľúbených aj neodoslané zmeny sa uchovávajú v lokálnych preferenciách zariadenia a samotné karty sa ukladajú do súborovej vyrovnávacej pamäte, takže zoznam obľúbených prevádzkovateľov je dostupný aj bez pripojenia.



Obr. 16: Obrazovka `FavView` (Favorites) so zoznamom obľúbených poskytovateľov, pri každej je uvedená vzdialenosť k najbližšej pobočke.

Spoločné správanie

Obe obrazovky reagujú na zmenu obľúbených prostredníctvom udalosti `FavoritesChanged` – po prepnutí obľúbenosti sa zosúladí stav srdc všetkých zobrazených kariet vrátane ostatných pobočiek toho istého poskytovateľa. Klepnutie na kartu otvorí vyskakovacie okno s detailom prevádzky (`ProviderDetailPopup`), ktoré zobrazí jej ponuky a ďalšie informácie.

4.3.9 Detail ponuky

Detail ponuky je implementovaný ako komponent `ProviderDetailPopup` – vyskakovacie okno ukotvené pri spodnom okraji obrazovky, ktoré sa vysúva zdola nahor a prekrýva zvyšok obsahu stmaveným pozadím. Zatvára sa klepnutím na tlačidlo zatvorenia, klepnutím mimo karty alebo potiahnutím. Ide o centrálny prvok, ktorý je zdieľaný naprieč celou aplikáciou.
Detail ponuky je dostupný zo všetkých miest, kde sa zobrazuje ponuka, teda:

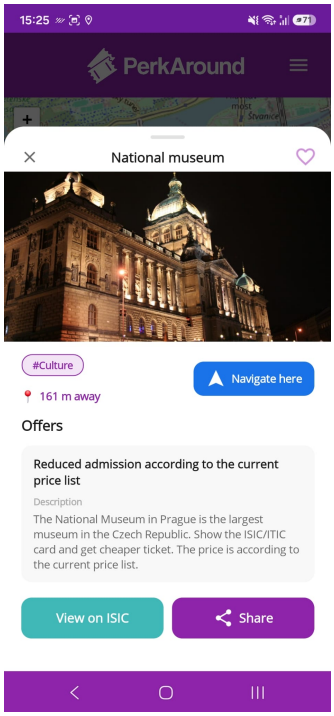
- z mapy (`MapView`) po klepnutí na marker a tlačidlo „View more“,
- zo zoznamu ponúk v okolí (`OffersView`) po klepnutí na kartu,
- zo zoznamu obľúbených (`FavView`),
- z úplného katalógu a vyhľadávania (`AllOffersPage`).

Vo všetkých prípadoch sa otvára tá istá inštancia komponentu, čím je zabezpečená konzistentná podoba detailu bez ohľadu na miesto vyvolania.

Po otvorení sa najprv zobrazia údaje, ktoré sú okamžite k dispozícii z ViewModelu karty: názov prevádzky, ilustračný obrázok, štítok kategórie a vzdialenosť (prípadne označenie „Online provider“ pri prevádzkach bez fyzickej pobočky). Súčasne sa z API dotiahnu podrobnejšie údaje (metóda `GetProviderDetailAsync`), počas čoho sa zobrazuje indikátor načítavania. Po ich načítaní detail obsahuje:

- zoznam konkrétnych zľavových ponúk poskytovateľa (názov a popis ponuky),
- ikonu srdca na pridanie alebo odobranie prevádzky z obľúbených,
- tlačidlo „Navigate here“ (len pre fyzickú pobočku v dosahu), ktoré otvorí navigáciu k pobočke v aplikácii Google Maps s pripravenou pešou trasou,
- tlačidlo „View on ISIC“, ktoré otvorí pôvodnú stránku ponuky na portáli ISIC vo webovom prehliadači,
- tlačidlo „Share“ na zdieľanie ponuky prostredníctvom systémového dialógu (odkaz na ponuku spolu s krátkym propagačným textom).

Tlačidlom „Share“ sa dá daná ponuka zdieľať prostredníctvom rôznych sociálnych sietí a chatovacích aplikácií. Text je tiež možné skopírovať a naložiť s ním iným spôsobom. Samotný text obsahuje krátku správu spolu s hypertextovým odkazom na detail ponuky na portáli ISIC. Tlačidlá „View on ISIC“ a „Share“ sa zobrazia len vtedy, ak má prevádzka vyplnenú zdrojovú adresu ponuky.



Obr. 17: Detail ponuky (`ProviderDetailPopup`) otvorený z mapy.

4.3.10 SettingsView

Obrazovka `SettingsView` sústreďuje všetky nastavenia aplikácie. Je rozdelená do dvoch logických skupín: *používateľské nastavenia* („User Settings“), ktoré platia naprieč zariadeniami, a *lokálne nastavenia* daného zariadenia („App Settings“). Ich technické uloženie a synchronizáciu opisuje časť 4.8. Používateľské nastavenia („User Settings“) tvorí sekcia notifikácií:

- „Push notifications“ – zapnutie alebo vypnutie notifikácií (pri zapnutí si aplikácia vyžiada systémové povolenie),
- „Notify me about“ – rozsah notifikácií (napríklad všetky ponuky v okolí, obľúbené, výber kategórií alebo ich kombinácia), ktorý sa volí na samostatnej stránke,
- „Quiet hours“ – časový interval, počas ktorého sa notifikácie nezobrazujú,
- „Newsletter (email)“ – frekvencia e-mailového newslettera (vypnuté, denne, týždenne alebo mesačne),

- „Birthday greeting“ – narodeninový pozdrav e-mailom, riadok sa zobrazí len vtedy, ak má používateľ vyplnený dátum narodenia.

Lokálne nastavenia aplikácie na lokálnom zariadení („App Settings“) sú členené do sekcií:

- „General“ – okruh vyhľadávania ponúk (diskrétny posuvník potvrdzovaný tlačidlom „Save range“), preferované kategórie (vyberané na samostatnej stránke) a jednotky vzdialenosti (metrické alebo imperiálne),
- „Appearance“ – téma aplikácie (System, Light, Dark), samostatná téma mapy (Light, Dark) a jazyk rozhrania (momentálne iba anglický jazyk s možnosťou rozšírenia pri potenciálnom budúcom vývoji),
- „Location & privacy“ – zapnutie sledovania polohy (po vypnutí sa zobrazí oranžová výstražná lišta), režim šetrenia batérie (predĺži interval aktualizácie polohy), vymazanie lokálnej vyrovnávacej pamäte so zobrazením jej veľkosti a odkaz na zásady ochrany osobných údajov.

Tlačidlom „Restore default“ možno obnoviť predvolené hodnoty lokálnych nastavení. Tlačidlo je aktívne len vtedy, keď sa aktuálne nastavenia od predvolených líšia. Poslednou skupinou je sekcia „About“ s podobným obsahom ako burger menu prístupné z pravej hornej časti obrazovky aplikácie, teda s informáciami o aplikácii – položkou „About PerkAround“, hodnotením aplikácie, odoslaním spätnej väzby e-mailom a číslom aktuálnej verzie aplikácie. Zmena okruhu vyhľadávania a preferovaných kategórií priamo ovplyvňuje zobrazovanie ponúk, preto si aplikácia po uložení nového okruhu okamžite znovu načíta polohu a ponuky v okolí.

4.3.11 ProfileView

Obrazovka `ProfileView` slúži na správu používateľského účtu. Pri otvorení načíta údaje prihláseného používateľa zo servera (metóda `GetCurrentUserAsync`) a zobrazí jeho profilovú fotografiu (alebo zástupný obrázok, ak nie je nastavená), zobrazované meno (zložené z mena a priezviska, prípadne odvodené z e-mailovej adresy) a samotnú e-mailovú adresu.

V hornej časti sa nachádza tlačidlo na úpravu profilu, ktoré podľa úplnosti vyplnených údajov nesie text „Edit profile“, alebo výzvu „Complete your profile“. Otvára samostatnú stránku (`EditProfilePage`), na ktorej môže používateľ upraviť svoje meno, priezvisko, dátum narodenia a profilovú fotografiu. Nahrávanie profilových fotografií je obmedzené veľkosťou snímky do 1MB z dôvodu úspory úložného priestoru.

Ďalej obrazovka ponúka nasledujúce operácie s účtom, z ktorých každá otvára samostatnú stránku alebo dialóg:

- „Change email“ – zmena e-mailovej adresy, ktorá si vyžaduje potvrdenie aktuálnym heslom,
- „Change password“ – zmena hesla,
- „Log out“ – odhlásenie, ktoré zastaví sledovanie polohy a vráti aplikáciu do prihlasovacieho kontextu,
- „Delete account“ – trvalé zmazanie účtu a všetkých súvisiacich údajov, operácia vyžaduje potvrdenie a zadanie hesla a napĺňa právo na výmaz podľa nariadenia GDPR (článok 17),
- odkaz na zásady ochrany osobných údajov (`PrivacyPolicyPage`).

4.3.12 GDPR

Aplikácia PerkAround spracúva osobné údaje v súlade s nariadením GDPR [19]. Jej súčasťou sú úplné zásady ochrany osobných údajov (*Privacy Policy*) dostupné priamo v aplikácii na obrazovke `PrivacyPolicyPage`, na ktorú sa používateľ dostane z nastavení, z profilu, z ponuky burger menu aj z prihlasovacej a registračnej obrazovky. Dokument uvádza prevádzkovateľa systému (autor aplikácie ako študent Univerzity Palackého v Olomouci vrátane kontaktu), spracúvané kategórie údajov, účely a právne základy spracúvania, dobu uchovávania, príjemcov údajov aj práva používateľa.

Aplikácia pracuje s nasledujúcimi kategóriami osobných údajov:

- e-mailová adresa – slúži ako prihlasovací identifikátor, uložená v databáze na strane servera,
- heslo – uložené výhradne vo forme BCrypt hašu [9] a pôvodné heslo nie je nikde uchované,
- GPS poloha – najcitlivejší údaj a zároveň jadro funkcionality aplikácie, spracúva sa iba počas behu aplikácie na účel zobrazenia blízkych ponúk a odoslania notifikácií. Na serveri sa neuchováva história pohybu, ale len posledná známa poloha (tabuľka `UserLocationCache`), pričom sa odosiela až pri pohybe väčšom ako 50 metrov,
- voliteľné profilové údaje – meno, priezvisko, dátum narodenia (použitý len na narodeninový pozdrav) a profilová fotografia,
- obľúbené prevádzky a preferencie notifikácií a newslettera,
- technické údaje – pri načítaní mapových dlaždíc, obrázkov alebo otvorení externých služieb sa týmto tretím stranám nevyhnutne odovzdáva IP adresa zariadenia.

Právne základy spracúvania

Podstatným rozlíšením oproti zjednodušenému chápaniu je, že spracúvanie nie je založené výlučne na súhlase. Vytvorenie a správa účtu, autentifikácia a samotné vyhľadávanie ponúk podľa polohy sa opierajú o plnenie služby požadovanej používateľom podľa článku 6 ods. 1 písm. b) nariadenia – keďže účelom aplikácie je zobrazovať zľavy v okolí, bez polohy nemôže plniť svoju funkciu. Súhlas podľa článku 6 ods. 1 písm. a) sa vzťahuje výhradne na marketingovú komunikáciu, teda na e-mailový newsletter a narodeninový pozdrav, tie sú predvolene vypnuté, používateľ ich aktivuje sám a kedykoľvek sa z nich môže odhlásiť. Zabezpečenie služby a transakčné e-maily (uvítací, obnova hesla, zmazanie účtu) sú podložené oprávneným záujmom podľa písm. f), prípadne plnením služby podľa písm. b).

Príjemcovia údajov a uchovávanie

Osobné údaje nie sú na predaj tretím stranám. Na poskytovanie svojich funkcií sa aplikácia napája na tretie strany, ktoré môžu spracúvať technické údaje: OpenStreetMap (mapové dlaždice), Google Maps (navigácia po klepnutí na „Navigate here“), Microsoft Office 365 (odosielanie e-mailov), Amazon S3 (obrázky ponúk) a aplikácia Alive App či Google Play (po klepnutí na tlačidlo ISIC). Backend a databáza sú prevádzkované na infraštruktúre Univerzity Palackého v Olomouci. Údaje účtu sa uchovávajú, kým účet existuje a po jeho zmazaní v aplikácii sú všetky súvisiace záznamy (profil, obľúbené, poloha, notifikácie, fotografia) natrvalo odstránené. Kódy na obnovu hesla sa mažú po použití alebo po 15 minútach a keďže ide o časovo ohraničený akademický projekt, celá služba a uložené dáta môžu byť po odovzdaní a obhajobe práce vyradené z prevádzky.

Práva používateľa

Používateľ má podľa GDPR právo na prístup k údajom, ich opravu (meno a e-mail možno upraviť priamo v aplikácii), výmaz (zmazanie účtu v aplikácii, tzv. právo byť zabudnutý podľa článku 17), obmedzenie spracúvania, namietanie, prenosnosť údajov a odvolanie súhlasu s marketingom. Automatizované rozhodovanie s právnym alebo obdobne významným účinkom sa nevykonáva – notifikácie sa opierajú výlučne o vzdialenosť medzi polohou používateľa a prevádzkami. Aplikácia nie je určená deťom mladším ako 15 rokov (vek digitálneho súhlasu v Českej republike). Prípadnú sťažnosť môže používateľ podať dozornému úradu, ktorým je v Českej republike Úřad pro ochranu osobních údajů.

Z hľadiska zabezpečenia sú heslá uchovávané výhradne ako haš, prístup k API je chránený podpísanými tokenmi JWT a celá komunikácia medzi aplikáciou a serverom prebieha cez protokol HTTPS, autentifikačné tokeny sú na zariadení chránené úložiskom Android Keystore [8].

4.4 Komunikácia klient-server

Mobilná aplikácia komunikuje so serverom výhradne cez HTTP rozhranie. Všetky požiadavky prechádzajú centrálne nakonfigurovaným klientom s automatickou správou autorizačných hlavičiek.

4.4.1 HttpClient

HTTP komunikácia je realizovaná cez pomenovaný HttpClient registrovaný v `MauiProgram.cs` pomocou `IHttpClientFactory`. Klient má nastavenú bázovú adresu API a predvolenú hlavičku `Accept: application/json`. Bázová adresa sa líši medzi produkčným nasadením, kde smeruje na verejné doménové meno servera, a vývojovým prostredím, kde smeruje na lokálne bežiaci backend. Do pipeline je zaradený `AuthHeaderHandler` – vlastný `DelegatingHandler`, ktorý pred každou odchádzajúcou požiadavkou automaticky doplní hlavičku `Authorization: Bearer {token}` s aktuálnym access tokenom načítaným z `TokenService`.

Ak server vráti odpoveď `401 Unauthorized`, handler sa pokúsi obnoviť reláciu zavolaním `TryRestoreSessionAsync()` – t.j. odošle obnovovací token a získa nový pár tokenov (prístupový aj obnovovací), ktoré si uloží do bezpečného úložiska. Následne naklonuje pôvodnú požiadavku vrátane tela a hlavičiek, doplní nový token a požiadavku zopakuje. Klonovanie je nevyhnutné, pretože objekt `HttpRequestMessage` je po prvom odoslaní spotrebovaný a nie je možné ho opätovne použiť.

Keďže server obnovovacie tokeny pri každom obnovení rotuje, musí byť obnova na strane klienta serializovaná. Zabezpečuje to statický zámok `SemaphoreSlim`: ak dve súbežné požiadavky naraz dostanú odpoveď `401`, obnovu vykoná len prvá z nich, zatiaľ čo druhá počká a po uvoľnení zámku zistí, že prístupový token už bol medzitým obnovený, a refresh preto nezopakuje. Bez tejto synchronizácie by druhá požiadavka odoslala už zrotovaný, a teda zneplatnený obnovovací token, čo by server vyhodnotil ako pokus o znovupoužitie a používateľa by odhlásil.

Tokeny sú na zariadení uložené pomocou `SecureStorage.Default` [22] (MAUI natívne bezpečné úložisko, na Androide kryté Android Keystore) pod kľúčmi `jwt_access_token` a `jwt_refresh_token`. Trieda `TokenService` zapuzdruje všetky operácie čítania, zápisu a mazania tokenov.

4.4.2 JSON serializácia

Na serializáciu a deserializáciu dát sa používa knižnica `System.Text.Json` zabudovaná v .NET 8. Požiadavky sa odosielajú pomocou rozširovacej metódy `PostAsJsonAsync()`, ktorá automaticky serializuje C# objekt do formátu JSON a nastaví hlavičku `Content-Type: application/json`. Odpovede sa deserializujú pomocou `ReadFromJsonAsync<T>`. Väčšina DTO tried (`LoginRequest`, `LoginResponse`, `LocationUpdate`) používa predvolené

pomenovanie bez dodatočných atribútov. Výnimkou je trieda `NearbyBranch`, ktorej vlastnosti sú označené atribútom `[JsonPropertyName()]` s lowercase názvami, čím sa prispôsobujú formátu, v akom ich vracia server.

4.4.3 Ošetrenie chýb

Základným pravidlom je, že všetka sieťová komunikácia je sústredená výhradne do servisnej vrstvy (priečinok `Services/`) a žiadna služba vykonávajúca HTTP volanie nesmie prepustiť výnimku do prezentačnej vrstvy. Každá takáto metóda je obalená v bloku `try/catch` a pri chybe – či už ide o výpadok siete, časový limit alebo neúspešnú odpoveď servera – vracia bezpečnú predvolenú hodnotu (napríklad `null`, `false` alebo prázdny zoznam) namiesto vyhodenia výnimky. Jednotlivé obrazovky tak nikdy nepracujú priamo s triedou `HttpClient` ani neošetrujú sieťové výnimky a môžu sa spoľahnúť, že služba vždy vráti použiteľný výsledok.

Toto ošetrenie sa prejavuje na dvoch typických miestach. V `AuthService` sa úspešnosť HTTP odpovede overuje vlastnosťou `IsSuccessStatusCode` – metódy vracajú `bool` alebo `null` podľa výsledku, bez propagácie výnimky. Špeciálnym prípadom je odhlásenie (`LogoutAsync`), kde je `HttpRequestException` zachytená a potlačená – tokeny sa vymažú lokálne aj v prípade nedostupnosti servera, čím je zaistené čisté odhlásenie používateľa bez ohľadu na stav siete.

V `LocationTrackingService` je celý blok odosielania polohy obalený v `try/catch(Exception)`, ktorý zachytí akúkoľvek chybu – výpadok siete, chybu deserializácie aj zrušenie požiadavky – a zapíše ju do interného debug logu. Sledovanie polohy tak pokračuje v ďalšom cykle bez pádu aplikácie.

Osobitnú pozornosť si vyžaduje obnovovanie relácie (`TryRestoreSessionAsync`): pri sieťovej chybe sa uložené tokeny zámerne nemažú – odstránia sa len pri jednoznačnom zamietnutí serverom (stavový kód triedy 4xx). Vďaka tomu spustenie aplikácie bez pripojenia neodhlási používateľa s inak platnou reláciou. Ošetrenie chýb na strane komunikácie úzko súvisí s celkovou odolnosťou aplikácie voči výpadkom siete – vrátane indikácie offline stavu, offline-first prístupu k obľúbeným prevádzkam a zachytávania natívnych výnimiek na platforme Android – ktorú podrobne rozoberá časť 4.9.

4.5 Geolokácia

Geolokácia je jednou z kľúčových funkcií aplikácie PerkAround. Systém pracuje na dvoch úrovniach: mobilná aplikácia periodicky získava GPS polohu zariadenia a odosiela ju na server, ktorý na základe nej vyhľadá blízke prevádzky s aktívnymi ponukami a vyhodnotí, či má odoslať notifikáciu.

4.5.1 Získavanie polohy

Na strane mobilnej aplikácie zabezpečuje sledovanie polohy trieda `LocationTrackingService`. Poloha sa získava pomocou `Geolocation.Default.GetLocationAsync()` z knižnice `Microsoft.Maui.Devices.Sensors` s nastavenou presnosťou `GeolocationAccuracy.High` a timeoutom 10 sekúnd.

Interval aktualizácie sa dynamicky prispôsobuje stavu aplikácie: keď je aplikácia v popredí, poloha sa načítava každých 5 sekúnd, po prechode do pozadia každých 30 sekúnd. V režime šetrenia batérie sa tieto intervaly predĺžia na 20, resp. 60 sekúnd. Prepínanie medzi režimami obsluhujú metódy `OnSleep()` a `OnResume()` v `App.xaml.cs`.

Aby sa predišlo zbytočným API volaniam, platí pravidlo minimálneho pohybu: poloha sa na server odošle len v prípade, že zariadenie sa od predchádzajúceho odoslania posunulo o viac ako 50 metrov (v režime šetrenia batérie o viac ako 100 metrov). Aplikácia na odoslanie polohy vyžaduje oprávnenia `ACCESS_FINE_LOCATION` a `ACCESS_COARSE_LOCATION` deklarované v `AndroidManifest.xml` [23]. Tieto oprávnenia umožňujú získavať najpresnejšie údaje o polohe používateľa.

4.5.2 Výpočet vzdialenosti

Reálna vzdialenosť dvoch GPS súradníc sa na serveri vypočítava pomocou Haversínovho vzorca [24] implementovaného v triede `GeoHelper`. Vzorec pracuje so sférickým modelom Zeme s polomerom 6 371 000 m a určuje ortodromickú vzdialenosť, teda dĺžku najkratšieho oblúka po povrchu gule medzi dvoma bodmi. Keďže goniometrické funkcie pracujú v radiánoch, súradnice zadané v stupňoch sa najprv prevedú na radiány:

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \cdot \operatorname{atan2}\left(\sqrt{a},\, \sqrt{1-a}\right), \qquad d = r \cdot c$$

kde:

- φ_1, φ_2 – zemepisné šírky (*latitude*) prvého a druhého bodu (v radiánoch),
- λ_1, λ_2 – zemepisné dĺžky (*longitude*) prvého a druhého bodu (v radiánoch),
- $\Delta\varphi = \varphi_2 - \varphi_1$ – rozdiel zemepisných šírok,

- $\Delta\lambda = \lambda_2 - \lambda_1$ – rozdiel zemepisných dĺžok,
- a – pomocná veličina rovná haversínu stredového uhla, teda $\sin^2(c/2)$,
- c – uhlová (stredová) vzdialenosť bodov v radiánoch,
- r – polomer Zeme (6 371 000 m),
- d – výsledná vzdialenosť v metroch.

Medzivýpočet a kombinuje príspevok rozdielu zemepisných šírok a rozdielu zemepisných dĺžok, pričom váhu druhého člena určujú kosínusy oboch šírok – poludníky sa smerom k pólom zbiehajú, takže rovnaký rozdiel dĺžky tam zodpovedá kratšej vzdialenosti. Funkcia $\operatorname{atan2}(\sqrt{a}, \sqrt{1-a})$ z neho spätne určí polovicu stredového uhla, jej zdvojnásobením vznikne celý stredový uhol c a jeho vynásobením polomerom Zeme r sa uhol prevedie na dĺžku oblúka, teda na hľadanú vzdialenosť d v metroch. Tomuto postupu v kóde zodpovedajú premenné `a` a `c` a návratová hodnota `EarthRadius * c`. Výsledok slúži na presné filtrovanie pobočiek po hrubom predvýbere pomocou mriežky.

4.5.3 Filtrovanie podľa vzdialenosti

Priame výpočty vzdialenosti pre každú pobočku v databáze by boli pri väčšom množstve záznamov výkonnostne neúnosné. Systém preto využíva dvojstupňové filtrovanie implementované v `GridGeoService`.

V prvom stupni sa GPS súradnice mapujú na diskrétne dlaždice mriežky s rozlíšením 100 m × 100 m:

$$\texttt{TileX} = \left\lfloor \frac{\lambda \cdot 111320 \cdot \cos\varphi}{100} \right\rfloor, \quad \texttt{TileY} = \left\lfloor \frac{\varphi \cdot 111320}{100} \right\rfloor$$

Pre zadaný polomer vyhľadávania metóda `GetNeighborTiles()` vypočíta všetky susedné dlaždice v štvorcovej oblasti a databázový dotaz filtruje pobočky podľa ich indexov `TileX` a `TileY`, na ktorých je definovaný kompozitný index. Tým sa z celej databázy okamžite vyselektuje len malá podmnožina pobočiek.

V druhom stupni sa nad touto podmnožinou spustí presný výpočet vzdialenosti pomocou Haversínovho vzorca a pobočky mimo skutočného polomeru sa odfiltrujú v pamäti. Polomer vyhľadávania (okruh) si určuje používateľ v nastaveniach aplikácie a klient ho posiela v každej požiadavke. Ak chýba, server použije predvolených 1000 m a z bezpečnostných dôvodov ho zhora obmedzuje na 5000 m. Dotaz na blízke pobočky sa vykonáva vždy okolo aktuálnej polohy používateľa, pričom frekvenciu volaní obmedzuje už spomenuté pravidlo minimálneho pohybu na strane klienta. Poslednú známu polohu si server ukladá do tabuľky `UserLocationCache`, no aktuálne ju využíva len informatívne. Samotné vyhodnotenie a odosielanie notifikácií na základe polohy je podrobne opísané v nasledujúcej časti.

4.6 Notifikačný systém

Jednou z kľúčových vlastností aplikácie PerkAround je proaktívne upozorňovanie používateľa na zľavy v jeho okolí, teda *push* služba v zmysle služieb založených na polohe. Cieľom notifikačného systému je informovať používateľa o dostupných študentských zľavách v blízkosti jeho aktuálnej polohy bez toho, aby musel aplikáciu sám aktívne otvárať.

4.6.1 Architektúra

Notifikácie sú riešené ako *lokálne*. Aplikácia ich zobrazuje sama, kým beží, či už na popredí alebo na pozadí prostredníctvom služby opísanej ďalej. Zámerne nevyužívajú žiadnu cloudovú *push* infraštruktúru typu Firebase Cloud Messaging. Toto rozhodnutie zjednodušuje architektúru aj nasadenie, nie je potrebná registrácia v externej službe, správa push tokenov ani závislosť na treťom poskytovateľovi a pre aplikáciu, ktorá aj tak priebežne sleduje polohu, je postačujúce.

Zodpovednosť je rozdelená medzi server a klienta: server rozhoduje, o čom notifikovať, a klient notifikáciu zobrazuje. Pri každom odoslaní polohy na endpoint `POST /api/location/update` server vyhodnotí situáciu v okolí používateľa (počty dostupných ponúk, kandidátske ponuky a časové odstupy) a vráti ju v odpovedi spolu so zoznamom pobočiek. Klient z tohto podkladu vyberie konkrétny typ notifikácie, zostaví jej text a zobrazí ju natívne. Rozhodovacia logika (napríklad filtrovanie podľa preferencií) tak zostáva centrálne na serveri, zatiaľ čo prezentácia prebieha na zariadení.

4.6.2 Preferencie a rozsah notifikácií

Notifikačné preferencie sú viazané na používateľský účet, ukladajú sa v databáze a medzi zariadením a serverom sa synchronizujú cez endpointy `GET` a `PUT /api/Users/me/notifications`. Na zariadení sa zároveň držia v lokálnych preferenciách (trieda `NotificationSettingsService`) kvôli okamžitému a offline prístupu. Používateľ ich spravuje v nastaveniach aplikácie a zahŕňajú hlavný prepínač notifikácií, rozsah (o čom chce byť upozorňovaný), zoznam preferovaných kategórií a interval, počas ktorého sa notifikácie nezobrazujú.

Rozsah je reprezentovaný výčtovým typom `NotifyScope`, ktorého hodnoty musia byť zhodné na klientovi aj serveri. Rozlišujú sa štyri možnosti: „Nearby offers“ (všetky ponuky v okolí), „Select categories“ (len ponuky vo vybraných kategóriách), „Favourites only“ (len obľúbení poskytovatelia) a „Favourites + categories“ (obľúbení poskytovatelia spolu s vybranými kategóriami). Server podľa zvoleného rozsahu filtruje pobočky, ktoré prichádzajú do úvahy pre notifikáciu.

4.6.3 Spúšťanie notifikácií a obmedzovanie frekvencie

Notifikácie sú naviazané na priebežné sledovanie polohy (`Location TrackingService`) v dvoch režimoch: v režime *walk-by* sa pri posune používateľa o prahovú vzdialenosť (50 m, v režime šetrenia batérie 100 m) odošle

aktualizácia polohy, na ktorú môže server vrátiť notifikáciu o ponukách v okolí. V režime *idle* klient po dlhšej nečinnosti (2 hodiny) odošle jeden update, aby pripomenul používateľovi zľavy a ponuky v jeho okolí. Aby notifikácie neboli rušivé, uplatňuje sa globálny časový odstup (walk-by 15 minút, evidovaný v poli `User.LastNotifiedAt`, idle 3 hodiny), čas bez notifikácií (tzv. *quiet hours*) a hlavný prepínač. Keďže server pracuje v čase UTC, čas bez notifikácií vyhodnocuje klient a výsledok posiela v príznaku `NotificationsActive`. Pri vypnutých notifikáciách alebo tichom období server notifikačnú logiku úplne preskočí.

4.6.4 Vyhodnotenie a zostavenie notifikácie

Ak sú notifikácie aktívne, server v metóde `BuildNotificationInfoAsync` z pobočiek v okruhu vyhľadávania (zoradených podľa reálnej vzdialenosti) vyberie tie zodpovedajúce zvolenému rozsahu a spočíta agregované údaje: celkový počet ponúk v okolí, počet obľúbených, počty podľa kategórií a kandidátske ponuky. Tieto vráti klientovi ako objekt `NotificationInfoDto`. Samotný typ a text notifikácie potom vyberá klient (`NotificationService`), aby boli notifikácie rozmanité a situačné. Prednosť majú dva špecifické typy s vlastnými odstupmi: blízkosť obľúbenej prevádzky (do 150 m, odstup 45 minút) a zóna s vysokou hustotou ponúk (25 a viac, odstup 2 hodiny). Inak sa strieda konkrétna ponuka so súhrnom počtu dostupných ponúk. Používateľovi bez obľúbených sa občas zobrazí výzva, aby si nejaké pridal, a v režime idle súhrn typu „N deals still nearby“. Text vždy dodržiava správne množné číslo a systém nikdy nezobrazí notifikáciu s nulovým počtom. Radšej danú notifikáciu vynechá, než by zobrazil nepresný údaj.

4.6.5 Zobrazenie a beh na pozadí

Notifikácie sa zobrazujú natívne cez `NotificationCompat` v samostatnom kanáli s vysokou dôležitosťou (tzv. heads-up). Po klepnutí na notifikáciu sa aplikácia otvorí a prepne na kartu ponúk. Na Androide 13 a novšom si aplikácia vyžiada povolenie `POST_NOTIFICATIONS`. Keďže aplikácia je inštalovaná ako APK baličeк, po povolení notifikácií na inteligentných hodinkách sa zobrazujú aj tam.

Osobitným problémom bolo doručovanie notifikácií, keď je aplikácia na pozadí. Bežná časovaná slučka totiž neprežije dlho. Operačný systém (najmä Samsung One UI) proces po čase zmrazí alebo ukončí, čím by sa zastavilo sledovanie polohy aj časovač nečinnosti. Riešením je služba na pozadí (`LocationForegroundService`, tzv. foreground service typu „location“), ktorá udržiava proces nažive. Táto služba zobrazuje trvalú notifikáciu nízkej dôležitosti s možnosťou aplikáciu ukončiť a po prípadnom ukončení procesu sa automaticky obnoví. Vyžaduje príslušné oprávnenia deklarované v manifeste (`FOREGROUND_SERVICE`, `FOREGROUND_SERVICE_LOCATION`, prístup k polohe na pozadí a výnimku z optimalizácie batérie).

4.7 E-maily a newsletter

Aplikácia okrem notifikácií komunikuje s používateľom aj prostredníctvom e-mailov. Rozlišujeme transakčné e-maily (uvítací e-mail po registrácii, kód na obnovu hesla, potvrdenie zmazania účtu), ktoré sú nevyhnutnou súčasťou fungovania účtu, a voliteľné e-maily, ako pravidelný newsletter a narodeninový pozdrav, ktoré používateľ dostáva len na základe svojho súhlasu (v súlade s GDPR). Všetky e-maily využívajú spoločnú infraštruktúru opísanú nižšie.

4.7.1 Odosielanie e-mailov

Odosielanie zastrešuje rozhranie `IEmailSender` a jeho implementácia `SmtpEmailSender`, ktorá posiela e-maily cez SMTP server so šifrovaním STARTTLS. Prihlasovacie údaje SMTP sa načítavajú z konfigurácie, pričom heslo sa nastavuje mimo zdrojového kódu cez premennú prostredia. Telo e-mailu je predpripravená HTML šablóna.

Pre potreby tejto práce sa ako odosielací SMTP server využil e-mailový server Univerzity Palackého v Olomouci (Microsoft 365) spolu so študentskou e-mailovou adresou autora. Toto riešenie je postačujúce pre akademický projekt s malým počtom používateľov. V reálnom produkčnom nasadení by však bolo vhodnejšie použiť dedikovanú e-mailovú službu (napríklad špecializovaného poskytovateľa hromadného rozosielania) a vlastnú doménu odosielateľa, a to najmä kvôli limitom na počet odoslaných e-mailov, lepšej doručiteľnosti a oddeleniu od osobného účtu.

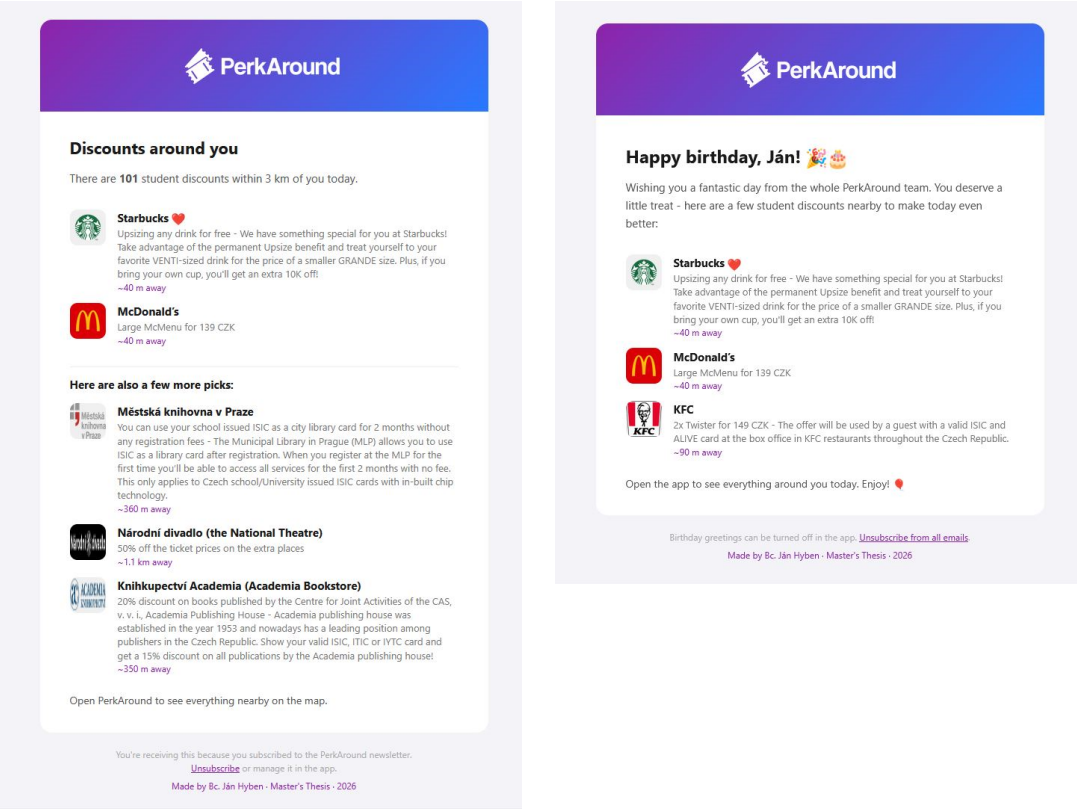
Vzhľad e-mailov je daný HTML šablónami v priečinku `Templates`. Šablóny obsahujú zástupné značky v tvare `{{...}}`, ktoré sa pred odoslaním nahradia konkrétnym obsahom (pomocná trieda `EmailTemplates`). Logo v hlavičke sa vkladá ako odkaz na URL (`/logo.png`), nie ako príloha, pretože niektorí e-mailoví klienti (napríklad Gmail) by prílohu zobrazili medzi súbormi e-mailu.

4.7.2 Plánovanie e-mailov

Newsletter aj narodeninové e-maily rozosiela dlhodobo bežiaca služba `NewsletterBackgroundService` (typu `BackgroundService`), ktorá sa v päťminútových intervaloch prebúdza a kontroluje, či nastal čas na rozoslanie. Časovanie prebieha v lokálnom čase: bežné newslettery sa rozosielajú o 08:00 (denný každý deň, týždenný v pondelok, mesačný prvého dňa v mesiaci) a narodeninové pozdravy o 10:00. Dôležitou vlastnosťou je idempotentné rozoslanie. Každá dávka je evidovaná v tabuľke `NewsletterDispatch` s unikátnym indexom nad dvojicou (druh, dátum), pričom služba si tento slot „zamkne“ ešte pred začiatkom rozosielania. Vďaka tomu sa dávka odošle najviac raz za deň aj v prípade, že sa server počas rozosielania reštartuje. Príjemcami sú používatelia s príslušne nastavenou frekvenciou newslettera. Medzi jednotlivými e-mailami je krátka pauza kvôli limitom SMTP servera a tesne pred každým odoslaním znovu overí nastavenie používateľa, keďže sa medzičasom mohol z odberu odhlásiť.

4.7.3 Personalizovaný obsah

Obsah e-mailov zostavuje trieda `NewsletterComposer` personalizovane podľa používateľa. Bežný newsletter (metóda `ComposeAsync`) obsahuje dve najbližšie ponuky vypočítané z poslednej známej polohy používateľa, oddeľovač a sekciu „more picks“ s tipmi na ponuky. Ak má používateľ zvolené preferované kategórie, tipy na ponuky sú z nich, inak ide o kombináciu ponúk s pobočkou a jednej online ponuky. Vzdialenosti rešpektujú zvolené jednotky (metrické alebo imperiálne) a počet ponúk v úvode je z okruhu s polomerom 3 km od poslednej známej polohy užívateľa. Narodeninový e-mail (metóda `ComposeBirthdayAsync`) obsahuje pozdrav oslovujúci používateľa menom a niekoľko ponúk v jeho okolí. Predmet aj text e-mailu sa líšia podľa frekvencie. Výsledná podoba oboch typov e-mailov je na obrázku 18.



Obr. 18: Ukážka e-mailov generovaných serverom: personalizovaný newsletter s ponukami v okolí (vľavo) a narodeninový pozdrav (vpravo).

4.7.4 Odhlásenie z odberu

Každý používateľ má priradený jedinečný token (`UnsubscribeToken`), pomocou ktorého sa dá z odberu odhlásiť aj bez prihlásenia. Pätička každého newslettera i narodeninového e-mailu obsahuje odkaz na verejný endpoint `GET /api/newsletter/unsubscribe`, ktorý po overení tokenu vypne používa-

teľovi newsletter aj narodeninové pozdravy a zobrazí potvrdzujúcu stránku. Rovnaké nastavenia vie používateľ kedykoľvek zmeniť aj priamo v nastaveniach aplikácie.

4.8 Nastavenia a personalizácia

Rozloženie a ovládacie prvky obrazovky nastavení sú opísané v časti 4.3.10. Táto časť sa venuje tomu, ako sú nastavenia technicky uložené a ako jednotlivé voľby ovplyvňujú správanie systému. Aplikácia rozlišuje dve skupiny nastavení: nastavenia viazané na zariadenie (aplikáciu) a nastavenia viazané na používateľský účet.

4.8.1 Dvojvrstvová perzistencia

Nastavenia lokálne pre dané zariadenie spravuje statická trieda `AppSettings` postavená nad úložiskom `Preferences` (kľúč-hodnota). Patrí sem okruh vyhľadávania, preferované kategórie ponúk, jednotky vzdialenosti, téma aplikácie, téma mapy, prepínač sledovania polohy a režim šetrenia batérie. Trieda poskytuje aj niekoľko pomocných operácií: okruh vyhľadávania automaticky zaokrúhli na najbližšiu z povolených hodnôt (posuvník v nastaveniach ponúka len pevné kroky), vlastnosťou `IsDefault` zistí, či sa aktuálne nastavenia líšia od predvolených, a metódou `RestoreDefaults` ich naraz obnoví na predvolené hodnoty. Keďže ide o nastavenia zariadenia, nie sú viazané na účet a pri prihlásení na inom zariadení sa neprenášajú.

Nastavenia, ktoré sú viazané na používateľský účet spravuje služba `NotificationSettingsService`. Zahŕňajú hlavný prepínač notifikácií, ich rozsah (`NotifyScope`), zoznam kategórií pre notifikácie, čas bez notifikácií a frekvenciu newslettera spolu s narodeninovým pozdravom. Aj tieto sa lokálne uchovávajú v úložisku `Preferences` kvôli okamžitému a offline dostupnému čítaniu, no ich zdrojom pravdy je server: po prihlásení sa stiahnu z endpointu `GET /api/Users/me/notifications` (metóda `SyncFromServerAsync`) a po každej úprave v nastaveniach sa odošlú späť metódou `PushAsync` (`PUT`). Obidve synchronizačné operácie sú ošetrené tak, aby pri výpadku siete nezhodili aplikáciu. Zmena ostáva uložená lokálne a odošle sa pri najbližšej príležitosti. Vďaka serverovému uloženiu sú tieto preferencie dostupné aj službám bežiacim na serveri, napríklad plánovaču newslettera.

Zoznam kategórií pre notifikácie je pritom oddelený od preferovaných kategórií uložených v `AppSettings`: prvý určuje, o ktorých ponukách má byť používateľ upozorňovaný, druhý ovplyvňuje poradie zobrazovaných ponúk. Ide o dve nezávislé voľby s odlišným účelom.

4.8.2 Vplyv personalizácie na správanie

Jednotlivé preferencie sa premietajú do viacerých častí systému. Okruh vyhľadávania sa po zmene neuplatní len lokálne, ale klient ho posiela serveru v každej

aktualizácii polohy, takže priamo určuje, ktoré pobočky sa považujú za blízke pri zobrazovaní ponúk aj vyhodnocovaní notifikácií. Preferované kategórie ovplyvňujú zoradenie ponúk (prevádzky z preferovaných kategórií sa zobrazujú ako prvé). Voľba jednotiek vzdialenosti je síce nastavením zariadenia, no keďže sa vzdialenosti uvádzajú aj v e-mailoch generovaných serverom, premietne sa aj do účtových nastavení a synchronizuje sa na server.

Zmena témy aplikácie sa uplatní okamžite volaním `ApplyTheme`, ktoré nastaví aktívnu tému (svetlú, tmavú alebo podľa systému), jednotlivé obrazovky potom vyberajú farby cez `AppThemeBinding`, ako opisuje časť 4.3.3. Tému mapy si používateľ volí samostatne, keďže sa vyhodnocuje osobitne pri načítaní mapového podkladu. Čas bez notifikácií sa napokon vyhodnocuje výhradne na strane klienta v lokálnom čase, pretože server pracuje v čase UTC a nepozná časové pásmo zariadenia.

4.9 Odolnosť a offline režim

Keďže aplikácia PerkAround je závislá od sieťovej komunikácie so serverom, musí sa vedieť vyrovnať s výpadkami pripojenia bez toho, aby prestala fungovať alebo spadla. Odolnosť voči výpadkom siete je riešená na troch úrovniach: viditeľnou indikáciou stavu pripojenia, offline dostupnosťou obľúbených prevádzok a dôsledným ošetrením sieťových chýb vrátane tých, ktoré vznikajú mimo spravovaného kódu.

4.9.1 Indikácia stavu pripojenia

Aktuálny stav pripojenia sleduje komponent `AppLayout` prostredníctvom udalosti `Connectivity.ConnectivityChanged`. Pri strate pripojenia sa zhora zosunie červený informačný pruh s textom „You are currently offline. Some functionality may be limited.“, ktorý používateľa upozorní, že zobrazené údaje nemusia byť aktuálne. Po obnovení pripojenia pruh nezmizne skokovo. Najprv sa prekryje zeleným pruhom „Back online!“, ktorý sa po krátkej chvíli zasunie smerom hore. Používateľ tak dostáva jednoznačnú spätnú väzbu o oboch prechodoch. Rovnakým vizuálnym spôsobom, no v oranžovej farbe, aplikácia upozorňuje aj na vypnuté sledovanie polohy.

4.9.2 Offline dostupnosť obľúbených

Obľúbené prevádzky sú navrhnuté ako *offline-first*, teda použiteľné aj bez pripojenia. Službu obľúbených (`FavoritesService`) tvorí kombinácia lokálneho úložiska a servera, ktorý je zdrojom pravdy. Identifikátory obľúbených a ešte neodoslané zmeny sa uchovávajú v úložisku `Preferences` a zoznam kariet obľúbených sa navyše ukladá do súborovej vyrovnávacej pamäte (`favorites_cache.json`). Vďaka tomu je zoznam obľúbených dostupný okamžite po štarte aj úplne bez siete.

Označenie a odznačenie obľúbenej prevádzky je optimistické, teda zmena sa v rozhraní prejaví ihneď a funguje aj offline. Ak sa zmenu nepodarí odoslať na server, zostáva v zozname neodoslaných zmien a automaticky sa dosynchronizuje pri najbližšom spustení alebo obnove relácie. Až keď sú všetky zmeny potvrdené, aplikácia preberie stav zo servera ako zdroj pravdy. V offline režime, keď server nie je dostupný, sa vzdialenosti k pobočkám výnimočne dopočítavajú priamo na zariadení z poslednej známej polohy (za bežných okolností ich počíta server).

4.9.3 Ošetrenie sieťových chýb

Aby výpadok siete nikdy nezhodil aplikáciu, dodržiava sa pravidlo, že všetka HTTP komunikácia je sústredená výhradne do servisnej vrstvy (zložka `Services/`) a žiadna metóda vykonávajúca sieťové volanie nesmie prepustiť výnimku ďalej. Každá takáto metóda je obalená v bloku `try/catch` a pri chybe vracia bezpečnú predvolenú hodnotu (napríklad `null` alebo prázdny zoznam), ako podrobnejšie opisuje časť 4.4.3. Osobitne je ošetrené obnovenie relácie: pri sieťovej chybe sa uložené tokeny nemažú (na rozdiel od zamietnutia s kódom `401`), aby offline štart aplikácie neodhlásil používateľa s inak platnou reláciou.

Samotné obalenie do `try/catch` však na platforme Android nestačí. Natívna sieťová výnimka vznikajúca v podkladovej vrstve sa totiž dokáže prepadnúť mimo spravovaného bloku `try/catch` a ukončiť proces ako nezachytená výnimka. Ako posledná poistka preto slúžia dva globálne zachytávače: `AndroidEnvironment.UnhandledExceptionRaiser` (v triede `MainActivity`) označí takúto výnimku za ošetrenú a `TaskScheduler.UnobservedTaskException` (v `MauiProgram`) ošetruje výnimky z úloh spustených bez čakania na ich výsledok. Primárnou obranou pritom zostáva úplné vyhýbanie sa sieťovým volaniam v offline stave. Periodické operácie, napríklad odosielanie polohy, sa pred každým volaním pýtajú na stav pripojenia a pri jeho absencii server vôbec nekontaktujú.

4.10 Aktualizácia aplikácie

Keďže aplikácia PerkAround je distribuovaná mimo obchodu Google Play (ako samostatný súbor APK), nemôže využívať jeho vstavaný mechanizmus aktualizácií. Aby používatelia nemuseli novú verziu zakaždým ručne vyhľadávať a znovu inštalovať, je do aplikácie zabudovaný vlastný mechanizmus vnútornej aktualizácie. Ten je rozdelený medzi server, ktorý zverejňuje informáciu o najnovšej verzii, a klienta, ktorý ju porovnáva s nainštalovanou verziou a v prípade potreby ponúkne stiahnutie a inštaláciu novej verzie.

4.10.1 Zverejnenie verzie na serveri

Informáciu o aktuálne vydanej verzii poskytuje verejný endpoint `GET /api/app/version`. Vracia číslo verzie, jej názov, adresu súboru APK

(`/downloads/perkaround.apk`), zoznam zmien (tzv. *changelog*) a príznak povinnosti aktualizácie. Tieto hodnoty sa načítavajú z konfiguračnej sekcie `AppUpdate`, takže vydanie novej verzie spočíva iba vo zvýšení čísla verzie v konfigurácii a reštarte kontajnera bez potreby opätovnej kompilácie backendu.

4.10.2 Kontrola dostupnosti aktualizácie

Na strane mobilnej aplikácie zastrešuje aktualizáciu služba `UpdateService`. Kontrola prebieha po prihlásení a následne periodicky v hodinových intervaloch. Keďže tento cyklus udržiava nažive služba na pozadí (foreground service opísaná v časti o notifikáciách), aplikácia dokáže zistiť dostupnosť novej verzie aj počas behu na pozadí. Služba porovná `versionCode` zo servera s verziou nainštalovanej aplikácie a za aktualizáciu považuje len vyššie číslo verzie. Aby sa server zbytočne nezaťažoval, opakované automatické kontroly sú obmedzené časovým odstupom.

Spôsob upozornenia závisí od toho, či je aplikácia v popredí. Ak áno, otvorí sa dialógové okno (`UpdatePage`) s názvom novej verzie a zoznamom zmien. Ak je aplikácia na pozadí, zobrazí sa notifikácia „Update available“, po klepnutí na ktorú sa dialóg otvorí. Používateľ môže aktualizáciu vyvolať aj ručne položkou „Check for updates“ v rozbaľovacej ponuke, v tom prípade sa vždy zobrazí výsledok kontroly, teda aj oznámenie, že je nainštalovaná najnovšia verzia.

4.10.3 Stiahnutie a inštalácia

Po potvrdení aktualizácie sa súbor APK stiahne do vyrovnávacej pamäte aplikácie, pričom dialóg zobrazuje ukazovateľ priebehu sťahovania. Inštalácia balíka na platforme Android vyžaduje odovzdať jeho súbor systémovému inštalátoru bezpečným spôsobom: aplikácia preto vytvorí obsahové URI (`content://`) prostredníctvom komponentu `FileProvider` a spustí systémový inštalačný zámer (tzv. *intent*) s príslušným typom obsahu a oprávnením na čítanie súboru. Na spustenie inštalácie je nutné oprávnenie `REQUEST_INSTALL_PACKAGES`. Samotnú inštaláciu následne potvrdzuje používateľ v systémovom dialógu, rovnako ako pri bežnej inštalácie APK balíčku (tzv. *sideload*).

Povinnú (kritickú) aktualizáciu nie je možné odložiť. Dialóg v takom prípade neponúka možnosť „Later“ a nedá sa zavrieť ani systémovým tlačidlom späť. Podmienkou fungovania celého mechanizmu je, aby boli všetky vydania aplikácie podpísané tým istým podpisovým kľúčom. Android inak inštaláciu aktualizácie s odlišným podpisom odmietne, čím by si vynútil úplné preinštalovanie aplikácie. Keďže je aktualizačná služba súčasťou už prvej vydanej verzie, prebehne bez nutnosti manuálneho preinštalovania aj prvá aktualizácia priamo z aplikácie.

5 Testovanie

Správnosť a kvalita systému PerkAround sa overovali priebežne počas celého vývoja. Nasledujúce časti sa venujú nástrojom a postupom použitým počas vývoja, testovaniu mobilnej aplikácie a overeniu bezpečnostných vlastností autentifikácie.

5.1 Vývojové a testovacie nástroje

Počas vývoja aplikácie PerkAround bolo testovanie priebežné a prebiehalo pomocou kombinácie nástrojov pokrývajúcich rôzne vrstvy systému: od testovania API endpointov cez správu databázy až po overovanie na mobilnom zariadení. Kľúčové nástroje (Swagger, Docker, pgAdmin) sa využívali počas celého vývoja, samotné vývojové a testovacie prostredie však počas projektu prešlo výraznou zmenou.

V prvej fáze prebiehal vývoj celý lokálne na vývojovom počítači: databáza PostgreSQL bežala v kontajneri Docker, backend sa spúšťal priamo z vývojového prostredia a mobilná aplikácia sa testovala v emulátore BlueStacks pripojenom na tento lokálny backend. Približne v polovici vývoja sa prostredie priblížilo produkčnému nasadeniu. Aplikácia sa začala nasadzovať priamo na fyzický telefón Samsung Galaxy A54 (živým nasadením cez USB) a backend bol spolu s databázou zabalený do kontajnerov a prenesený na školský server, kde beží rovnako ako v produkcii (bližšie v kapitole 6). Testovanie sa tým presunulo z emulovaného a lokálneho prostredia do reálnych podmienok: na skutočnom zariadení, so skutočnou GPS polohou a výpočtovou kapacitou. Toto rozdelenie na dve fázy sa odráža aj v opise jednotlivých nástrojov nižšie.

5.1.1 Swagger

Na interaktívne testovanie REST API bol v projekte `DiscountAPI` nakonfigurovaný Swagger UI (OpenAPI). Swagger generuje prehľadnú webovú dokumentáciu všetkých endpointov priamo z kódu a umožňuje zasielať HTTP požiadavky bez potreby externého nástroja. V konfigurácii v `Program.cs` je Swagger rozhranie rozšírené o podporu JWT autorizácie – do rozhrania možno zadať Bearer token a testovať chránené endpointy rovnako ako nechránené. Swagger bol využívaný najmä pri vývoji a ladení `AuthController`, `LocationController` a `ProvidersImportController`. [25] V nasadenom prostredí zostáva rozhranie Swagger verejne dostupné z dôvodu možnosti testovania a kontroly počas hodnotenia práce. V reálnom produkčnom nasadení by bolo vhodné ho zakázať alebo naň obmedziť prístup, keďže odhaľuje úplný prehľad endpointov API.

5.1.2 Docker

Docker sa využíval počas celého vývoja, jeho úloha sa však v priebehu projektu rozšírila. V prvej fáze slúžil na lokálne spustenie databázy PostgreSQL v izolova-

nom kontajneri, bez nutnosti inštalovať databázový server priamo na vývojový stroj. Backend bežiaci lokálne sa naň pripájal cez štandardný port. Vďaka tomu bolo vývojové prostredie reprodukovateľné a nezávislé od konfigurácie operačného systému. Prevádzkovať súčasne backend, databázu aj emulátor na jednom vývojovom sstroji sa však s rozrastajúcou sa aplikáciou ukázalo ako výkonovo neúnosné – zariadenie nestíhalo udržať všetky súčasti v plynulom chode. V druhej fáze sa preto celý systém zabalil do kontajnerov a nasadil na školský server, čím kontajnerizácia prerástla z lokálneho vývojového nástroja na platformu produkčného nasadenia. Jej podrobný opis je súčasťou kapitoly 6. [26]

Na produkčnom serveri beží systém rozdelený do štyroch samostatných kontajnerov, ktoré zhŕňa tabuľka 1.

Tabuľka 1: Bežiace kontajnery systému na serveri (výstup príkazu `docker ps`).

Kontajner	**Obraz**	**Úloha**
`discountapi-db-1`	`postgres:15`	Databáza PostgreSQL
`discountapi-api-1`	`discountapi-api`	Backendové REST API
`discountapi-caddy-1`	`caddy:2`	Reverzný proxy server a HTTPS
`discountapi-scraper-1`	`discountapi-scraper`	Služba na zber dát

5.1.3 pgAdmin

Na správu a kontrolu obsahu databázy bol použitý pgAdmin. Poskytuje grafické rozhranie pre administráciu databázy PostgreSQL. pgAdmin umožňoval priame prehliadanie tabuliek, spúšťanie SQL dotazov a overovanie správnosti dát rôzneho typu. Využíval sa najmä pri kontrole tabuliek `Providers`, `Branches`, `Offers` a `RefreshTokens` počas ladenia webscrapingu a geolokačnej logiky. [27]

5.1.4 Zariadenie s operačným systémom Android

Počas vývoja bolo potrebné aplikáciu priebežne spúšťať a overovať na platforme Android. Spočiatku bol pre tento účel používaný emulátor BlueStacks [28], ktorý umožňuje spúšťať Android aplikácie priamo na počítači s operačným systémom Windows 10. Pre potreby tejto aplikácie sa však ukázal ako nevhodný. Keďže je PerkAround založená na polohe, emulátor nedokáže vierohodne simulovať reálny pohyb používateľa, GPS senzor ani ďalšie vlastnosti mobilného zariadenia (dotykové ovládanie, systémové notifikácie, skutočný výkon). Preto pre vývojové testovanie bolo neskôr zvolené riešenie iné, a to testovanie priamo na fyzickom zariadení, konkrétne na telefóne Samsung Galaxy A54 5G s operačným systémom Android 16. Aplikácia naň bola nasadzovaná priamo z vývojového prostredia VS 2022 prostredníctvom USB ladenia (tzv. *live deployment*), čo umožnilo overovať zmeny v reálnych podmienkach vrátane skutočnej GPS polohy a notifikácií.

5.1.5 Fake GPS

Pri testovaní funkcií závislých od polohy bola okrem fyzického presunu do rôznych miest využitá aj aplikácia Fake GPS [29], ktorá umožňuje simulovať polohu zariadenia na ľubovoľnom mieste vo svete (po povolení tzv. fingovanej polohy vo vývojárskych nastaveniach systému Android). Vďaka nej bolo možné na diaľku otestovať správanie aplikácie aj v lokalitách, ktoré neboli fyzicky dostupné. Stačilo nastaviť simulovanú polohu telefónu a overiť, ako aplikácia reaguje na zobrazenie ponúk, mapu a notifikácie v danom mieste. Táto možnosť bola počas vývoja obzvlášť užitočná, keďže overovanie polohovo podmienených funkcií si inak vyžaduje fyzický pohyb. Simuláciou polohy sa tak dali rýchlo otestovať rôzne mestá a oblasti s odlišnou hustotou prevádzok bez nutnosti sa v nich fyzicky nachádzať.

5.2 Testovanie mobilnej aplikácie

Funkčné a používateľské testovanie mobilnej aplikácie overuje správne fungovanie jednotlivých obrazoviek a funkcií v reálnych podmienkach, odhaľuje chyby a prispieva k zvýšeniu používateľskej prívetivosti pred ukončením vývoja. Bezpečnostné vlastnosti autentifikácie sú testované samostatne v nasledujúcej časti.

5.2.1 Metóda testovania

Testovanie prebiehalo tromi navzájom sa doplňujúcimi spôsobmi. Prvým bolo priebežné manuálne funkčné testovanie počas vývoja, pri ktorom bola každá funkcia overovaná hneď po jej implementácii. Druhým bol neformálny používateľský test (angl. *user acceptance testing*): aplikáciu ďalej otestovali traja testeri, ktorí nepochádzajú z informatického prostredia. Tým sa nadväzuje na používateľské overovanie wireframov opísané v kapitole o návrhu a zabezpečuje sa kontinuita hodnotenia použiteľnosti z pohľadu bežného používateľa. Tretím spôsobom bolo terénne testovanie, teda reálne používanie aplikácie v exteriéri a za pohybu, ktoré je pri aplikácii založenej na polohe kľúčové.

5.2.2 Testovacie prostredie a distribúcia

Aplikácia je distribuovaná mimo obchodu Google Play, preto sa testerom inštalovala priamo zo súboru APK (tzv. *sideload*) a ďalšie verzie sa šírili cez vstavaný mechanizmus vnútornej aktualizácie. Keďže každý z testerov mal vlastný telefón, aplikácia sa prirodzene overila na viacerých modeloch s rôznymi verziami operačného systému Android a rôznymi veľkosťami a rozlíšeniami obrazoviek. Tým sa zároveň bez dodatočného úsilia overila responzivita rozhrania a správne fungovanie svetlého a tmavého režimu a ostatných UI aspektov aplikácie na odlišných zariadeniach.

Osobitná pozornosť bola venovaná terénnemu testovaniu vo viacerých mestách, konkrétne v Prahe, Brne, Olomouci a Jindřichovom Hradci. Tento výber nebol náhodný. Pri aplikácii založenej na polohe je dôležité overiť jej správanie

v rôznom prostredí: testovanie v hustom centre veľkého mesta aj v menšom meste umožnilo overiť presnosť určovania polohy v odlišných podmienkach. Jednotlivé mestá sa navyše líšia hustotou dostupných ISIC prevádzok: vo väčších (Praha, Brno) je ponúk mnoho, zatiaľ čo v menších (Jindřichův Hradec) výrazne menej, čím sa otestovali aj prázdne stavy zoznamov a správne vyhodnocovanie notifikácií v oblastiach s malým počtom prevádzok. Overilo sa aj fungovanie aplikácie na mobilných dátach a pri pohybe používateľa v teréne.

5.2.3 Testované scenáre

Testovanie pokrývalo všetky hlavné funkcie aplikácie. Pri každej obrazovke sa overovalo jej správne zobrazenie, reakcia na vstupy používateľa aj chybové stavy. Prehľad testovaných oblastí je uvedený v tabuľke 2.

Tabuľka 2: Prehľad testovaných oblastí a scenárov.

Oblasť	**Overované scenáre**
Používateľský účet	Registrácia, prihlásenie, obnova hesla, odhlásenie, zmazanie účtu
Ponuky v okolí	Zobrazenie prevádzok na mape, presnosť polohy, aktualizácia pri pohybe
Zoznam a vyhľadávanie	Zoznam ponúk, filtrovanie podľa kategórií, vyhľadávanie podľa názvu
Obľúbené	Pridanie a odobranie poskytovateľa, dostupnosť aj offline
Notifikácie	Upozornenie pri priblížení, rozsah notifikácií, tiché hodiny
Newsletter	Odber, frekvencia, obsah e-mailu, odhlásenie
Nastavenia a profil	Okruh vyhľadávania, jednotky, téma vzhľadu, úprava profilu
Odolnosť	Režim offline, výpadok siete počas používania
Aktualizácia	Vnútorná aktualizácia na novú verziu

5.2.4 Nájdené chyby a ich riešenie

Testovanie viedlo k odhaleniu viacerých chýb, ktoré boli následne opravené. Časť z nich bola ohlásená testermi, ďalšie boli odhalené pri vývojovom testovaní na reálnom zariadení a vo vydanej (*release*) verzii aplikácie. Vybrané chyby a spôsob ich riešenia sú zhrnuté nižšie:

- Zlyhanie prihlásenia vo vydanej verzii APK – automatické orezávanie nepoužívaného kódu (*trimming*), ktoré prebieha pri jej zostavovaní, odstránilo aj údaje, ktoré knižnica `System.Text.Json` potrebuje na spracovanie prihlasovacích dát. Vyriešené nastavením projektu, ktoré tieto údaje pri orezávaní zachová.

- Pád prihlásenia po preinštalovaní aplikácie – prihlasovacie údaje obnovené zo zálohy sa v bezpečnom úložisku nedali dešifrovať (chyba Android Keystore „MAC verification failed“), pretože príslušný šifrovací kľúč sa zo zálohy neobnovil. Vyriešené vypnutím zálohovania aplikácie (`allowBackup=false`).
- Pád pri výpadku internetového pripojenia – natívna sieťová výnimka sa prepadla mimo bloku `try/catch`. Vyriešené kontrolou dostupnosti pripojenia a globálnym zachytávaním výnimiek.
- Nesprávne fungujúce notifikácie a newsletter – chyby v logike ich vyhodnotenia a odosielania, opravené na základe spätnej väzby testerov.

Osobitne prínosný bol rozdiel medzi ladiacou (*debug*) a vydanou (*release*) verziou aplikácie. Niektoré chyby sa prejavili až vo vydanej verzii, pričom najvýraznejšou z nich bolo spomenuté zlyhanie prihlásenia. Jeho príčinou bolo orezávanie nepoužívaného kódu (tzv. *trimming*), ktoré prebieha automaticky pri zostavovaní vydanej verzie s cieľom zmenšiť výsledný balík APK. Trimming však odstránil aj metadáta, na ktoré sa pri reflexii spolieha knižnica `System.Text.Json`, takže prevod prihlasovacích údajov medzi objektmi jazyka C# a formátom JSON za behu zlyhal a prihlásenie skončilo chybou. Riešením bolo explicitné zapnutie reflexie knižnice `System.Text.Json` v konfigurácii projektu, čím sa potrebné metadáta pri orezávaní zachovali. Táto skúsenosť potvrdila dôležitosť testovania priamo na tej verzii aplikácie, ktorá sa reálne distribuuje používateľom, a nielen na ladiacej verzii z vývojového prostredia.

5.2.5 Spätná väzba a použiteľnosť

Chyby a podnety testeri hlásili priebežne, osobne alebo správou. Vznikla tak neformálna slučka spätnej väzby. Nahlásený problém bol opravený a nová verzia im bola obratom sprístupnená cez mechanizmus aktualizácie, čo umožnilo rýchle iterácie bez opätovnej manuálnej inštalácie. Okrem samotných chýb prispeli testeri aj podnetmi na zrozumiteľnosť rozhrania, napríklad na pomenovanie tlačidiel a usporiadanie prvkov. Tieto pripomienky boli zapracované s cieľom čo najväčšej univerzálnej používateľskej prívetivosti.

5.2.6 Zhrnutie

Po zapracovaní opráv fungovala aplikácia stabilne naprieč rôznymi zariadeniami aj mestami a testeri ju dokázali bez problémov ovládať. Testovanie malo aj svoje obmedzenia: prebiehalo manuálne, bez sady automatizovaných testov, a zapojená vzorka používateľov bola malá. Tieto obmedzenia sú bližšie rozobraté v časti venovanej limitom riešenia.

5.3 Testovanie bezpečnosti

Bezpečnosť autentifikačného mechanizmu opísaného v častiach o návrhu a implementácii autentifikácie bola overená sériou cielených scenárov, ktoré preverujú správanie serveru aj klienta pri práci s tokenmi. Testovanie prebiehalo manuálne, prevažne prostredníctvom rozhrania Swagger, ktoré umožňuje zasielať požiadavky s ľubovoľným tokenom v hlavičke, a pozorovaním reakcie aplikácie na reálnom zariadení. Cieľom bolo potvrdiť, že chránené zdroje sú dostupné výhradne s platným prístupovým tokenom a že systém korektne reaguje na vypršané, pozmenené aj chýbajúce tokeny.

5.3.1 Expirácia tokenu

Prvý scenár overuje životný cyklus prístupového tokenu po vypršaní jeho platnosti. Vypršanie bolo možné pozorovať buď po uplynutí štandardnej 15-minútovej platnosti tokenu, alebo rýchlejšie po jej dočasnom skrátení v konfigurácii servera. Na požiadavku s takýmto vypršaným tokenom server odpovedal stavovým kódom `401 Unauthorized`, keďže parameter `ValidateLifetime` vyhodnotí čas expirácie (`exp`) ako neplatný.

Následne sa overilo automatické obnovenie na strane klienta. Pri behu aplikácie s vypršaným prístupovým tokenom klient (`AuthHeaderHandler`) po prijatí odpovede `401` transparentne odoslal obnovovací token na endpoint `/api/auth/refresh`, získal nový pár tokenov a pôvodnú požiadavku automaticky zopakoval. Z pohľadu používateľa prebehla operácia bez prerušenia a bez nutnosti opätovného prihlásenia. Overilo sa tiež, že po vypršaní alebo zneplatnení obnovovacieho tokenu server jeho obnovenie odmietne a aplikácia používateľa odhlási.

Otestovala sa aj detekcia zneužitia pri rotácii tokenov: opätovné predloženie obnovovacieho tokenu, ktorý už bol pri predchádzajúcej obnove zrotovaný (a teda zneplatnený), server vyhodnotil ako pravdepodobnú krádež a zneplatnil všetky aktívne obnovovacie tokeny daného používateľa, čím ho odhlásil zo všetkých zariadení.

5.3.2 Neplatný token

Druhý scenár preveruje odolnosť voči neplatným a úmyselne pozmeneným tokenom. Do hlavičky `Authorization` boli postupne vložené tri druhy chybných tokenov: náhodný reťazec bez štruktúry JWT, inak platný token s pozmeneným obsahom, napríklad s upraveným identifikátorom používateľa a token s ručne prepísaným podpisom. Server všetky tri odmietol s kódom `401 Unauthorized`. Rozhoduje o tom parameter `ValidateIssuerSigningKey`: server pri overovaní z doručenej hlavičky a obsahu sám znova vypočíta podpis (algoritmom HMAC SHA-256 pomocou svojho tajného kľúča) a porovná ho s podpisom priloženým v tokene. Akákoľvek zmena obsahu preto vedie k nezhode oboch podpisov a token je zamietnutý. Podpisový kľúč je pritom uložený výhradne na serveri,

takže útočník ho nepozná a platný podpis pre pozmenený token nedokáže vytvoriť. Osobitne sa overilo aj overovanie vydavateľa a príjemcu. Token so správnym podpisom, no s iným vydavateľom (`iss`) alebo príjemcom (`aud`), než aké server očakáva („discount-api“ a „discount-client“), bol takisto zamietnutý, a to vďaka parametrom `ValidateIssuer` a `ValidateAudience`. Potvrdilo sa tým, že server neprijme token vydaný iným systémom, aj keby bol inak korektne podpísaný.

5.3.3 Neoprávnený prístup

Tretí scenár overuje, že chránené endpointy nie sú prístupné bez platnej autentifikácie. Požiadavky na endpointy označené atribútom `[Authorize]` (napríklad `GET /api/users/me` či `GET /api/favorites`) boli odoslané úplne bez hlavičky `Authorization`. Server ich vo všetkých prípadoch odmietol s kódom `401 Unauthorized` a nevrátil žiadne dáta. Naopak, verejné endpointy (prihlásenie, registrácia, obnovenie tokenu) zostali dostupné aj bez tokenu, v súlade s atribútom `[AllowAnonymous]`.

Overilo sa tiež to, že importný endpoint, ktorý namiesto tokenu JWT chráni vlastný atribút `[ImportApiKey]`, odmietne požiadavku bez správneho kľúča v hlavičke. Tým je import dát zo scrapingovej služby oddelený od bežnej používateľskej autentifikácie a nedá sa vyvolať platným používateľským tokenom.

Overené scenáre a zodpovedajúce reakcie systému možno zhrnúť takto:

- Požiadavka s vypršaným prístupovým tokenom je zamietnutá s kódom `401 Unauthorized`. Klient však na pozadí automaticky vymení obnovovací token za nový pár tokenov a pôvodnú požiadavku zopakuje, takže operácia z pohľadu používateľa prebehne bez prerušenia.
- Ak je vypršaný alebo zneplatnený už samotný obnovovací token, jeho výmena zlyhá s kódom `401`. Platnú reláciu už nemožno predĺžiť, preto aplikácia používateľa odhlási a vyžiada prihlásenie.
- Opätovné predloženie už zrotovaného obnovovacieho tokenu server vyhodnotí ako pravdepodobnú krádež a zneplatní všetky aktívne obnovovacie tokeny používateľa, čím ho odhlási zo všetkých zariadení.
- Token s pozmeneným obsahom alebo neplatným podpisom je zamietnutý s kódom `401`, keďže zlyhá overenie podpisu.
- Požiadavka na chránený endpoint bez hlavičky `Authorization` je zamietnutá s kódom `401` a server nevráti žiadne dáta, takže chránené zdroje zostávajú neprístupné bez platnej autentifikácie.
- Požiadavka na importný endpoint bez platného kľúča je odmietnutá.

6 Nasadenie systému

Nasadenie systému PerkAround do produkčného prostredia zahŕňa serverové prostredie, na ktorom systém beží, zabalenie jednotlivých komponentov do kontajnerov a ich orchestráciu, zabezpečenie komunikácie protokolom HTTPS, správu konfigurácie a citlivých údajov, ako aj distribúciu mobilnej aplikácie koncovým používateľom. Cieľom nasadenia bolo sprevádzkovať celý systém (API, databázu aj službu na zber dát) ako ucelený, reprodukovateľný a jednoducho aktualizovateľný celok.

6.1 Serverové prostredie

Serverová časť systému je nasadená na virtuálnom serveri poskytnutom Katedrou informatiky Univerzity Palackého v Olomouci. Server hostí všetky serverové komponenty systému, teda backend (REST API), databázu PostgreSQL, reverzný proxy server aj službu na automatizovaný zber dát. Používateľský účet pridelený autorovi disponuje oprávneniami `sudo`, čo umožnilo samostatne spravovať operačný systém, inštalovať softvérové balíky a konfigurovať sieťové služby bez zásahu správcu infraštruktúry.

Na serveri beží operačný systém Ubuntu Server 24.04 LTS [30]. Jedná sa o stabilný operačný systém s dlhodobou podporou (*Long Term Support*) a pravidelné bezpečnostné aktualizácie. Ide o virtuálny stroj prevádzkovaný platformou KVM (*Kernel-based Virtual Machine*). Serveru boli pridelené štyri virtuálne procesorové jadrá odvodené od fyzického procesora Intel Xeon Silver 4314, približne 3,8 GiB operačnej pamäte a virtuálny disk s kapacitou 35 GiB. Tieto prostriedky sa počas prevádzky ukázali ako dostatočné – súbežné spracovanie HTTP požiadaviek, komunikácia s databázou aj chod pomocných služieb využívali len časť dostupnej pamäte aj výpočtového výkonu.

Server je dostupný cez verejnú adresu IPv4, vďaka čomu sa naň mobilná aplikácia pripája priamo cez internet bez ďalších sprostredkujúcich serverov. Je na ňom nainštalovaná platforma Docker [26], ktorá tvorí základ nasadenia opísaného v ďalších častiach. Kľúčové technické parametre serverového prostredia zhŕňa tabuľka 3.

Tabuľka 3: Technické parametre serverového prostredia.

Parameter	**Hodnota**
Typ servera	Virtuálny server
Virtualizačná platforma	KVM (Kernel-based Virtual Machine)
Operačný systém	Ubuntu Server 24.04.4 LTS
Jadro systému	Linux 6.8.0-49-generic
Architektúra	x86-64
Procesor (CPU)	Intel Xeon Silver 4314 @ 2,40 GHz
Počet virtuálnych CPU	4 vCPU
Operačná pamäť (RAM)	3,8 GiB
Disk	35 GiB
Typ tabuľky oddielov	GPT
Hlavný oddiel	`/dev/sda2`
Verejná IPv4 adresa	158.194.80.103
Oprávnenia	`sudo`
Kontajnerizačná platforma	Docker

6.2 Kontajnerizácia

Všetky serverové komponenty systému sú zabalené do kontajnerov a spravované nástrojom Docker Compose, ktorý ich definuje, prepája a spúšťa ako jeden celok. Kontajnerizácia zaručuje reprodukovateľné prostredie nezávislé od konfigurácie hostiteľského systému, izoluje závislosti jednotlivých služieb a zjednodušuje nasadzovanie nových verzií. Systém je zložený zo štyroch služieb definovaných v súbore `docker-compose.yml`:

- `db` – databázový server PostgreSQL, ktorý ukladá dáta do perzistentného zväzku (volume). Jeho port je zverejnený len na lokálne rozhranie servera (`127.0.0.1`), takže databáza nie je prístupná priamo z internetu, ale výhradne z ostatných kontajnerov.
- `api` – backendové REST API na platforme ASP.NET Core 8. Nemá zverejnený žiadny port navonok – z internetu je dostupné výhradne cez reverzný proxy server, inak len interne v sieti kontajnerov. Profilové fotografie a súbor APK sú uložené vo zväzkoch, takže prežijú prestavbu kontajnera.
- `caddy` – reverzný proxy server Caddy, ktorý ako jediný zverejňuje porty 80 a 443 a zabezpečuje šifrovanie HTTPS. Podrobnejšie ho opisuje časť 6.3.
- `scraper` – služba na automatizovaný zber dát. Využíva knižnicu Playwright a so zvyškom systému komunikuje interne cez API.

Jednotlivé služby sú prepojené spoločnou privátnou sieťou, v rámci ktorej sa navzájom adresujú menom služby (napríklad `db` alebo `api`) namiesto IP adries. Perzistentné dáta, ako sú obsah databázy, profilové fotografie, súbor APK na stiahnutie aj TLS certifikáty, sú uchovávané v pomenovaných zväzkoch, čím sú oddelené od životného cyklu kontajnerov a prežijú ich reštart aj prestavbu.

Backendový obraz sa zostavuje vo dvoch fázach (tzv. *multi-stage build*): v prvej sa v obraze s .NET SDK aplikácia skompiluje a publikuje, v druhej sa výsledok skopíruje do menšieho behového obrazu obsahujúceho iba .NET runtime. Výsledný obraz je tak menší a neobsahuje nástroje potrebné len na kompiláciu. Do behového obrazu je navyše doinštalovaný balík časových pásiem a nastavené pásmo Europe/Bratislava, aby plánovač newslettera pracoval v správnom lokálnom čase. Bez neho by kontajner bežal v čase UTC a e-maily by sa rozosielali s časovým posunom.

Pri štarte API kontajnera sa automaticky aplikujú databázové migrácie. Keďže databázový kontajner nemusí byť v tom okamihu ešte pripravený prijímať spojenia, API sa o pripojenie pokúša opakovane, a to až dvanásťkrát s exponenciálne narastajúcim odstupom, čím sa ošetrí poradie spúšťania služieb. Po úspešnom pripojení sa schéma databázy zosúladí s aktuálnym dátovým modelom bez nutnosti ručného zásahu.

6.3 Reverzný proxy server a zabezpečenie HTTPS

Aby komunikácia medzi mobilnou aplikáciou a serverom prebiehala šifrovane, je pred backendom umiestnený reverzný proxy server Caddy [31]. Ten prijíma prichádzajúce požiadavky HTTPS na porte 443, ukončuje na nich šifrovanie TLS a preposiela ich backendovému kontajneru interne po nešifrovanom HTTP v rámci privátnej siete kontajnerov. Vďaka tomu sa samotný backend nemusí zaoberať správou certifikátov ani šifrovaním a zostáva jednoduchý, zatiaľ čo navonok je všetka komunikácia šifrovaná. Toto usporiadanie (tzv. ukončenie TLS na proxy) je bezpečné za predpokladu, že privátna sieť medzi proxy serverom a backendom je dôveryhodná. V našom prípade ide o izolovanú internú sieť kontajnerov na jednom hostiteľovi. V náročnejšom produkčnom prostredí, kde by komponenty bežali na viacerých uzloch, by bolo vhodné šifrovať aj túto internú komunikáciu (napríklad vzájomným TLS).

Hlavnou výhodou servera Caddy je automatická správa TLS certifikátov. Pri prvom spustení si sám vyžiada dôveryhodný certifikát od certifikačnej autority Let's Encrypt [32] a následne ho pred vypršaním platnosti automaticky obnovuje, takže certifikáty nie je potrebné spravovať ručne. Jeho konfigurácia je pritom minimalistická – postačuje uviesť doménové meno a cieľovú službu, ktorej sa majú požiadavky preposielať.

Certifikačné autority ako Let's Encrypt vydávajú certifikáty len pre doménové mená. Serveru preto bolo potrebné priradiť doménové meno. Na tento účel bola použitá bezplatná služba dynamického DNS DuckDNS [33], prostredníctvom ktorej bolo zaregistrované meno `perkaround.duckdns.org` smerujúce

na IP adresu servera. Toto meno zároveň tvorí základ produkčnej adresy API (`https://perkaround.duckdns.org/api/`), na ktorú sa pripája mobilná aplikácia.

Nasadenie za reverzným proxy si vyžiadalo drobnú úpravu na strane backendu. Keďže spojenie od proxy servera k backendu prebieha po HTTP, aplikácia by inak vyhodnotila požiadavku ako nešifrovanú a absolútne adresy, ktoré generuje (napríklad odkazy na profilové fotografie alebo na logo v e-mailoch), by tvorila s predponou `http`. Mobilná aplikácia má však nešifrovanú komunikáciu z bezpečnostných dôvodov zakázanú a takéto adresy by odmietla načítať. Riešením bolo odovzdať backendu informáciu, ktorá sa pri prechode na interné HTTP stráca – totiž že klient sa pôvodne pripojil cez HTTPS. Caddy ju pripája ku každej preposlanej požiadavke v podobe štandardných hlavičiek `X-Forwarded-*`, ktoré nesú pôvodný protokol (`https`), pôvodné doménové meno a IP adresu klienta. Backend je nastavený tak, aby týmto hlavičkám dôveroval a riadil sa nimi, takže vie, že pôvodná požiadavka bola šifrovaná, a absolútne adresy potom skladá správne s predponou `https`. Presmerovanie z nešifrovaného `http` na `https` pritom zabezpečuje Caddy automaticky.

6.4 Konfigurácia a tajné údaje

Konfigurácia backendu je usporiadaná do viacerých vrstiev, ktoré sa navzájom prekrývajú podľa priority. Základné, necitlivé nastavenia (napr. názvy vydavateľa a príjemcu tokenu, doby platnosti tokenov, časové pásmo či adresa SMTP servera) sú uložené v súbore `appsettings.json`, ktorý je súčasťou zdrojového kódu. Citlivé údaje, teda podpisový kľúč JWT, importný kľúč a heslo k e-mailovému účtu, však v tomto súbore zámerne chýbajú a do aplikácie sa dopĺňajú až v mieste behu.

V produkčnom nasadení sa tieto citlivé hodnoty odovzdávajú kontajneru prostredníctvom premenných prostredia definovaných v konfigurácii Docker Compose (v súbore `docker-compose.yml`). Tie majú prednosť pred hodnotami v `appsettings.json`. Rovnakým spôsobom sa nastavuje aj pripojovací reťazec k databáze alebo adresa a číslo verzie súboru APK pre mechanizmus vnútornej aktualizácie, čo umožňuje meniť nasadenie bez opätovnej kompilácie aplikácie.

Vďaka tomu nie sú v aplikačnom zdrojovom kóde uložené žiadne zneužiteľné tajné údaje. Pre ešte prísnejšie produkčné nasadenie by bolo vhodné využiť dedikovaný nástroj na správu tajomstiev (napríklad Docker secrets alebo špecializovaný trezor), ktorý by citlivé hodnoty sprístupňoval kontajnerom bez toho, aby sa nachádzali v konfiguračných súboroch.

6.5 Distribúcia aplikácie

Mobilná aplikácia PerkAround nie je distribuovaná prostredníctvom Android obchodu Google Play, ale ako samostatný inštalačný súbor APK (Android Package), ktorý si používateľ nainštaluje priamo do zariadenia (tzv. *sideload*). Tento spôsob bol pre akademický projekt s malým počtom používateľov postačujúci a nevyžadoval registráciu vývojárskeho účtu ani schvaľovací proces obchodu.

Súbor APK servíruje priamo backend na adrese odvodenej od produkčnej domény (`/downloads/perkaround.apk`). Fyzicky je uložený v priečinku na hostiteľskom serveri, ktorý je do API kontajnera pripojený ako zväzok, takže novú verziu možno nasadiť jednoduchou výmenou súboru bez prestavby kontajnera. Aby statická vrstva servera dokázala súbor korektne odovzdať, bol pre príponu `.apk` doplnený zodpovedajúci typ obsahu (MIME). Bez neho by server na požiadavku o stiahnutie odpovedal chybou.

Šírenie nových verzií zabezpečuje mechanizmus vnútornej aktualizácie, ktorému sa venuje časť 4.10. Podmienkou aktualizácie bez nutnosti preinštalovania je, aby boli všetky vydania aplikácie podpísané tým istým podpisovým kľúčom. Android inak inštaláciu aktualizácie s odlišným podpisom odmietne. Každé nové vydanie zároveň zvyšuje číslo verzie (`versionCode`), podľa ktorého server aj aplikácia rozpoznajú dostupnosť novšej verzie.

7 Výsledky práce

Výsledkom tejto diplomovej práce je funkčný a do produkčného prostredia nasadený systém PerkAround, ktorý študentom sprístupňuje zľavy v ich okolí a aktívne ich upozorňuje na ponuky v bezprostrednej blízkosti. Systém tvoria tri samostatné, no vzájomne prepojené časti: mobilný klient pre platformu Android (*DiscountAppMAUI*), serverové REST API (*DiscountApi*) a služba na automatizovaný zber dát (*IsicScraperService*). Nasledujúce časti vyhodnocujú splnenie stanovených cieľov, pomenúvajú známe obmedzenia riešenia a načrtávajú možné smery ďalšieho rozvoja projektu.

7.1 Splnenie cieľov

Všetky čiastkové ciele stanovené v úvode práce sa podarilo naplniť:

- *Analýza problematiky.* Boli rozobraté služby založené na polohe (LBS) vrátane rozdielu medzi *pull* a *push* prístupom, oblasť študentských zliav, existujúce riešenia (webová stránka ISIC a aplikácia Alive App) aj princípy autentifikácie tokenmi a architektúry REST. Z identifikovaných nedostatkov existujúcich riešení bola odvodená hlavná pridaná hodnota vlastného systému, a to proaktívne, polohovo podmienené upozorňovanie na zľavy.
- *Návrh systému.* Boli definované funkčné aj nefunkčné požiadavky a navrhnutá trojvrstvová architektúra vrátane konceptuálneho dátového modelu (ER diagram) a autentifikačného mechanizmu založeného na kombinácii JWT a obnovovacích tokenov.
- *Serverové REST API.* Backend bol implementovaný v prostredí ASP.NET Core 8 s databázou PostgreSQL a ORM Entity Framework Core. Efektívne vyhľadávanie prevádzok podľa polohy zabezpečuje dvojstupňové filtrovanie.
- *Automatizovaný zber dát.* Samostatná služba *IsicScraperService* postavená na knižnici Playwright pravidelne získava dáta o prevádzkach a zľavách z portálu ISIC a transakčne, idempotentne ich synchronizuje do databázy cez chránený importný endpoint.
- *Mobilný klient.* Klient postavený na frameworku .NET MAUI umožňuje zobrazovať ponuky na interaktívnej mape aj v zozname, vyhľadávať a filtrovať ich podľa kategórií, spravovať obľúbené prevádzky aj celý používateľský účet vrátane profilu a nastavení.
- *Bezpečná autentifikácia.* Prihlásenie je riešené krátkodobými JWT *access* tokenmi a rotujúcimi *refresh* tokenmi, ktoré server uchováva len v podobe hašu a pri ktorých deteguje znovupoužitie. Citlivé údaje sú na zariadení chránené úložiskom Android Keystore.

- *Súlad s GDPR.* Systém spracúva len nevyhnutné osobné údaje, rozlišuje právne základy spracúvania, obsahuje úplné zásady ochrany osobných údajov a napĺňa právo na výmaz účtu a všetkých súvisiacích záznamov.
- *Testovanie a nasadenie.* Riešenie bolo priebežne a terénne otestované a nasadené do produkčného prostredia na školskom serveri. Je kontajnerizované pomocou nástroja Docker Compose, so šifrovanou komunikáciou cez HTTPS prostredníctvom reverzného proxy servera Caddy a certifikátu Let's Encrypt.

Hlavný cieľ, teda funkčná Android aplikácia proaktívne upozorňujúca na zľavy v okolí používateľa, bol tak naplnený a systém je reálne prevádzkovaný.

7.2 Limity riešenia

Napriek splneniu vytýčených cieľov má výsledné riešenie aj známe obmedzenia, ktoré vyplývajú najmä z jeho charakteru ako časovo ohraničeného akademického projektu:

- *Rozsah testovania.* Testovanie prebiehalo manuálne, bez sady automatizovaných testov, a zapojená vzorka používateľov bola malá.
- *Lokálne notifikácie.* Notifikácie zobrazuje samotná aplikácia, kým beží. Jej beh na pozadí síce udržiava služba typu *foreground service*, no pri agresívnej správe spotreby energie niektorých výrobcov môže byť proces napriek tomu obmedzený a po úplnom ukončení aplikácie upozornenia neprídu.
- *Zameranie na platformu Android.* Hoci je .NET MAUI multiplatformový, klient bol vyvinutý a otestovaný výhradne pre Android. Verzia pre iOS nebola realizovaná.
- *Závislosť od jediného zdroja dát.* Dáta pochádzajú výlučne z portálu `isic.cz`. Webscrapingová služba je citlivá na zmeny štruktúry stránky a rozsah zliav je obmedzený na ponuky uvedené na tomto portáli.
- *Distribúcia mimo obchodu.* Aplikácia sa šíri ako súbor APK (*sideload*), čo vyžaduje povolenie inštalácie z neznámych zdrojov a neposkytuje organickú objaviteľnosť ani dôveru obchodu Google Play.
- *Odosielanie e-mailov.* Transakčné e-maily aj newsletter sa odosielajú cez školský účet Microsoft 365, ktorého limity a doručiteľnosť nie sú vhodné pre veľký počet používateľov a nasadenie v reálnom svete.
- *Škálovanie a bezpečnostné spevnenie.* Systém beží na jednom serveri, databázové migrácie sa spúšťajú pri štarte a niektoré dotazy načítavajú väčšie množiny záznamov do pamäte. Na autentifikačných endpointoch navyše nie je zavedené obmedzovanie počtu pokusov (tzv. *rate limiting*) a tajné údaje sa v nasadení odovzdávajú cez premenné prostredia namiesto dedikovaného správcu tajomstiev.

7.3 Možné rozšírenia

Systém PerkAround ponúka viacero prirodzených smerov ďalšieho rozvoja, ktoré nadväzujú na súčasnú funkcionalitu aj na uvedené obmedzenia:

- *Podpora platformy iOS.* Keďže mobilný klient je postavený na multiplatformovom frameworku .NET MAUI, s primeraným úsilím pri platformovo špecifických častiach (poloha, notifikácie, mapa) by bolo možné vydať aj verziu pre operačný systém iOS a osloviť tak širší okruh používateľov.
- *Serverom riadené push notifikácie.* Nahradenie lokálnych notifikácií cloudovou push infraštruktúrou (napríklad Firebase Cloud Messaging) by umožnilo doručovať upozornenia aj bez neustále bežiacej aplikácie a znížiť spotrebu batérie, v kombinácii s mechanizmom získavania polohy na pozadí.
- *Rozšírenie zdrojov dát.* Doplnenie ďalších zdrojov zliav popri portáli ISIC, prípadne samoobslužné pridávanie ponúk zo strany prevádzok cez administrátorské rozhranie, by systém zbavilo závislosti od jediného portálu a rozšírilo ponuku zliav.
- *Publikovanie do Google Play a automatizované testy.* Oficiálna distribúcia v obchode (objaviteľnosť, dôvera, automatické aktualizácie) spolu so sadou jednotkových, integračných a UI testov spúšťaných v rámci priebežnej integrácie (CI/CD) by zvýšila spoľahlivosť aj dosah aplikácie.
- *Bezpečnostné spevnenie.* Zavedenie obmedzovania počtu pokusov (*rate limiting*) na autentifikačných endpointoch, použitie dedikovaného správcu tajomstiev a overovanie e-mailovej adresy pri registrácii by ďalej zvýšili odolnosť systému.
- *Personalizované odporúčania.* Inteligentnejší výber zobrazovaných ponúk a cielenie notifikácií nad rámec samotnej vzdialenosti, napríklad na základe obľúbených kategórií prevádzok (reštaurácie, kaviarne), histórie či hodnotení používateľov. Tento prístup by zvýšil relevantnosť upozornení.

Záver

Cieľom tejto diplomovej práce bolo navrhnúť a implementovať mobilnú aplikáciu pre platformu Android, ktorá študentom sprístupní prehľad zliav dostupných v ich bezprostrednom okolí a na základe ich aktuálnej polohy ich upozorní na relevantné ponuky. Motiváciou bola skutočnosť, že informácie o študentských zľavách sú roztrúsené medzi rôzne zdroje, a študenti tak o mnohých ponukách vôbec nevedia, hoci na ne majú nárok.

Práca najprv analyzovala problematiku mobilných služieb založených na polohe, oblasť študentských zliav a existujúce riešenia, konkrétne webovú stránku ISIC a aplikáciu Alive App. Z analýzy vyplynulo, že tieto nástroje fungujú na princípe aktívneho vyhľadávania a samy od seba na zľavy v okolí neupozorňujú. Práve v tom spočíva hlavný prínos navrhnutého systému: proaktívne, polohovo podmienené upozorňovanie na zľavy bez toho, aby ich používateľ musel sám vyhľadávať.

Na základe analýzy bol navrhnutý a implementovaný ucelený systém PerkAround, ktorý pozostáva z troch častí: mobilného klienta vyvinutého vo frameworku .NET MAUI, serverového REST API v prostredí ASP.NET Core s databázou PostgreSQL a samostatnej služby na automatizovaný zber dát z portálu ISIC postavenej na knižnici Playwright. Systém zobrazuje ponuky na mape aj v zozname, umožňuje ich vyhľadávať, filtrovať a ukladať medzi obľúbené, spravovať používateľský účet a priebežne upozorňuje na zľavy v okolí. Bezpečnosť je zabezpečená autentifikáciou pomocou tokenov JWT v kombinácii s rotujúcimi obnovovacími tokenmi a spracúvanie osobných údajov je v súlade s nariadením GDPR.

Výsledné riešenie bolo riadne otestované, a to priebežne pri vývoji aj používateľmi v teréne vo viacerých mestách, a následne nasadené do produkčného prostredia na serveri katedry, kde je kontajnerizované pomocou služby Docker a komunikuje šifrovane cez protokol HTTPS. Všetky stanovené ciele práce sa podarilo naplniť a systém je reálne prevádzkovaný. Známe obmedzenia riešenia a možné smery jeho ďalšieho rozvoja, ako je podpora platformy iOS, serverom riadené push notifikácie či rozšírenie zdrojov dát, sú podrobnejšie rozobraté v kapitole venovanej výsledkom práce.

Conclusions

The objective of this Master thesis was to design and implement a mobile application for the Android platform that enables students to discover discounts available in their immediate surroundings and notifies them about relevant offers based on their current location. The motivation for this work stems from the fact that information about student discounts is scattered across multiple sources, causing students to remain unaware of many offers to which they are entitled.

The thesis first analyzes the field of location-based mobile services, the domain of student discounts, and existing solutions, specifically the ISIC website and the Alive App. The analysis revealed that these solutions rely on active user searches and do not proactively notify users about nearby discounts. This represents the main contribution of the proposed system: proactive, location-based notifications about nearby discounts without requiring users to search for them manually.

Based on the analysis, a comprehensive system named PerkAround was designed and implemented. The system consists of three main components: a mobile client developed using the .NET MAUI framework, a REST API built with ASP.NET Core and PostgreSQL, and a standalone service for automated data collection from the ISIC portal implemented using the Playwright library. The system displays discount offers on both a map and in a list, enables users to search, filter, and save offers as favorites, manage their user accounts, and continuously receive notifications about nearby discounts. Security is ensured through JWT-based authentication combined with rotating refresh tokens, while the processing of personal data complies with the requirements of the General Data Protection Regulation (GDPR).

The resulting solution was thoroughly tested throughout the development process and subsequently by users in real-world conditions across multiple cities. It was then deployed to the production environment on a university department server, where it runs in Docker containers and communicates securely over the HTTPS protocol. All objectives of the thesis were successfully achieved, and the system is currently in active operation. The identified limitations of the proposed solution and potential directions for future development, including support for the iOS platform, server-driven push notifications, and the integration of additional data sources, are discussed in detail in the chapter dedicated to the results of the thesis.

A Obsah elektronických dát

Text práce, zdrojové kódy a všetky ostatné prílohy, ktoré sú súčasťou diplomovej práce boli riadne odovzdané prostredníctvom elektronického systému katedry informatiky.

`install/`
: Adresár so súborom formátu *Android Package* (APK) potrebného na korektnú inštaláciu aplikácie *PerkAround*.

`source/`
: Adresár, v ktorom sú umiestnené zdrojové kódy celého systému *PerkAround*, tz. mobilného klienta aj serverových komponentov, teda REST API, databázy a služby na automatizovaný zber dát.

`thesis/`
: Adresár obsahujúci text diplomovej práce vo formáte PDF spolu so .zip archívom, kde sú zabalené zdrojové LaTeX kódy generujúce tento text, obrázky a ďalšie prílohy. Text diplomovej práce je vytvorený s použitím záväzného štýlu katedry informatiky pre záverečné práce.

`README.md`
: Súbor, ktorý obsahuje inštrukcie pre korektnú inštaláciu aplikácie *PerkAround*, jej spustenie a použitie.

Literatúra

[1] *ISIC Association. : ISIC Card* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://www.isic.org/cards/isic/⟩.

[2] Steiniger, Stefan; Neun, Moritz; Edwardes, Alistair. *Foundations of Location Based Services: Lecture Notes on LBS, V. 1.0* [online]. 2006 [cit. 2026-7-1]. Dostupné z: ⟨https://www.spatial.cs.umn.edu/Courses/Fall11/8715/papers/IM7_steiniger.pdf⟩.

[3] *GPS.gov. : GPS* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://www.gps.gov/⟩.

[4] *ISIC.cz. : Slevy pro studenty* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://www.isic.cz/slevy⟩.

[5] *Alive App. : Google Play* [online]. [cit. 2026-7-10]. Dostupné z: ⟨https://play.google.com/store/apps/details?id=com.bootiq2.gtsisic&hl=cs⟩.

[6] Jones, Michael; Bradley, John; Sakimura, Nat. *JSON Web Token (JWT)* [online]. 2015 [cit. 2026-4-1]. Dostupné z: ⟨https://datatracker.ietf.org/doc/html/rfc7519⟩.

[7] Hardt, Dick. *The OAuth 2.0 Authorization Framework* [online]. 2012 [cit. 2026-4-1]. Dostupné z: ⟨https://datatracker.ietf.org/doc/html/rfc6749⟩.

[8] *Android Developers. : Android Keystore system* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://developer.android.com/privacy-and-security/keystore⟩.

[9] Provos, Niels; Mazières, David. A Future-Adaptable Password Scheme. In. *Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference* [online]. 1999 [cit. 2026-4-1]. Dostupné z: ⟨https://www.usenix.org/legacy/events/usenix99/provos/provos.pdf⟩.

[10] *OWASP. : JSON Web Token Cheat Sheet for Java* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://cheatsheetseries.owasp.org/cheatsheets/JSON_Web_Token_for_Java_Cheat_Sheet.html⟩.

[11] Fielding, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures* [online]. 2000 [cit. 2026-4-1]. Dostupné z: ⟨https://roy.gbiv.com/pubs/dissertation/top.htm⟩.

[12] *Microsoft Learn. : What is .NET MAUI?* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui⟩.

[13] *Microsoft .NET. : ASP.NET – Open-source web framework for .NET* [online]. [cit. 2026-7-9]. Dostupné z: ⟨https://dotnet.microsoft.com/en-us/apps/aspnet⟩.

[14] *PostgreSQL. : PostgreSQL Documentation* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://www.postgresql.org/docs/⟩.

[15] *Npgsql. : Npgsql – .NET Access to PostgreSQL* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://www.npgsql.org/⟩.

[16] *Microsoft Playwright. : Playwright Documentation* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://playwright.dev/dotnet/docs/intro⟩.

[17] *Leaflet. : An open-source JavaScript library for mobile-friendly interactive maps* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://leafletjs.com/⟩.

[18] *OpenStreetMap. : The Free Wiki World Map* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://www.openstreetmap.org/about⟩.

[19] Európsky parlament; Rada Európskej únie. *Nariadenie Európskeho parlamentu a Rady (EÚ) 2016/679 z 27. apríla 2016 o ochrane fyzických osôb pri spracúvaní osobných údajov a o voľnom pohybe takýchto údajov, ktorým sa zrušuje smernica 95/46/ES (všeobecné nariadenie o ochrane údajov)* [online]. 2016 [cit. 2026-4-1]. Úradný vestník Európskej únie, L 119, s. 1–88. Dostupné z: ⟨https://eur-lex.europa.eu/legal-content/SK/TXT/?uri=CELEX:32016R0679⟩.

[20] *Microsoft Learn. : Entity Framework Core documentation* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://learn.microsoft.com/en-us/ef/core/⟩.

[21] *Microsoft Learn. : Overview of ASP.NET Core authentication* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://learn.microsoft.com/en-us/aspnet/core/security/authentication/⟩.

[22] *Microsoft Learn. : Secure storage* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://learn.microsoft.com/en-us/dotnet/maui/platform-integration/storage/secure-storage⟩.

[23] *Android Developers. : Location and context overview* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://developer.android.com/develop/sensors-and-location/location⟩.

[24] Veness, Chris. *Movable Type Scripts: Calculate distance, bearing and more between Latitude/Longitude points* [online]. [cit. 2026-7-19]. Dostupné z: ⟨https://www.movable-type.co.uk/scripts/latlong.html⟩.

[25] *Swagger. : OpenAPI Specification* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://swagger.io/specification/⟩.

[26] *Docker. : Docker Documentation* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://docs.docker.com/⟩.

[27] *pgAdmin. : pgAdmin – PostgreSQL Tools* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://www.pgadmin.org/⟩.

[28] *BlueStacks. : BlueStacks 5 – Android Emulator for PC* [online]. [cit. 2026-4-1]. Dostupné z: ⟨https://www.bluestacks.com/⟩.

[29] *Fake GPS location. : Google Play* [online]. [cit. 2026-7-11]. Dostupné z: ⟨https://play.google.com/store/apps/details?id=com.blogspot.newapphorizons.fakegps&hl=en-US⟩.

[30] *Ubuntu Server. : Ubuntu Server documentation* [online]. [cit. 2026-7-11]. Dostupné z: ⟨https://ubuntu.com/server/docs⟩.

[31] *Caddy. : The Ultimate Server with Automatic HTTPS* [online]. [cit. 2026-7-11]. Dostupné z: ⟨https://caddyserver.com/docs/⟩.

[32] *Let's Encrypt. : A nonprofit Certificate Authority* [online]. [cit. 2026-7-11]. Dostupné z: ⟨https://letsencrypt.org/⟩.

[33] *Duck DNS. : Free dynamic DNS* [online]. [cit. 2026-7-11]. Dostupné z: ⟨https://www.duckdns.org/⟩.