Izel Repuh

IB210166

4Paw – Recommender dokumentacija

1. Opis sistema preporuke

Sistem preporuke temelji se na analizi podataka o terminima, prihodima i aktivnostima korisnika.
Umjesto primjene kompleksnih algoritama mašinskog učenja, koristi se jednostavna **rule-based logika** (pravila zasnovana na podacima):

- Za svaku uslugu izračunava se **ukupan broj obavljenih termina** i **ostvareni prihod**.
- Na osnovu tih vrijednosti sistem **automatski prepoznaje najtraženije usluge** i prikazuje ih kao „**Top usluge**“ u vizualizacijama (grafovima).
- Dodatno, prati se i **aktivnost klijenata** – oni koji imaju najviše zakazanih termina ili su ostvarili najveću potrošnju označavaju se kao „**Top klijenti**“.

Primjena

- **Veterinar:** dobija pregled vlastitih *najtraženijih usluga* i područja s najvećim učinkom, što mu pomaže da prepozna koje vrste tretmana ili pregleda donose najviše angažmana i prihoda.
- **Administrator:** ima uvid u *globalne rezultate* svih veterinara — najprofitabilnije usluge i ukupne prihode — što omogućava donošenje informisanih poslovnih odluka, poput fokusiranja na isplativije ili najčešće tražene usluge.

Prikaz

- **Desktop dashboard:** uključuje grafikon *„Prihod po uslugama”*, tabelu *„Usluge po prihodu”* i listu *„Najbolji klijenti”*, koji vizualno prikazuju ključne pokazatelje uspješnosti.
- **Veterinarske statistike:** nude personalizirani pregled poput *„Moje top usluge”* i *„Moji dnevni termini”*, čime se olakšava praćenje individualnog učinka i planiranje rada.

2. Opis implementacije

Tehnologije:

- **Backend:** ASP.NET Core Web API, Entity Framework Core, SQL Server
- **UI:** Flutter (desktop i mobilna verzija), Dio HTTP klijent, Provider za state management
- **Shared sloj:** *veterinarska_shared* biblioteka (zajednički servisi i modeli)

Ključni API endpointi:

- GET /financial/veterinarian/my-stats
- GET /financial/veterinarian/daily-appointments
- GET /financial/veterinarian/top-services
- GET /financial/admin/financial-summary
- GET /financial/admin/revenue-by-services

Algoritam:

- Iz baze se dohvaćaju zapisi iz tabele **Appointments** sa statusom *Completed*.
- Podaci se projektuju na naziv usluge (reason/service), te se računa **ukupan prihod** i **broj obavljenih termina**.
- Rezultati se **grupišu po usluzi**, zatim **sortiraju prema ukupnom prihodu**, i izračunava se **procentualno učešće svake usluge** u ukupnom prihodu, koje se vizualno prikazuje u UI dijelu sistema.

Sigurnost

Autentikacija i autorizacija:

- Sistem koristi **JWT (JSON Web Token)** mehanizam za autentikaciju korisnika.
- Prilikom svakog zahtjeva token se validira na backendu, čime se osigurava da samo prijavljeni korisnici imaju pristup zaštićenim resursima.

- Dodatno, korištenje **[RoleRequired]** atributa omogućava **ograničen pristup po korisničkim ulogama** – npr. rutu *„top-services"* mogu pozivati isključivo korisnici s ulogom **Veterinarian**, dok su administrativne rute dostupne samo korisnicima s ulogom **Admin**.

```
public class RoleRequiredAttribute : Attribute, IAuthorizationFilter
{
    private readonly UserRole[] _requiredRoles;

    public RoleRequiredAttribute(params UserRole[] requiredRoles)
    {
        _requiredRoles = requiredRoles;
    }

    public void OnAuthorization(AuthorizationFilterContext context)
    {
        // Check if user is authenticated
        if (!context.HttpContext.User.Identity?.IsAuthenticated ?? true)
        {
            context.Result = new UnauthorizedResult();
            return;
        }

        // Get user role from claims
        var roleClaim = context.HttpContext.User.FindFirst(ClaimTypes.Role)?.Value;

        if (string.IsNullOrEmpty(roleClaim) || !Enum.TryParse<UserRole>(roleClaim, out var userRole))
        {
            context.Result = new ForbidResult();
            return;
        }

        // Check if user has required role
        if (!_requiredRoles.Contains(userRole))
        {
            context.Result = new ForbidResult();
            return;
        }
    }
}
```

Komunikacija

Veza između UI i API sloja:

- Flutter korisnički interfejs komunicira s backendom putem servisa definiranog u **veterinarska_shared/lib/services/financial_service.dart**.
- Svaki HTTP zahtjev koristi **Dio klijent**, pri čemu se **JWT token automatski ubacuje u zaglavlje Authorization** u formatu:
- Authorization: Bearer <token>
- Na ovaj način osigurana je **sigurna i autentifikovana komunikacija** između korisničke aplikacije i API-ja, dok se token validira na serverskoj strani prije izvršavanja svake zaštićene operacije.

```
Future<VeterinarianStats> getVeterinarianStats() async {
  try {
    final token = await serviceLocator.authService.getAccessToken();

    if (token == null || token.isEmpty) {
      throw Exception('Niste prijavljeni. Molimo prijavite se ponovo.');
    }

    // Učitaj osnovne statistike
    final statsResponse = await serviceLocator.apiClient.dio.get(
      '/financial/veterinarian/my-stats',
      options: Options(
        headers: {
          'Authorization': 'Bearer $token',
          'Content-Type': 'application/json',
        },
      ),
    );
```

```
// Učitaj top usluge
final servicesResponse = await serviceLocator.apiClient.dio.get(
  '/financial/veterinarian/top-services',
  options: Options(
    headers: {
      'Authorization': 'Bearer $token',
      'Content-Type': 'application/json',
    },
  ),
);
```

3. Putanje i screenshot glavne logike (backend)

Glavna logika „Top usluge” (veterinar) – kontroler finansija:

```
[HttpGet("veterinarian/top-services")]
[RoleRequired(UserRole.Veterinarian)]
public async Task<ActionResult<List<RevenueByService>>> GetVeterinarianTopServices()
{
    try
    {
        var userIdClaim = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;
        if (string.IsNullOrEmpty(userIdClaim) || !int.TryParse(userIdClaim, out int veterinarianId))
        {
            return BadRequest("Nevaljan korisnik ID");
        }
```

```
[HttpGet("veterinarian/top-services")]
[RoleRequired(UserRole.Veterinarian)]
public async Task<ActionResult<List<RevenueByService>>> GetVeterinarianTopServices()
{
    try
    {
        var userIdClaim = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;
        if (string.IsNullOrEmpty(userIdClaim) || !int.TryParse(userIdClaim, out int veterinarianId))
        {
            return BadRequest("Nevaljan korisnik ID");
        }

        // Uzmi SVE termine veterinara (ne samo ovaj mjesec)
        _logger.LogInformation($"🔍 Getting top services for veterinarian ID: {veterinarianId}");

        // Primarni pristup: uzmi potrebne podatke iz baze, pa grupiši u memoriji
        var rawItems = await _context.Appointments
            .Where(a => a.VeterinarianId == veterinarianId && a.Status == AppointmentStatus.Completed)
            .Select(a => new
            {
                ServiceName = a.Service != null ? a.Service.Name : null,
                a.Reason,
                a.Type,
                Amount = (a.ActualCost ?? a.EstimatedCost) ?? 0
            })
            .ToListAsync();
```

4. Putanja i screenshot orkestracije na UI-u (shared servis)

FinancialService.getVeterinarianStats – dohvaća my-stats, daily-appointments, top-services i spaja u jedan objekat:

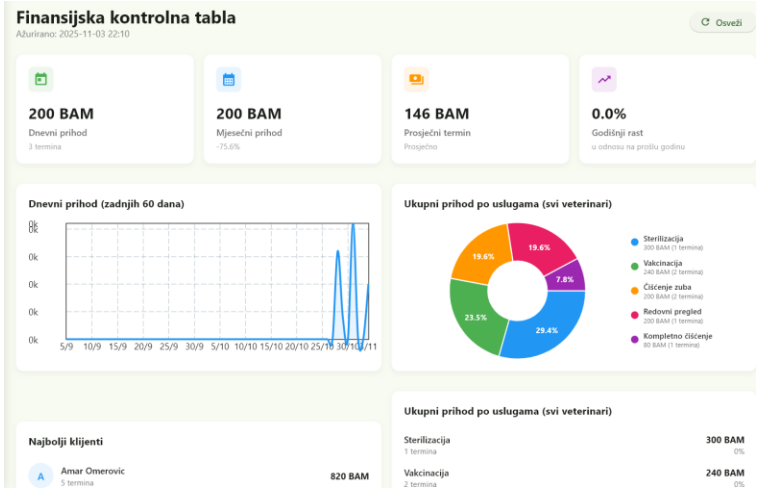
```
// Primarni pristup: uzmi potrebne podatke iz baze, pa grupiši u memoriji
var rawItems = await _context.Appointments
    .Where(a => a.VeterinarianId == veterinarianId && a.Status == AppointmentStatus.Completed)
    .Select(a => new
    {
        ServiceName = a.Service != null ? a.Service.Name : null,
        a.Reason,
        a.Type,
        Amount = (a.ActualCost ?? a.EstimatedCost) ?? 0
    })
    .ToListAsync();

string MapType(AppointmentType t)
{
    return t == AppointmentType.Checkup ? "Pregled" :
           t == AppointmentType.Vaccination ? "Vakcinacija" :
           t == AppointmentType.Surgery ? "Operacija" :
           t == AppointmentType.Emergency ? "Hitno" :
           t == AppointmentType.Grooming ? "Šišanje/Njega" :
           t == AppointmentType.Dental ? "Stomatologija" :
           t == AppointmentType.Consultation ? "Konsultacija" :
           t == AppointmentType.FollowUp ? "Kontrola" : t.ToString();
}

var topServices = rawItems
    .Select(x => new
    {
        Name = !string.IsNullOrWhiteSpace(x.ServiceName)
            ? x.ServiceName!
            : (!string.IsNullOrWhiteSpace(x.Reason) ? x.Reason! : MapType(x.Type)),
        x.Amount
    })
    .GroupBy(x => x.Name)
    .Select(g => new RevenueByService
    {
        ServiceName = g.Key,
        Revenue = g.Sum(i => i.Amount),
        Count = g.Count()
    })
    .OrderByDescending(r => r.Revenue)
    .Take(5)
    .ToList();
```

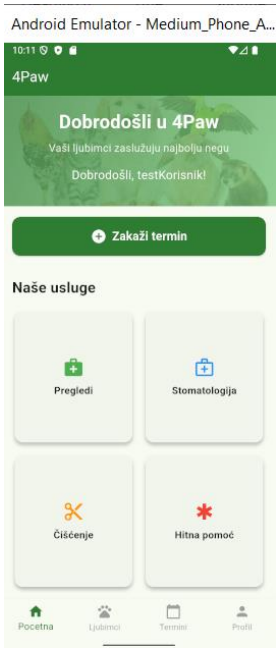
5. Putanja i screenshot iz pokrenute aplikacije (vizuelne preporuke)

Desktop finansijski dashboard – pie chart i tabela „Usluge po prihodu”:

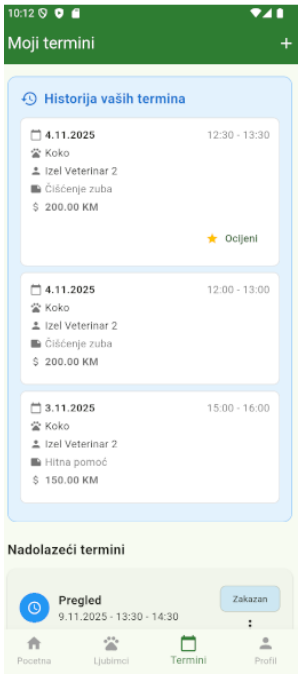


6. Početna (Mobile):

veterinarska_mobile/lib/screens/home/dashboard_screen.dart – marketinški prikaz i CTA „Zakaži termin”:



7. Lista termina + ocjenjivanje:
veterinarska_mobile/lib/screens/appointments/appointments_list_screen.dart.



8. Zaključak

Recommender u ovoj verziji je „explainable" i lagan: sve se može objasniti kroz izvještaje. Kasnije se može nadograditi na **CF/CBF modele** (npr. preporuke veterinara korisniku prema istoriji, vrstama ljubimaca i ocjenama).