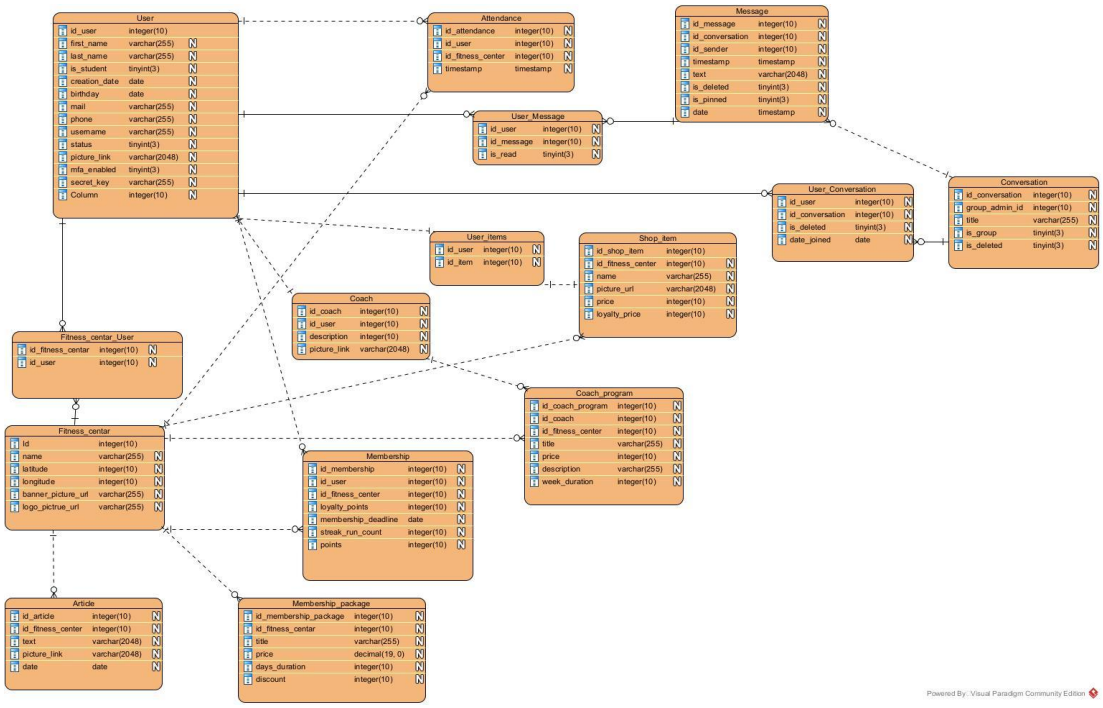
Fitness Centar

.NET Backend + Android Kotlin application

Lovro Lešić

Travanj – Srpanj 2025.

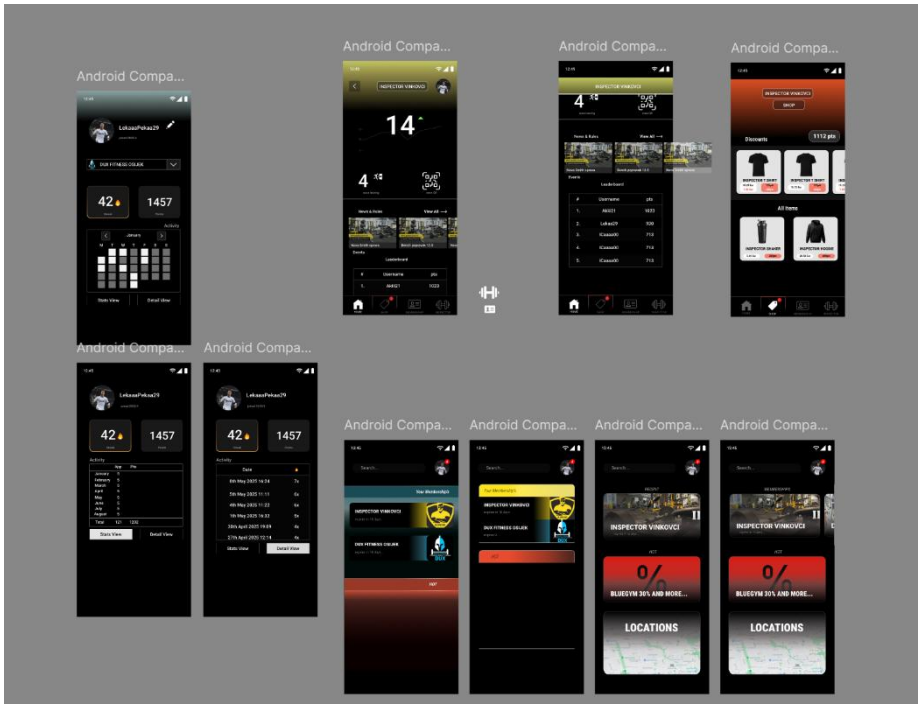


Funkcionalni zahtjevi fitness centar aplikacije:

- **Autentifikacija i registracija korisnika**: Korisnik može stvoriti novi korisnički račun, prijaviti se, a sustav generira i pohranjuje autentifikacijski token s korisničkom e-mail adresom kao referencom.
- **Obnavljanje trajanja autentifikacijskog tokena**: Token se može obnoviti za nastavak autentifikacije.
- **Dodavanje fitness centra**: Omogućava se unos novog fitness centra u sustav.
- **Dodavanje fitness centar pretplatnik paketa**: Korisnik može dodavati, brisati i ažurirati pretplatničke pakete, te im se mogu dodavati ili brisati popusti.
- **Ažuriranje članarine korisnika**: Korisnik može ažurirati članarinu, pratiti datum isteka, skupljati bodove i pratiti broj kontinuiranih dolazaka, a paketi omogućuju produžetak članarine.

- **Dodavanje trenera**: Trener može dodati svoj profil s informacijama o stručnosti, znanju i trening paketima.
- **Dodavanje trening program paketa**: Treneri mogu dodavati i ažurirati pakete treninga s ciljevima, cijenama i trajanjem.
- **Dodavanje članka**: Fitness centar može dodavati članke s tekstom, slikama i datumom za obavijesti korisnicima.
- **Dodavanje dolaznosti**: U svakom dolasku u centar korisnik automatski bilježi dolazak u sustav.
- **Prikaz korisnikove dolaznosti**: Korisnik može vidjeti svoje dolaznosti na mjesečnom kalendaru i vizualiziranoj listi dolazaka.
- **Prikaz fitness centar dolaznosti**: Fitness centar prikazuje broj dolazaka svojih korisnika kroz grafički prikaz, pokazujući popularnost centra.
- **Tablica dolaznih bodova**: Svaki fitness centar prati korisničke bodove temeljem dolazaka, s prikazom najboljih korisnika na glavnom ekranu.
- **Trgovina fitness centra**: Svaki centar ima prodajnu ponudu s artiklima za prodaju.
- **Dodavanje artikla u trgovinu**: Centar može dodati artikle u svoju trgovinu s cijenama u eurima i bodovima.
- **Prikaz korisnikovih kupljenih artikala**: Korisnik može vidjeti povijest kupljenih artikala iz svih centara.
- **Chat između korisnika**: Omogućava slanje poruka između korisnika ili u grupama korisnika.
- **Dodavanje poruke**: Korisnici mogu slati poruke u grupe ili direktni chat.
- **Označavanje poruke kao pročitane**: Poruke u sustavu označavaju se kao pročitane nakon što su otvorene.
- **Mapa fitness centra**: Prikazuje sve lokacije centara u sustavu i omogućuje navigaciju do odabranog centra.
- **Prikaz bližih centara**: Na glavnom ekranu prikaz 3 najbliža fitness centra s podacima o udaljenosti i tipkom za prikaz na mapi.
- **Prikaz fitness centara s popustom na pakete**: Glavni ekran prikazuje teretane koje nude popuste na svoje pakete.
- **Prikaz trenutne popunjenosti centra**: Na glavnom ekranu centra prikazuje se broj korisnika koji su ušli u centar u zadnja 1.5h.
- **Prikaz broja korisnika koji završavaju s treningom**: Prikazuje broj korisnika koji završavaju trening (dolaznost starosti između 1h I 1.5h).

- **Ekran za pretplatničke pakete**: Na ekranu centra prikazuju se svi dostupni pretplatnički paketi s mogućnošću kupnje.
- **Profil ekran**: Korisnički profil prikazuje informacije o korisniku te statistiku vezanu za dolaske u fitness centar. Mogućnost promjene slike profilne.



Development komentari I bilješke

- Arhitektura
- Async
- ClaimTypes
- IConfiguration
- DTO
- Bearer
- JWT Token
- Retrofit
- RetrofitBuilder
- Dizajn ekrana
- Repozitoriji
- ViewModel
- UIState
- Cloudflare
- Razmjena poruka korisnika
- Baza
- Redoslijed projektiranja aplikacije

Arhitektura

Backend je napravljen na načelu Controller-Repository-Service pattern-a, što omogućava čistu separaciju logike i lakše održavanje koda.

Android aplikacija koristi modernu strukturu: Retrofit za mrežnu komunikaciju, Repository pattern, ViewModel, UIState management i kompozitne ekrane.

Async pristup

Backend koristi asinkrone funkcije kako bi baza mogla raditi asinkrone pozive. Ovo je posebno važno jer u produkciji latencija do servera koji je udaljen u stranoj zemlji može značajno utjecati na performanse i koštati nas.

ClaimTypes

U kontroleru preko ClaimTypes iz JWT tokena možemo izvući odabrane spremljene podatke. Spremam email kao indikator koji korisnik šalje zahtjev, te također token koji govori je li korisnik autentificiran.

IConfiguration

Koristimo built-in interface za čitanje appsettings.json datoteke u kojoj čuvamo admin mail kojem jedinom dopuštamo neke operacije. Za ovu MVP verziju to nam je supstitucija za role-based pristup koji bi se inače dodavao u ASP.NET Identity roles i stvarale više različitih korisnika po pravima.

DTO (Data Transfer Objects)

Objekt koji se može serijalizirati jer je napravljen od jednostavnih podataka ili drugih DTO objekata istog pravila.

U glavnom pravilu mora nam podržavati podatke za dodavanje modela kojeg predstavlja i također podatke koje ćemo koristiti za prezentaciju (koje većinom nisu različite). Postaje jako koristan kad nam trebaju nepredviđeni podaci za prezentaciju ili kad spajamo više njih jer nam u logici ekrana olakšava logiku GET zahtjeva.

Bearer Token

Bearer token je način autentifikacije gdje klijent šalje token u HTTP header-u (Authorization: Bearer <token>). Server tada može verificirati identitet korisnika na temelju tog tokena.

JWT Token

Token koji čuva korisnik i baza te time prepoznaju korisnikovu sigurnost pristupa. Tokeni po odabiru imaju svoj istekni rok nakon kojeg baza više ne prihvaća korisnika kao prijavljenog. Korisnik koji je već prijavljen treba imati automatsko produženje tokena.

Retrofit

Retrofit je HTTP klijent za Android koji omogućava elegantno pozivanje REST API-ja. Bitno je da Retrofit ima potpise endpointova baze te da pazimo da su naši modeli na Android aplikaciji i oni na backend-u podržani za komunikaciju u svakom endpoint-u - važno je paziti na vrstu podataka, imenovanje i nullability.

RetrofitBuilder

Gradi Retrofit instancu i dodajemo interceptor koji svaki poziv prema bazi potpisuje s Bearer tokenom. To osigurava razmjenu tokena između backend-a i aplikacije te time bazi rješava pitanje sigurnosti zahtjeva i dobiva pristup endpoint-ovima.

Dizajn ekrana

Navbar, mogući ekrani i podaci su preloadani tako što ne koristimo NavController što daje dojam bržeg učitavanja i prijelaza između ekrana koji bi bio prisutan dok se obavljaju pozivi i crtanje kompleksnih ekrana.

Nisam koristio strings.xml za spremanje boja i stringova jer sam u ovoj MVP verziji prioritizirao ispunjavanje Figma zahtjeva i vremenske rokove nego toliko clean code načela i "single point of truth" princip.

Pozadine dva glavna ekrana napravljene su preko animiranog linearnog gradijenta tako što je prva boja u gradijent animirana i razvija se kroz odabrane nijanse i snagu u ponavljajućem kruženju odabranog intervala.

Repository

Obavljaju komunikaciju s Retrofit endpoint-ovima. Za GET operacije većinom jednostavna implementacija, dok nekada treba koristiti više Android repozitorija za ispunjavanje jednog zahtjeva. Također korištenje Flow-a kako bi automatizirali ponavljanje nekih GET zahtjeva svakog odabranog intervala vremena.

U POST i PUT operacijama repozitoriji moraju ispuniti i pripremiti objekt koji se šalje. Za slanje poruka koristio sam Flow umjesto WebSocket-a zbog vremenskog roka projekta.

ViewModel

Koristimo suspend funkcije i launchModel korutine kako aplikacija može slati zahtjeve prema bazi preko drugih procesorskih niti te tako ne gubimo responsivnost aplikacije dok ispunjava zahtjeve. Zbog Kotlin development-a mislim da je olakšan dizajn korutina i manje opasnosti od curenja memorije i "callback hell" padanja aplikacije.

UiState

Podaci ekrana koji sadrže različite liste objekata su sve spremljeni u ekranov zaseban UiState koji se ViewModelom obnavlja po pozivu ili automatizirano. Preko lifecycle remember na UiState podatku u ekranu omogućujemo slušanje promjena na cijelom UiState-u što svakom promjenom obnavlja ekran te tako na jednostavan način naš ekran responsivno komunicira s bazom.

Cloudflare Images

Za slike aplikacije koristio sam Cloudflare Images koji omogućava čuvanje slika u obliku URL linka. Na mobilnoj aplikaciji konkretno za profilnu sliku šaljemo na Cloudflare zahtjev zajedno sa slikom, API ključem i potpisom te dobivamo nazad informaciju je li uspješno spremljena i slikin URL link koji tek zatim šaljemo na backend da baza unese URL nove slike. Ostale slike aplikacije ubacivao sam manualno u Cloudflare, a u bazi zapisivao URL-ove.

Razmjena poruka korisnika

Zbog odluke da postoji mogućnost dopisivanja u grupama sama logika modela pa tako i backend-a postaje zahtjevnija jer više poruka nije samo komunikacija između sendera i receivera. U ovom slučaju poruke čuvaju samo sendera a povezane su na neki razgovor što nam logički predstavlja receivera točnije odredište poruke. Članovi se moraju također povezati na razgovor kako bi se ubrajali u korisnike koji vide poruke.

Dodatan problem je odluka da imamo mogućnost označavanja poruke kao pročitane koja na vrh svega još jednom stavlja apstrakciju - poruka ima zasebnu poveznicu na svakog korisnika razgovora kako bi za svakog korisnika individualno mogli uređivati podatak pročitanosti jer u suprotnom bi bilo koji korisnik svima mogao prebaciti tu zastavicu.

Baza

Baza je napravljena u SQLite preko DBeaver-a što nam je zatim omogućilo automatsko povlačenje modela u .NET bazu (Database First pristup). Novije modele sam dodavao direktno u DataContext i pravio .NET migracije.

Redoslijed projektiranja aplikacije

- **ERA MODEL, INICIJALNI FUNKCIONALNI ZAHTJEVI** – rani travanj
- **FIGMA DIZAJN** – travanj
- **SECURITY .NET** – travanj
- **SQLITE I .NET DATACONTEXT** – travanj
- **BACKEND IZVRŠENI KONTROLERI** – travanj-svibanj
- **ANDROID ENDPOINT APP** – svibanj-lipanj
- **ANDROID UX APP** – lipanj
- **INTEGRACIJA UX APP U ENDPOINT APP** – kraj lipnja

FitnessCenter **GET/api/FitnessCenter** **GET/api/FitnessCenter/{fitnessCenterId}** **GET/api/FitnessCenter/ClosestFitnessCentars** **GET/api/FitnessCenter/coaches/{fitnessCentarId}** **GET/api/FitnessCenter/PromoFitnessCentars** **POST/api/FitnessCenter/AddFitnessCenter** **PUT/api/FitnessCenter/UpdateFitnessCentar/{fitnessCentarId}** **DELETE/api/FitnessCenter/DeleteFitnessCentar/{fitnessCentarId}**
Membership **GET/api/Membership/user** **GET/api/Membership/user/{fitnessCenterId}** **GET/api/Membership/FitnessCenter/{fitnessCenterId}** **GET/api/Membership/FitnessCenter/Leaderboard/{fitnessCenterId}** **POST/api/Membership/AddMembership** **POST/api/Membership/AddMembershipPackage** **POST/api/Membership/UpdateMembership** **GET/api/Membership/MembershipPackage/{membershipPackageId}** **GET/api/Membership/MembershipPackages/{fitnessCentarId}** **PUT/api/Membership/UpdateMembership/{membershipId}** **DELETE/api/Membership/DeleteMembership/{membershipId}**

Coach
GET/{coachId}
GET/coachProgram/{coachProgramId}
GET/coach/programs/{coachId}
POST/api/Coach/AddCoach
POST/api/Coach/AddCoachProgram
PUT/api/Coach/UpdateCoach/{coachId}
DELETE/api/Coach/DeleteCoach/{coachId}

Attendance
GET/api/Attendance/users
GET/api/Attendance/users/{fitnessCenterId}
GET/api/Attendance/fitnesscenters/{fitnessCenterId}
GET/api/Attendance/fitnesscenters/recent/{fitnessCenterId}
GET/api/Attendance/fitnesscenters/leaving/{fitnessCenterId}
POST/api/Attendance/AddAttendance
PUT/api/Attendance/UpdateAttendance/{attendanceId}
DELETE/api/Attendance/DeleteAttendance/{attendanceId}

Article
GET/api/Article/articles/{fitnessCenterId}
POST/api/Article/AddArticle
PUT/api/Article/UpdateArticle/{articleId}
DELETE/api/Article/DeleteArticle/{articleId}

Account
GET/api/Account/{userId}
POST/api/Account/Register
POST/api/Account/login
POST/api/Account/UpdateUser

Message
GET/api/Conversation/{conversationId}
GET/api/Conversation/chats
GET/api/Conversation/{conversationId}/participants
GET/api/Conversation/{conversationId}/search
POST/api/Conversation/message/send/{recipientId}
POST/api/Conversation/message/markAsRead
DELETE/api/Conversation/message/remove/{messageId}
DELETE/api/Conversation/remove/{conversationId}
PUT/api/Conversation/message/update/{messageId}
GET/api/Conversation/{conversationId}/message/unreadCount
GET/api/Conversation/unreadCount
POST/api/Conversation/createGroup
POST/api/Conversation/{conversationId}/addParticipant
POST/api/Conversation/{conversationId}/removeParticipant
POST/api/Conversation/{conversationId}/leaveGroup

POST/api/Conversation/{messageId}/pinMessage
Shop **GET/api/Shop/items/{fitnessCenterId}** **GET/api/Shop/items/users** **GET/api/Shop/item/{shopItemId}** **POST/api/Shop/BuyShopItem/{shopItemId}** **POST/api/Shop/AddShopItem** **PUT/api/Shop/UpdateShopItem/{shopItemId}** **DELETE/api/Shop/DeleteShopItem/{shopItemId}**