TIPPR

Fas 1: MVP Implementation Guide

Fotbolls-VM 2026 (Q1-Q2 2026)

Version 1.0 | 2025-12-15

Innehållsförteckning

1. Översikt & Mål

1.1 Fas 1 Mål (MVP)

Målet med Fas 1 är att skapa en fullt fungerande tippningsapplikation för Fotbolls-VM 2026. Detta är din Minimum Viable Product (MVP) som ska vara redo att lanseras innan VM-start.

Kärnfunktioner:

- ■ Användare kan registrera sig och logga in
- ■ Skapa/gå med i privata ligor
- ■ Tippa alla VM-matcher
- ■ Tippa bonusfrågor (skyttekung, vinnare)
- ■ Live-resultat från API-FOOTBALL
- ■ Automatisk poängberäkning
- ■ Ligatabeller med realtidsuppdateringar
- ■ Grundläggande notifikationer

1.2 Tidsplan

Total tid: 16 veckor (4 månader)

Arbetstempo: Part-time (15-20 timmar/vecka vid sidan av studier)

Flexibilitet: Du kan justera tempot efter dina studier

Sprint	Veckor	Fokus	Leverans
1	1	Setup	Repo, projekt struktur, CI/CD
2	2	Database & Auth	Supabase, EF Core, Auth flow
3-4	3-4	Backend API	REST API, CQRS, Business logic
5-6	5-6	Frontend Core	React app, routing, state management
7-8	7-8	Predictions	Tippningssystem, deadline-hantering
9-10	9-10	Live Results	API-FOOTBALL integration, scoring
11-12	11-12	Realtime	Supabase Realtime, ligatabeller
13-14	13-14	Testing	E2E tests, bug fixes, polish
15-16	15-16	Deployment	Production deploy, monitoring

2. Sprint 1: Projektuppsättning (Vecka 1)

2.1 Mål

Sätt upp hela projektet med korrekt arkitektur, CI/CD och utvecklingsmiljö.

2.2 Uppgifter

■ Uppgift 1: Skapa GitHub Repository

1. Skapa nytt GitHub repo: **tippr**
2. Välj .gitignore för .NET och Node
3. Lägg till README.md med projektbeskrivning
4. Skapa LICENSE (MIT eller valfri)

■ Uppgift 2: Backend Projektstruktur (.NET)

Skapa Clean Architecture struktur:

```
mkdir Tippr.Backend
cd Tippr.Backend
dotnet new sln -n Tippr

# Domain Layer
dotnet new classlib -n Tippr.Domain
dotnet sln add Tippr.Domain

# Application Layer
dotnet new classlib -n Tippr.Application
dotnet sln add Tippr.Application

# Infrastructure Layer
dotnet new classlib -n Tippr.Infrastructure
dotnet sln add Tippr.Infrastructure

# API Layer
dotnet new webapi -n Tippr.Api
dotnet sln add Tippr.Api
```

Projektstruktur:

```
Tippr.Backend/
■■■ Tippr.Domain/ # Entities, Enums, Interfaces
■■■ Tippr.Application/ # CQRS, Commands, Queries, DTOs
■■■ Tippr.Infrastructure/ # EF Core, Supabase, External APIs
■■■ Tippr.Api/ # Controllers, Middleware
```

■ Uppgift 3: Frontend Projektstruktur (React + Vite)

```
npm create vite@latest tippr-frontend -- --template react-ts
cd tippr-frontend
npm install

# Installera dependencies
npm install @supabase/supabase-js
npm install @tanstack/react-query
npm install react-router-dom
npm install zustand

# UI & Styling
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
npx shadcn-ui@latest init
```

Mappstruktur:

```
tippr-frontend/
■■■ src/
■ ■■■ components/ # Återanvändbara komponenter
■ ■■■ features/ # Feature-baserade komponenter
■ ■■■ pages/ # Sidor/routes
■ ■■■ lib/ # Supabase client, API client
■ ■■■ hooks/ # Custom React hooks
■ ■■■ store/ # Zustand stores
■ ■■■ types/ # TypeScript types
■ ■■■ utils/ # Helper functions
```

■ Uppgift 4: Installera NuGet Packages (.NET)

```
cd Tippr.Infrastructure

dotnet add package Npgsql.EntityFrameworkCore.PostgreSQL

dotnet add package Microsoft.EntityFrameworkCore.Design

dotnet add package Supabase

cd ../Tippr.Application

dotnet add package MediatR

dotnet add package FluentValidation

dotnet add package AutoMapper

cd ../Tippr.Api

dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer

dotnet add package Serilog.AspNetCore
```

■ Uppgift 5: Sätt upp GitHub Actions CI/CD

Skapa .github/workflows/backend.yml:

```
name: Backend CI/CD on: push: branches: [ main, develop ] pull_request: branches: [ main ] jobs: build: runs-on: ubuntu-latest steps: - uses: actions/checkout@v3 - name: Setup .NET uses: actions/setup-dotnet@v3 with: dotnet-version: 8.0.x - name: Restore dependencies run: dotnet restore - name: Build run: dotnet build --no-restore - name: Test run: dotnet test --no-build --verbosity normal
```

Skapa .github/workflows/frontend.yml:

```
name: Frontend CI/CD on: push: branches: [ main, develop ] jobs: build: runs-on: ubuntu-latest steps: - uses: actions/checkout@v3 - name: Setup Node uses: actions/setup-node@v3 with: node-version: 20 - name: Install dependencies run: npm ci - name: Lint run: npm run lint - name: Build run: npm run build
```

■ Uppgift 6: Skapa .env templates

Backend (.env.example):

```
SUPABASE_URL=https://xxxxx.supabase.co SUPABASE_KEY=your-anon-key
SUPABASE_SERVICE_ROLE_KEY=your-service-role-key
DATABASE_CONNECTION=Host=db.xxxxx.supabase.co;Database=postgres;... JWT_SECRET=your-jwt-secret
API_FOOTBALL_KEY=your-api-key
```

Frontend (.env.example):

```
VITE_SUPABASE_URL=https://xxxxx.supabase.co VITE_SUPABASE_ANON_KEY=your-anon-key
VITE_API_URL=http://localhost:5000/api
```

2.3 Definition of Done

✓ GitHub repo skapad med korrekt .gitignore

✓ Backend har Clean Architecture struktur

✓ Frontend har React + TypeScript + Vite setup

✓ Alla NuGet packages och npm dependencies installerade

✓ GitHub Actions CI/CD pipelines fungerar

✓ .env templates skapade

✓ README.md uppdaterad med setup-instruktioner

3. Sprint 2: Database & Auth (Vecka 2)

3.1 Mål

Sätt upp Supabase, skapa alla database entities och implementera autentisering.

3.2 Uppgifter

■ Uppgift 1: Skapa Supabase Projekt

1. Gå till https://supabase.com och skapa konto
2. Skapa nytt projekt: **tippr-dev**
3. Välj region: **North Europe (Stockholm)**
4. Generera starkt database password
5. Spara credentials (Project URL, anon key, service_role key)

■ Uppgift 2: Konfigurera Supabase Auth

1. Gå till Authentication → Settings
2. Sätt **Site URL**: http://localhost:5173
3. Lägg till **Redirect URLs**: http://localhost:5173/**
4. Aktivera **Email Confirmation**
5. (Valfritt) Sätt upp Google OAuth för social login

■ Uppgift 3: Skapa Domain Entities

Se separat dokument: **Tippr_Database_Design.pdf**

Skapa alla 13 entities i Tippr.Domain/Entities/:

1. User.cs
2. Tournament.cs
3. Team.cs
4. Match.cs
5. League.cs
6. LeagueSettings.cs
7. LeagueMember.cs
8. Prediction.cs
9. BonusQuestion.cs
10. BonusPrediction.cs
11. LeagueStanding.cs

12. ChatMessage.cs
13. Notification.cs

■ Uppgift 4: Skapa DbContext & Configurations

Skapa ApplicationDbContext i Tippr.Infrastructure:

```
public class ApplicationDbContext : DbContext { public DbSet Users { get; set; } public DbSet Tournaments { get; set; } public DbSet Teams { get; set; } public DbSet Matches { get; set; } public DbSet Leagues { get; set; } // ... alla DbSets protected override void OnModelCreating(ModelBuilder modelBuilder) { modelBuilder.ApplyConfigurationsFromAssembly( Assembly.GetExecutingAssembly()); } }
```

Skapa EntityTypeConfigurations för varje entity (FluentAPI):

Tippr.Infrastructure/Persistence/Configurations/UserConfiguration.cs

Tippr.Infrastructure/Persistence/Configurations/MatchConfiguration.cs

... osv för alla entities

■ Uppgift 5: Skapa Initial Migration

```
cd Tippr.Infrastructure
```

```
dotnet ef migrations add InitialCreate --startup-project ../Tippr.Api --output-dir Persistence/Migrations
```

Granska genererad migration i Migrations-mappen

■ Uppgift 6: Applicera Migration till Supabase

Sätt connection string i appsettings.json

```
dotnet ef database update --startup-project ../Tippr.Api
```

Verifiera i Supabase Dashboard → Table Editor att alla tabeller finns

■ Uppgift 7: Implementera Supabase Auth i Backend

Se separat dokument: **Tippr_Supabase_Auth_Setup.pdf**

1. Konfigurera JWT Authentication i Program.cs
2. Skapa AuthService för user sync
3. Skapa AuthController med /auth/sync och /auth/me endpoints

■ Uppgift 8: Implementera Supabase Auth i Frontend

1. Skapa Supabase client (src/lib/supabase.ts)
2. Skapa AuthContext (src/contexts/AuthContext.tsx)
3. Skapa Login/Signup komponenter
4. Skapa Protected Route wrapper

■ Uppgift 9: Testa Authentication Flow

1. Registrera en testanvändare via frontend

2. Verifiera email (kolla Supabase dashboard för magic link)
3. Logga in och verifiera att JWT fungerar
4. Kolla att användare skapats i både auth.users och public.users

3.3 Definition of Done

✓ Supabase projekt skapad och konfigurerad
✓ Alla 13 domain entities skapade
✓ DbContext med FluentAPI configurations
✓ Initial migration skapad och applicerad
✓ Alla tabeller finns i Supabase
✓ Supabase Auth fungerar i både backend och frontend
✓ Kan registrera, logga in och få tillbaka user data

4. Sprint 3-4: Backend API Core (Vecka 3-4)

4.1 Mål

Bygga alla REST API endpoints med CQRS-pattern för kärnfunktionalitet: tournaments, teams, matches, leagues, och predictions.

4.2 Uppgifter

■ Uppgift 1: Konfigurera MediatR & CQRS

Program.cs:

```
builder.Services.AddMediatR(cfg =>
cfg.RegisterServicesFromAssembly(typeof(Application.AssemblyReference).Assembly));
```

■ Uppgift 2: Tournament Endpoints

Skapa Commands & Queries:

- **Query:** GetTournamentQuery (hämta specifik turnering)
- **Query:** GetAllTournamentsQuery (lista alla turneringar)
- **Command:** CreateTournamentCommand (admin only)

TournamentController endpoints:

- GET /api/tournaments - Lista alla
- GET /api/tournaments/{id} - Hämta specifik
- POST /api/tournaments - Skapa (admin)

■ Uppgift 3: Team Endpoints

Queries:

- GetTeamsByTournamentQuery
- GetTeamQuery

TeamController endpoints:

- GET /api/teams?tournamentId={id} - Alla lag för turnering
- GET /api/teams/{id} - Specifikt lag

■ Uppgift 4: Match Endpoints

Queries & Commands:

- GetMatchesByTournamentQuery
- GetMatchesByDateQuery
- GetMatchQuery

• UpdateMatchResultCommand (från API-FOOTBALL webhook)

MatchController endpoints:

• GET /api/matches?tournamentId={id} - Alla matcher

• GET /api/matches?date={date} - Matcher för specifikt datum

• GET /api/matches/{id} - Specifik match

• GET /api/matches/{id}/predictions - Alla tips för match (admin)

■ Uppgift 5: League Endpoints

Commands & Queries:

• CreateLeagueCommand

• JoinLeagueCommand (via invite code)

• GetUserLeaguesQuery

• GetLeagueQuery (med members)

• UpdateLeagueSettingsCommand

LeagueController endpoints:

• POST /api/leagues - Skapa liga

• GET /api/leagues - Användarens ligor

• GET /api/leagues/{id} - Specifik liga med members

• POST /api/leagues/{id}/join - Gå med via invite code

• PUT /api/leagues/{id}/settings - Uppdatera inställningar (owner)

• GET /api/leagues/{id}/standings - Ligatabellen

■ Uppgift 6: Prediction Endpoints

Commands & Queries:

• SubmitPredictionCommand

• UpdatePredictionCommand

• GetUserPredictionsQuery (för en liga)

• GetPredictionQuery (specifik match + liga)

PredictionController endpoints:

• POST /api/predictions - Skicka in tips

• PUT /api/predictions/{id} - Uppdatera tips (innan deadline)

• GET /api/predictions?leagueId={id} - Användarens alla tips

• GET /api/predictions/match/{matchId}?leagueId={id} - Tips för match

■ Uppgift 7: Validation med FluentValidation

Skapa validators för alla Commands:

• CreateLeagueCommandValidator

• SubmitPredictionCommandValidator

• etc...

Exempel:

```
public class CreateLeagueCommandValidator : AbstractValidator { public CreateLeagueCommandValidator() { RuleFor(x => x.Name) .NotEmpty().WithMessage('League name is required') .MaximumLength(100); RuleFor(x => x.TournamentId) .NotEmpty(); } }
```

■ Uppgift 8: DTOs & AutoMapper

Skapa DTOs för alla responses i Application/DTOs/:

• TournamentDto, TeamDto, MatchDto

• LeagueDto, LeagueMemberDto

• PredictionDto, LeagueStandingDto

Konfigurera AutoMapper profiles

■ Uppgift 9: Error Handling Middleware

Skapa global exception handling middleware:

```
public class ErrorHandlingMiddleware { public async Task InvokeAsync(HttpContext context, RequestDelegate next) { try { await next(context); } catch (Exception ex) { await HandleExceptionAsync(context, ex); } } }
```

■ Uppgift 10: Testa alla endpoints

Använd Postman eller Thunder Client (VS Code extension)

1. Importera collection med alla endpoints
2. Testa create/read operations
3. Verifiera validering fungerar
4. Testa [Authorize] attribut (kräver JWT)

4.3 Definition of Done

✓ MediatR och CQRS konfigurerat

✓ Alla Controllers skapade med endpoints

✓ Commands & Queries implementerade

✓ FluentValidation på alla Commands

✓ DTOs och AutoMapper konfigurerat

✓ Error handling middleware

✓ Alla endpoints testade och fungerar

✓ API dokumentation (Swagger)

5. Sprint 5-6: Frontend Foundation (Vecka 5-6)

5.1 Mål

Bygg frontend med routing, state management, UI components och integration med backend API.

5.2 Uppgifter

■ Uppgift 1: Sätt upp Routing

Installera React Router:

```
npm install react-router-dom
```

Skapa routes struktur:

/ - Landing page

/login - Login sida

/signup - Registrering

/dashboard - Huvudsida (protected)

/leagues - Lista över ligor

/leagues/{id} - Specifik liga

/leagues/{id}/predict - Tippningssida

/profile - Användarprofil

■ Uppgift 2: API Client Setup

Skapa src/lib/api.ts med axios eller fetch wrapper:

```
import { supabase } from './supabase' export async function apiClient(endpoint: string, options = {}) { const
{ data: { session } } = await supabase.auth.getSession() const headers = { 'Content-Type': 'application/json',
...options.headers, } if (session?.access_token) { headers['Authorization'] = `Bearer
${session.access_token}` } const response = await fetch( `${import.meta.env.VITE_API_URL}${endpoint}`, {
...options, headers } ) return response.json() }
```

■ Uppgift 3: React Query Setup

Konfigurera React Query för data fetching:

```
import { QueryClient, QueryClientProvider } from '@tanstack/react-query' const queryClient = new QueryClient({
defaultOptions: { queries: { refetchOnWindowFocus: false, retry: 1, staleTime: 5 * 60 * 1000, // 5 minutes },
}, }) // Wrap app in QueryClientProvider
```

■ Uppgift 4: Custom Hooks för API Calls

Skapa hooks i src/hooks/:

- useAuth() - Already exists from AuthContext
- useTournaments() - Hämta turneringar
- useLeagues() - Användarens ligor

- useLeague(id) - Specifik liga
- useMatches(tournamentId) - Matcher
- usePredictions(leagueId) - Användarens tips

Exempel:

```
export function useLeagues() { return useQuery({ queryKey: ['leagues'], queryFn: () => apiClient('/leagues') }) }
```

■ Uppgift 5: UI Components Library

Installera och konfigurera shadcn/ui:

```
npx shadcn-ui@latest add button
npx shadcn-ui@latest add card
npx shadcn-ui@latest add input
npx shadcn-ui@latest add dialog
npx shadcn-ui@latest add table
npx shadcn-ui@latest add badge
```

■ Uppgift 6: Skapa Huvudkomponenter

Navbar Component:

- Logo, navigation links
- User dropdown med logout
- Notifikations-ikon (visar antal olästa)

Dashboard Component:

- Lista användarens ligor
- Kommande matcher
- Quick stats (totala poäng, ranking)

LeagueCard Component:

- Ligans namn, bild, medlemsantal
- Användarens placering
- Link till ligasida

■ Uppgift 7: Skapa League Pages

LeaguesListPage:

- Lista alla ligor
- Knapp för 'Create New League'
- Sökfunktion/filter

CreateLeagueModal:

- Formulär: Name, Description, Tournament
- Settings: Deadline, Points, Prediction Mode

• Submit → POST /api/leagues

LeagueDetailPage:

• Ligainformation

• Tabs: Standings, Matches, Members, Settings

• Invite link/code

■ Uppgift 8: State Management med Zustand

Skapa stores för global state:

• useAuthStore - User info, login status

• useNotificationStore - Unread notifications count

• useLeagueStore - Selected league context

■ Uppgift 9: Loading & Error States

Skapa reusable komponenter:

• LoadingSpinner

• ErrorMessage

• EmptyState (när inga ligor/matches finns)

■ Uppgift 10: Responsive Design

Använd Tailwind breakpoints:

• Mobile-first design

• Testa på sm, md, lg breakpoints

• Drawer/sheet för mobile navigation

5.3 Definition of Done

✓ React Router konfigurerat med alla routes

✓ API client med authentication headers

✓ React Query setup för data fetching

✓ Custom hooks för alla API endpoints

✓ shadcn/ui komponenter installerade

✓ Navbar och Dashboard komponenter

✓ League pages (list, detail, create)

✓ Zustand stores för global state

✓ Loading/error states hanterade

✓ Responsive design fungerar på mobile

6-9. Återstående Sprints - Översikt

De återstående sprinten (7-8, 9-10, 11-12, 13-14, 15-16) innehåller:

Sprint	Fokus	Huvuduppgifter
7-8	Predictions	• Prediction form med deadline-check • Submit/update predictions • Visa användarens tips • Bulk prediction (tippa alla matcher)
9-10	Live Results	• API-FOOTBALL integration • Background job för resultat-synk • Poängberäknings-algoritm • Update LeagueStandings
11-12	Realtime	• Supabase Realtime för ligatabeller • Live match updates • Notifications system • WebSocket för chat (SignalR)
13-14	Testing	• Unit tests (xUnit) • Integration tests • E2E tests (Playwright) • Bug fixes och polish
15-16	Deployment	• Deploy backend (Railway/Render) • Deploy frontend (Vercel) • Setup monitoring (Sentry) • Documentation & launch

Observera: Detaljerade guider för Sprint 7-16 kommer i separata dokument. Fokusera på att slutföra Sprint 1-6 först innan du går vidare.

10. Resurser & Verktyg

Kategori	Resurser
Documentation	• .NET Docs: docs.microsoft.com • React Docs: react.dev • Supabase Docs: supabase.com/docs • API-FOOTBALL: api-football.com/documentation
Learning	• Clean Architecture: github.com/jasontaylordev/CleanArchitecture • CQRS with MediatR: YouTube tutorials • React Query: tanstack.com/query/latest/docs
Tools	• VS Code + C# Dev Kit • Postman/Thunder Client • Supabase Studio (web) • GitHub Desktop
Communities	• r/dotnet på Reddit • r/reactjs på Reddit • Supabase Discord • Stack Overflow

Sammanfattning & Nästa Steg

Detta dokument ger dig en komplett roadmap för att bygga Tippr MVP (Fas 1). Fokusera på att slutföra varje sprint metodiskt innan du går vidare till nästa.

Tips för framgång:

- **Börja smått:** Slutför Sprint 1-2 innan du tänker på resten
- **Testa ofta:** Verifiera att varje feature fungerar innan du går vidare
- **Commit ofta:** Pusha kod till GitHub dagligen
- **Dokumentera:** Skriv kommentarer och README för framtida dig
- **Var flexibel:** Justera tidsplanen efter dina studier
- **Fråga om hjälp:** Använd communities när du kör fast

Nästa dokument att läsa:

1. **Tippr_Database_Design.pdf** - För Sprint 2
2. **Tippr_Supabase_Auth_Setup.pdf** - För Sprint 2
3. **Tippr_Fas1_Sprint7-16.pdf** - Efter Sprint 6 (kommer snart)

Lycka till med utvecklingen! ■

Dokument genererat: 2025-12-15 10:14
Version 1.0 | Tippr Fas 1 MVP Guide