

📘 SPOR YÖNETİM UYGULAMASI - KAPSAMLI DOKÜMANTASYON

React Native ile geliştirilmiş, Firebase tabanlı spor ligi ve maç yönetim platformu

 React Native 0.73+  TypeScript 5.0+  Firebase 10.0+ License MIT

📘 İçindekiler

- 📕 SPOR YÖNETİM UYGULAMASI - KAPSAMLI DOKÜMANTASYON
 - 📕 İçindekiler
 - 1. Proje Hakkında
 - 1.1 Genel Bakış
 - 1.2 Temel Özellikler
 - 2. Teknoloji Yığını
 - 2.1 Frontend
 - 2.2 Backend
 - 2.3 Geliştirme Araçları
 - 3. Mimari Yapı
 - 3.1 Proje Klasör Yapısı
 - 3.2 Mimari Katmanlar
 - 4. Veritabanı Şeması
 - 4.1 Collection'lar (24 Adet)
 - ⚙ CORE Collections (7)
 - 💬 INTERACTION Collections (5)
 - 🔑 CONFIGURATION Collections (4)
 - 📄 CONTENT Collections (3)
 - 🧑 PROFILES & CONFIGS Collections (5)
 - 4.2 İlişki Diyagramı
 - 5. API Dokümantasyonu
 - 5.1 API Katman Yapısı
 - 5.2 Tüm API Class'ları
 - 6. Servis Katmanı
 - 6.1 Servis Yapısı
 - 6.2 Tüm Servisler
 - 7. Ekran Akışı & Navigasyon
 - 7.1 Navigation Yapısı (React Navigation)
 - 7.2 Ana Ekran Listesi (60+ Ekran)
 - 🏠 AUTH SCREENS (4)
 - 🏠 HOME/DASHBOARD SCREENS (5)
 - ⚽ LEAGUE SCREENS (12)
 - ⚽ MATCH SCREENS (18)
 - 🧑 PLAYER SCREENS (10)
 - 🧑 PROFILE SCREENS (11)

- **i INFO SCREENS (6)**
- 7.3 Tab Bar Yapısı
- 8. Kullanıcı Senaryoları
 - 8.1 Senaryo 1: Yeni Kullanıcı (İlk Kurulum)
 - 8.2 Senaryo 2: Lig Yöneticisi (Lig Oluşturma)
 - 8.3 Senaryo 3: Maç Organizatörü (Maç Günü)
 - 8.4 Senaryo 4: Oyuncu (Maça Katılım)
 - 8.5 Senaryo 5: Dostluk Maçı Organizasyonu
- 9. Özellik Bayrakları (Feature Flags)
 - 9.1 Global Özellikler (app_config)
 - 9.2 Lig Bazlı Özellikler (league_settings)
- 10. Kurulum
 - 10.1 Ön Gereksinimler
 - 10.2 Projeyi Başlatma
 - 10.3 Environment Variables (.env)
- 11. Geliştirme Rehberi
 - 11.1 Coding Standards
 - Naming Conventions
 - Component Yapısı
 - 11.2 Git Workflow
- 12. Testing
 - 12.1 Unit Tests (Jest)
 - 12.2 Component Tests
- 13. Deployment
 - 13.1 Android Build
 - 13.2 iOS Build
 - 13.3 Firebase Deployment
- 14. Push Notifications (FCM)
 - 14.1 FCM Setup
- 15. State Management (Redux Toolkit)
 - 15.1 Store Setup
 - 15.2 Slice Örneği
- 16. Güvenlik
 - 16.1 Firestore Security Rules
- 17. Performans Optimizasyonu
 - 17.1 React Native Optimizasyon
- 18. Sık Kullanılan Komutlar
- 19. Troubleshooting
 - 19.1 Sık Karşılaşılan Hatalar
 - Metro Bundler Hatası
 - Android Build Hatası
 - iOS Build Hatası
 - Firebase Connection Hatası
- 20. Lisans & Katkı
 - 20.1 Lisans
 - 20.2 Katkıda Bulunma

- 21. İletişim
 - Sonuç
 - Kapsanan Konular:
 - Başlangıç Adımları:
-

1. Proje Hakkında

1.1 Genel Bakış

Spor Yönetim Uygulaması, kullanıcıların spor ligleri ve maçları organize edip yönetmesini sağlayan kapsamlı bir **React Native** mobil platformudur.

Ana Hedefler:

- Spor liglerini kolayca yönetmek
- Maç organizasyonunu otomatikleştirmek
- Detaylı oyuncu istatistikleri sunmak
- Adil ve şeffaf değerlendirme sistemi
- Ödeme takibini kolaylaştırmak
- Mobil-first yaklaşım

1.2 Temel Özellikler

Kategori	Özellikler
Lig Yönetimi	Lig oluşturma, sezon yönetimi, fikstür planlama, puan durumu
Maç Türleri	Lig maçları (puan etkiler), Dostluk maçları (opsiyonel)
Oyuncu Yönetimi	Premium, Direkt, Misafir, Kayıtlı, Yedek listeleri
Rating Sistemi	1-5 yıldız + kategorik değerlendirme (beceri, takım çalışması, sportmenlik)
MVP Sistemi	Otomatik MVP hesaplama (rating + performans bazlı)
Ödeme Takibi	Ödeme durumu, hatırlatmalar, onaylama
Hibrit Oyuncu Listeleri	Lig/Fikstür'den otomatik al veya maça özel düzenle
Push Bildirimler	Maç hatırlatmaları, davetler, skor güncellemeleri
Detaylı İstatistikler	Oyuncu/takım istatistikleri, trend analizi, karşılaştırma
Çoklu Spor Desteği	Futbol, Basketbol, Voleybol, Tenis, Masa Tenisi, Badminton
Sosyal Özellikler	Yorumlar, beğeniler, davetler, aktivite akışı
Offline Destek	Firebase offline persistence

2. Teknoloji Yığını

2.1 Frontend

Framework:	React Native 0.73+
Dil:	TypeScript 5.0+
Navigation:	React Navigation 6
State Management:	Redux Toolkit + RTK Query
UI Library:	React Native Paper / NativeBase / Tamagui
Icons:	React Native Vector Icons
Charts:	Victory Native / React Native Chart Kit
Forms:	React Hook Form + Zod
Camera/Gallery:	React Native Image Picker
Date Picker:	React Native Date Picker
Maps:	React Native Maps
Animations:	React Native Reanimated

2.2 Backend

Platform:	Firebase
Kimlik Doğrulama:	Firebase Authentication (Email/Google/Apple)
Veritabanı:	Cloud Firestore (24 Collections)
Depolama:	Firebase Storage (Profil fotoğrafları, belgeler)
Fonksiyonlar:	Cloud Functions (Node.js)
Push Bildirimleri:	Firebase Cloud Messaging (FCM)
Analytics:	Firebase Analytics
Crash Reporting:	Firebase Crashlytics
Performance:	Firebase Performance Monitoring
A/B Testing:	Firebase Remote Config

2.3 Geliştirme Araçları

Paket Yöneticisi:	npm / yarn
Build Sistemi:	Metro Bundler
Linter:	ESLint (@react-native-community)
Formatter:	Prettier
Type Checker:	TypeScript
Versiyon Kontrol:	Git
CI/CD:	GitHub Actions / Bitrise / Fastlane
Test Framework:	Jest
Test Library:	React Native Testing Library
E2E Testing:	Detox / Appium
Debugging:	Flipper / Reactotron
API Testing:	Postman / Insomnia
Code Coverage:	Istanbul
Documentation:	JSDoc / TypeDoc

3. Mimari Yapı

3.1 Proje Klasör Yapısı

```
sport-management-app/
  └── src/
    ├── navigation/          # React Navigation
    │   ├── AppNavigator.tsx  # Root navigator
    │   ├── AuthNavigator.tsx # Auth stack
    │   ├── MainNavigator.tsx # Main tab navigator
    │   ├── LeagueNavigator.tsx # League stack
    │   ├── MatchNavigator.tsx # Match stack
    │   ├── PlayerNavigator.tsx # Player stack
    │   ├── ProfileNavigator.tsx # Profile stack
    │   └── types.ts           # Navigation types

    ├── screens/             # Ekranlar (60+)
    │   └── auth/              # Auth ekranları
    │       ├── LoginScreen.tsx
    │       ├── RegisterScreen.tsx
    │       ├── ForgotPasswordScreen.tsx
    │       └── OnboardingScreen.tsx

    └── dashboard/           # Dashboard
        ├── DashboardScreen.tsx
        ├── HomeScreen.tsx
        └── NotificationsScreen.tsx

    └── leagues/             # Lig ekranları (12)
        ├── LeaguesListScreen.tsx
        ├── LeagueDetailScreen.tsx
        ├── CreateLeagueScreen.tsx
        ├── EditLeagueScreen.tsx
        ├── StandingsScreen.tsx
        ├── FixturesScreen.tsx
        ├── CreateFixtureScreen.tsx
        ├── EditFixtureScreen.tsx
        ├── MembersScreen.tsx
        ├── InviteMembersScreen.tsx
        ├── LeagueSettingsScreen.tsx
        └── SeasonHistoryScreen.tsx

    └── matches/             # Maç ekranları (18)
        ├── MatchesListScreen.tsx
        ├── MatchDetailScreen.tsx
        ├── CreateLeagueMatchScreen.tsx
        ├── CreateFriendlyMatchScreen.tsx
        ├── MatchRegistrationScreen.tsx
        ├── TeamBuilderScreen.tsx
        ├── PlayerSelectionScreen.tsx
        ├── PositionAssignmentScreen.tsx
        ├── MatchLiveScreen.tsx
        ├── ScoreInputScreen.tsx
        ├── GoalScorersScreen.tsx
        ├── MatchRatingScreen.tsx
        └── MVPSelectionScreen.tsx
```

```
    └── MatchPaymentsScreen.tsx
    └── MatchCommentsScreen.tsx
    └── MatchInvitationsScreen.tsx
    └── MatchHistoryScreen.tsx
    └── UpcomingMatchesScreen.tsx

    └── players/          # Oyuncu ekranları (10)
        ├── PlayersListScreen.tsx
        ├── PlayerDetailScreen.tsx
        ├── PlayerStatsScreen.tsx
        ├── PlayerHistoryScreen.tsx
        ├── PlayerAchievementsScreen.tsx
        ├── PlayerRatingHistoryScreen.tsx
        ├── PlayerCompareScreen.tsx
        ├── SearchPlayersScreen.tsx
        ├── InvitePlayerScreen.tsx
        └── BlockedPlayersScreen.tsx

    └── profile/          # Profil ekranları (11)
        ├── ProfileScreen.tsx
        ├── EditProfileScreen.tsx
        ├── MyStatsScreen.tsx
        ├── MyMatchesScreen.tsx
        ├── MyLeaguesScreen.tsx
        ├── MyAchievementsScreen.tsx
        ├── SettingsScreen.tsx
        ├── NotificationSettingsScreen.tsx
        ├── PrivacySettingsScreen.tsx
        ├── AppearanceSettingsScreen.tsx
        └── AccountSettingsScreen.tsx

    └── components/
        └── ui/          # React Native Bileşenleri
            # Temel UI bileşenleri
            ├── Button.tsx
            ├── Input.tsx
            ├── Card.tsx
            ├── Modal.tsx
            ├── Badge.tsx
            ├── Avatar.tsx
            ├── Loading.tsx
            ├── EmptyState.tsx
            ├── ErrorBoundary.tsx
            └── ...

        └── league/      # Lig bileşenleri
            ├── LeagueCard.tsx
            ├── LeagueForm.tsx
            ├── StandingsTable.tsx
            ├── SeasonSelector.tsx
            ├── FixtureCard.tsx
            └── ...

        └── match/       # Maç bileşenleri
            ├── MatchCard.tsx
```

```
    └── MatchListItem.tsx
    └── TeamBuilder.tsx
    └── ScoreInput.tsx
    └── PlayerSelector.tsx
    └── RatingStars.tsx
    └── CommentItem.tsx
    └── ...
    └── ...
    └── player/          # Oyuncu bileşenleri
        └── PlayerCard.tsx
        └── PlayerAvatar.tsx
        └── StatsWidget.tsx
        └── RatingForm.tsx
        └── PerformanceChart.tsx
        └── AchievementBadge.tsx
        └── ...
    └── ...
    └── layout/          # Layout bileşenleri
        └── Header.tsx
        └── TabBar.tsx
        └── DrawerContent.tsx
        └── BottomSheet.tsx
        └── SafeAreaView.tsx
    └── ...
    └── api/             # Firebase API Katmanı
        └── base/
            └── BaseAPI.ts      # Base CRUD operations
            └── ApiError.ts     # Error handling
            └── ApiLogger.ts    # Logging utility
            └── ...
            └── PlayerAPI.ts    # users collection
            └── LeagueAPI.ts   # leagues collection
            └── SeasonAPI.ts   # seasons collection
            └── FixtureAPI.ts  # fixtures collection
            └── MatchAPI.ts    # matches collection
            └── StandingsAPI.ts # standings collection
            └── PlayerStatsAPI.ts # player_stats collection
            └── RatingAPI.ts   # ratings collection
            └── CommentAPI.ts  # comments collection
            └── InvitationAPI.ts # invitations collection
            └── NotificationAPI.ts # notifications collection
            └── ActivityLogAPI.ts # activity_logs collection
            └── AppConfigAPI.ts  # app_config collection
            └── UserSettingsAPI.ts # user_settings collection
            └── LeagueSettingsAPI.ts # league_settings collection
            └── FAQAPI.ts       # faqs collection
            └── AnnouncementAPI.ts # announcements collection
            └── FeedbackAPI.ts  # feedbacks collection
            └── PlayerProfileAPI.ts # player_profiles collection
            └── RatingProfileAPI.ts # player_rating_profiles collection
            └── FriendlyConfigAPI.ts # friendly_match_configs collection
            └── index.ts         # Tüm API'leri export
        └── ...
    └── services/         # İş Mantığı Katmanı
```

```
    ├── AuthService.ts          # Kimlik doğrulama
    ├── LeagueService.ts        # Lig işlemleri
    ├── MatchService.ts         # Maç yaşam döngüsü
    ├── PlayerService.ts        # Oyuncu işlemleri
    ├── StandingsService.ts     # Puan durumu
    ├── RatingService.ts        # Rating hesaplama
    ├── PaymentService.ts       # Ödeme yönetimi
    ├── NotificationService.ts  # FCM entegrasyonu
    ├── StorageService.ts       # Firebase Storage
    ├── AnalyticsService.ts     # Firebase Analytics
    └── index.ts

    └── store/                  # Redux Store
        ├── slices/
        │   ├── authSlice.ts
        │   ├── leagueSlice.ts
        │   ├── matchSlice.ts
        │   ├── playerSlice.ts
        │   ├── notificationSlice.ts
        │   └── uiSlice.ts
        ├── hooks.ts              # Typed hooks
        └── store.ts               # Store configuration

    └── types/                  # TypeScript Tip Tanımları
        ├── index.ts             # Tüm type'lar (24 collection)
        ├── navigation.types.ts  # Navigation types
        └── api.types.ts         # API types

    └── hooks/                  # Custom React Hooks
        ├── useAuth.ts
        ├── useLeague.ts
        ├── useMatch.ts
        ├── usePlayer.ts
        ├── useNotifications.ts
        ├── usePushNotifications.ts
        ├── useDebounce.ts
        ├── useKeyboard.ts
        └── ...

    └── utils/                  # Yardımcı Fonksiyonlar
        ├── date.ts              # Tarih formatla
        ├── format.ts             # String/number formatla
        ├── validation.ts         # Form validasyon
        ├── permissions.ts        # Permission check
        ├── storage.ts            # AsyncStorage wrapper
        ├── encryption.ts         # Şifreleme
        └── helpers.ts            # Genel yardımcılar

    └── config/                 # Konfigürasyon
        ├── firebase.config.ts   # Firebase init
        ├── constants.ts          # Sabitler
        ├── theme.ts              # Tema (renkler, fontlar)
        ├── routes.ts             # Route isimleri
        └── env.ts                # Environment wrapper
```

```
    └── assets/                      # Statik Dosyalar
        ├── images/
        ├── icons/
        ├── fonts/
        └── animations/                # Lottie files

    └── styles/                      # Global Stiller
        ├── colors.ts
        ├── typography.ts
        ├── spacing.ts
        ├── shadows.ts
        └── theme.ts

└── android/                      # Android Native Kod
    ├── app/
    │   ├── src/
    │   ├── build.gradle
    │   └── google-services.json
    └── build.gradle

└── ios/                          # iOS Native Kod
    ├── YourApp/
    │   ├── Info.plist
    │   └── GoogleService-Info.plist
    ├── YourApp.xcodeproj/
    └── Podfile

└── firebase/                     # Firebase Konfigürasyon
    ├── firestore.rules
    ├── firestore.indexes.json
    ├── firebase.json
    └── functions/
        ├── src/
        │   ├── index.ts
        │   ├── notifications.ts
        │   └── triggers.ts
        └── package.json

└── __tests__/                     # Test dosyaları
    ├── components/
    ├── services/
    ├── utils/
    └── integration/

└── e2e/                          # E2E test (Detox)
    ├── config.json
    └── firstTest.e2e.ts

└── .env                           # Environment variables
└── .env.example
└── .eslintrc.js
└── .prettierrc
└── .gitignore
```

```
└── app.json  
└── babel.config.js  
└── metro.config.js  
└── package.json  
└── tsconfig.json  
└── README.md
```

3.2 Mimari Katmanlar

PRESENTATION LAYER (UI)
React Native Screens & Components

- Navigation (React Navigation)
- Screens (60+ ekran)
- Components (UI/League/Match/Player)
- Modals, BottomSheets, Animations



STATE MANAGEMENT LAYER
Redux Toolkit / Zustand

- Auth State
- League/Match/Player State
- UI State (Loading, Errors, Modals)
- Notification State



BUSINESS LOGIC LAYER
Services & Custom Hooks

- LeagueService
- MatchService (Lifecycle)
- PlayerService
- RatingService (MVP calc)
- NotificationService (FCM)



DATA ACCESS LAYER
Firebase SDK & API Classes

- BaseAPI (CRUD operations)
- 22+ Specialized API classes
- Offline Persistence
- Cache Strategy



FIREBASE BACKEND
Firestore, Auth, Storage, FCM

- 24 Collections
- Cloud Functions (Triggers, HTTPS)
- Push Notifications (FCM)
- Analytics & Crashlytics

4. Veritabanı Şeması

4.1 Collection'lар (24 Adet)

⌚ CORE Collections (7)

#	Collection	Açıklama	Cache Alanları	İlişkiler
1	users	Kullanıcı/Oyuncu temel bilgileri	-	→ user_settings, player_profiles
2	leagues	Lig tanımı	totalSeasons, totalMatches, totalMembers	→ seasons, fixtures, matches
3	seasons	Sezon tanımı (örn: 2025 İlkbahar)	summary (topScorer, MVP)	→ league, standings
4	fixtures	Tekrarlayan maç şablonu	totalMatches, nextMatchDate	→ league, matches
5	matches	Maç (League/Friendly)	ratingSummary, mvp, totalComments	→ league, season, ratings
6	standings	Puan durumu (sezon bazlı)	playerName, performance.*	→ league, season
7	player_stats	Oyuncu istatistikleri	hesaplanmış metrikler	→ player, league, season

💬 INTERACTION Collections (5)

#	Collection	Açıklama	Cache Alanları	İlişkiler
8	ratings	Maç sonrası oyuncu puanlaması	-	→ match, rater, rated player
9	comments	Maç yorumları	playerName, playerPhoto	→ match, player
10	invitations	Maça davet sistemi	inviterName, inviteeName	→ match, inviter, invitee
11	notifications	Kullanıcı bildirimleri	-	→ user, related entity

#	Collection	Açıklama	Cache Alanları	İlişkiler
12	activity_logs	Aktivite kayıtları (audit trail)	userName, entityName	→ user, entity
⚙️ CONFIGURATION Collections (4)				
#	Collection	Açıklama	Cache Alanları	İlişkiler
13	app_config	Global ayarlar (singleton, id='main')	-	-
14	user_settings	Kullanıcı özel ayarları	quickActions	→ user
15	league_settings	Lig özel ayarları ve kuralları	-	→ league
16	system_logs	Sistem hata ve bilgi logları	-	-
📄 CONTENT Collections (3)				
#	Collection	Açıklama	Cache Alanları	İlişkiler
17	faqs	Sıkça sorulan sorular	views, helpful, notHelpful	-
18	announcements	Duyurular (global/lig bazlı)	stats (views, clicks)	→ league (optional)
19	feedbacks	Kullanıcı geri bildirimleri	userName, userEmail	→ user
👤 PROFILES & CONFIGS Collections (5)				
#	Collection	Açıklama	Cache Alanları	İlişkiler
20	player_profiles	Oyuncu genel profil özeti (cross-league)	TÜM ALAN CACHE	→ player
21	player_rating_profiles	Oyuncu rating profili (detaylı trend)	TÜM ALAN CACHE	→ player, league, season
22	friendly_match_configs	Dostluk maç oluşturma tercihleri	recentSettings	→ organizer

4.2 İlişki Diyagramı

```

users (IPlayer)
↓
├→ leagues (member, admin, creator)
├→ user_settings (1:1)
├→ player_profiles (1:1)
├→ player_rating_profiles (1:many - per league/season)
├→ friendly_match_configs (1:1)
├→ notifications (1:many)
└→ feedbacks (1:many)
  
```

```

leagues (ILeague)
↓
├→ seasons (1:many)
├→ fixtures (1:many)
├→ matches (1:many)
├→ standings (many via seasons)
├→ player_stats (many via seasons)
└→ league_settings (1:1)
└→ announcements (many)

seasons (ISession)
↓
├→ standings (1:1)
├→ matches (1:many)
└→ player_stats (1:many)

fixtures (IFixture)
↓
└→ matches (1:many - only LEAGUE type)

matches (IMatch)
↓
├→ ratings (1:many)
├→ comments (1:many)
├→ invitations (1:many)
└→ notifications (1:many)

```

5. API Dokümantasyonu

5.1 API Katman Yapısı

BaseAPI tüm collection'lar için temel CRUD işlemlerini sağlar:

```

export class BaseAPI<T extends { id?: string }> {
  // CRUD Operations
  async create(data: Omit<T, 'id'>): Promise<ApiResponse<T>>
  async createWithId(id: string, data: Omit<T, 'id'>): Promise<ApiResponse<T>>
  async getById(id: string): Promise<ApiResponse<T>>
  async getAll(options?: QueryOptions): Promise<ApiResponse<T[]>>
  async getPaginated(options: PaginationOptions): Promise<PaginatedResult<T>>
  async update(id: string, data: Partial<T>): Promise<ApiResponse<T>>
  async delete(id: string): Promise<ApiResponse<void>>

  // Utility
  async exists(id: string): Promise<ApiResponse<boolean>>
  async count(options?: QueryOptions): Promise<ApiResponse<number>>

  // Batch Operations
  createBatch(): WriteBatch
}

```

```
    async executeBatch(batch: WriteBatch): Promise<ApiResponse<void>>
}
```

5.2 Tüm API Class'ları

API Class	Collection	Özel Methodlar
PlayerAPI	users	getByEmail(), getByPhone(), searchPlayers(), getByIds(), updateLastLogin()
LeagueAPI	leagues	getBySportType(), getByCreator(), getByMember(), addMember(), removeMember(), addAdmin(), removeAdmin(), isAdmin(), updateDefaultPlayers()
SeasonAPI	seasons	getByLeague(), getActiveSeason(), getByStatus(), updateStatus(), updateSummary(), getLatestSeasonNumber()
FixtureAPI	fixtures	getByLeague(), getActiveFixtures(), updatePlayerListConfig(), updateSchedule(), updateVenue(), toggleStatus(), incrementTotalMatches()
MatchAPI	matches	getByLeague(), getBySeason(), getByFixture(), getByType(), getByStatus(), getUpcomingMatches(), getByOrganizer(), registerPlayer(), unregisterPlayer(), addGuestPlayer(), setTeams(), updateScore(), updateMVP(), updatePayment(), getPlayerMatches()
StandingsAPI	standings	getByLeagueAndSeason(), updatePlayerStanding(), addPlayer(), sortStandings()
PlayerStatsAPI	player_stats	getByPlayer(), getBySeason(), updateStats(), calculateMetrics(), recalculateAll()
RatingAPI	ratings	getByMatch(), getByPlayer(), calculateAverage(), getMVPCandidate(), getTopRated()
CommentAPI	comments	getByMatch(), approve(), toggleLike(), getByPlayer()
InvitationAPI	invitations	getByMatch(), getByUser(), accept(), decline(), expire()
NotificationAPI	notifications	getByUser(), getUnread(), markAsRead(), markAllAsRead(), sendPush()
ActivityLogAPI	activity_logs	getByEntity(), getByUser(), getRecent()
AppConfigAPI	app_config	get() (singleton), update(), getFeatureFlag()

API Class	Collection	Özel Methodlar
UserSettingsAPI	user_settings	getByUser(), updateNotifications(), updatePrivacy(), updatePreferences()
LeagueSettingsAPI	league_settings	getByLeague(), updateRules(), updateMatchRules(), updateRating()
FAQAPI	faqs	getByCategory(), getPublished(), incrementViews(), markHelpful()
AnnouncementAPI	announcements	getActive(), getByScope(), incrementViews()
FeedbackAPI	feedbacks	getByUser(), getByStatus(), updateStatus(), addResponse()
PlayerProfileAPI	player_profiles	getByPlayer(), updateSummary(), addAchievement()
RatingProfileAPI	player_rating_profiles	getByPlayer(), updateProfile(), calculateTrend()
FriendlyConfigAPI	friendly_match_configs	getByOrganizer(), updateDefaults(), addTemplate(), updateRecentSettings()

6. Servis Katmanı

6.1 Servis Yapısı

Servis katmanı, API'leri kullanarak karmaşık iş mantığını yönetir.

Örnek: MatchService

```
export class MatchService {
    // =====
    // MAÇ YAŞAM DÖNGÜSÜ
    // =====
    static async createLeagueMatch(fixtureId: string, date: Date): Promise<IMatch>
    static async createFriendlyMatch(data: CreateFriendlyMatchDTO): Promise<IMatch>
    static async openRegistration(matchId: string): Promise<void>
    static async closeRegistration(matchId: string): Promise<void>
    static async buildTeams(matchId: string, algorithm: 'random' | 'rating' | 'position'): Promise<void>
    static async startMatch(matchId: string): Promise<void>
    static async submitScore(matchId: string, scoreDTO: ScoreDTO): Promise<void>
    static async completeMatch(matchId: string): Promise<void>
    static async cancelMatch(matchId: string, reason: string): Promise<void>

    // =====
    // OYUNCU YÖNETİMİ
    // =====
    static async registerPlayerToMatch(matchId: string, playerId: string): Promise<void>
    static async unregisterPlayer(matchId: string, playerId: string): Promise<void>
    static async moveToReserve(matchId: string, playerId: string): Promise<void>
```

```

static async addGuestPlayer(matchId: string, playerId: string): Promise<void>

// =====
// SKOR & RATING
// =====
static async calculateMVP(matchId: string): Promise<string>
static async requestRatings(matchId: string): Promise<void>
static async updateRatingSummary(matchId: string): Promise<void>

// =====
// HELPER METHODS
// =====
static async getMatchById(matchId: string): Promise<ApiResponse<IMatch>>
static async getMatchesByLeague(leagueId: string):
Promise<ApiResponse<IMatch[]>>
static async getUpcomingMatches(userId: string): Promise<IMatch[]>
}

```

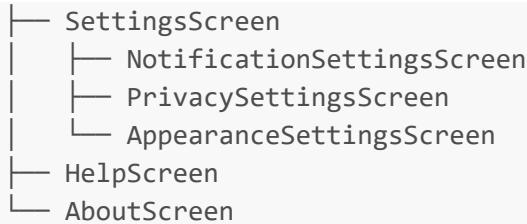
6.2 Tüm Servisler

Servis	Sorumluluk	Ana Metodlar
AuthService	Kimlik doğrulama, token yönetimi	<code>login()</code> , <code>register()</code> , <code>logout()</code> , <code>resetPassword()</code> , <code>getCurrentUser()</code>
LeagueService	Lig oluşturma, üye yönetimi, sezon geçişleri	<code>createLeague()</code> , <code>addMember()</code> , <code>startNewSeason()</code> , <code>archiveOldSeasons()</code>
MatchService	Maç yaşam döngüsü, takım kurma, skor yönetimi	<code>createMatch()</code> , <code>buildTeams()</code> , <code>submitScore()</code> , <code>completeMatch()</code>
PlayerService	Oyuncu profili, istatistik hesaplama	<code>updateProfile()</code> , <code>calculateStats()</code> , <code>getPlayerSummary()</code>
StandingsService	Puan durumu güncelleme, sıralama	<code>updateAfterMatch()</code> , <code>recalculateStandings()</code> , <code>sortByPoints()</code>
RatingService	Rating hesaplama, MVP seçimi, trend analizi	<code>calculateAverage()</code> , <code>selectMVP()</code> , <code>analyzeTrend()</code>
PaymentService	Ödeme takibi, hatırlatmalar	<code>trackPayment()</code> , <code>confirmPayment()</code> , <code>sendReminders()</code>
NotificationService	Bildirim gönderme, FCM entegrasyonu	<code>sendPush()</code> , <code>sendBulk()</code> , <code>scheduleNotification()</code>
StorageService	Dosya yükleme (profil fotoğrafı, belgeler)	<code>uploadImage()</code> , <code>deleteFile()</code> , <code>getDownloadURL()</code>
AnalyticsService	Firebase Analytics entegrasyonu	<code>logEvent()</code> , <code>setUserProperty()</code> , <code>logScreenView()</code>

7. Ekran Akışı & Navigasyon

7.1 Navigation Yapısı (React Navigation)





7.2 Ana Ekran Listesi (60+ Ekran)

AUTH SCREENS (4)

Ekran	Route	Açıklama
Onboarding	/onboarding	İlk açılış rehberi (3-4 sayfa)
Login	/login	Email/Şifre giriş
Register	/register	Kayıt formu
ForgotPassword	/forgot-password	Şifre sıfırlama

HOME/DASHBOARD SCREENS (5)

Ekran	Route	Açıklama
Dashboard	/dashboard	Ana ekran - özet kartlar, yaklaşan maçlar
Notifications	/notifications	Bildirimler listesi (okunmuş/okunmamış)
Announcements	/announcements	Duyurular
QuickActions	/quick-actions	Hızlı işlemler (maç oluştur, kayıt ol)
Search	/search	Global arama (lig, maç, oyuncu)

LEAGUE SCREENS (12)

Ekran	Route	Açıklama
LeaguesList	/leagues	Lig listesi (katıldığım/tüm ligler)
LeagueDetail	/leagues/:id	Lig ana sayfası (overview, son maçlar)
CreateLeague	/leagues/create	Yeni lig oluştur formu
EditLeague	/leagues/:id/edit	Lig düzenleme
Standings	/leagues/:id/standings	Puan durumu tablosu
Fixtures	/leagues/:id/fixtures	Fikstür listesi
CreateFixture	/leagues/:id/fixtures/create	Fikstür oluştur
EditFixture	/leagues/:id/fixtures/:fixtureId/edit	Fikstür düzenle

Ekran	Route	Açıklama
Members	/leagues/:id/members	Lig üyeleri listesi
InviteMembers	/leagues/:id/invite	Üye davet et
LeagueSettings	/leagues/:id/settings	Lig ayarları (kurallar, ödeme vb.)
SeasonHistory	/leagues/:id/seasons	Geçmiş sezonlar

⚽ MATCH SCREENS (18)

Ekran	Route	Açıklama
MatchesList	/matches	Maç listesi (yaklaşan/geçmiş)
MatchDetail	/matches/:id	Maç detay sayfası
CreateLeagueMatch	/matches/create/league	Lig maçı oluştur
CreateFriendlyMatch	/matches/create/friendly	Dostluk maçı oluştur
MatchRegistration	/matches/:id/register	Maça kayıt ol
TeamBuilder	/matches/:id/teams	Takımları oluştur/düzenle
PlayerSelection	/matches/:id/players	Oyuncu seçimi
PositionAssignment	/matches/:id/positions	Pozisyon atama
MatchLive	/matches/:id/live	Canlı maç takibi (skor, olaylar)
ScoreInput	/matches/:id/score	Skor girişi
GoalScorers	/matches/:id/scorers	Gol/Asist girişi
MatchRating	/matches/:id/rate	Oyuncuları değerlendir
MVPSelection	/matches/:id/mvp	MVP seçimi (manuel)
MatchPayments	/matches/:id/payments	Ödeme durumu listesi
MatchComments	/matches/:id/comments	Yorumlar
MatchInvitations	/matches/:id/invitations	Davet edilen oyuncular
MatchHistory	/matches/history	Geçmiş maçlar
UpcomingMatches	/matches/upcoming	Yaklaşan maçlar

㉚ PLAYER SCREENS (10)

Ekran	Route	Açıklama
PlayersList	/players	Oyuncu listesi (lig bazlı)
PlayerDetail	/players/:id	Oyuncu profili

Ekran	Route	Açıklama
PlayerStats	/players/:id/stats	Detaylı istatistikler
PlayerHistory	/players/:id/history	Maç geçmişi
PlayerAchievements	/players/:id/achievements	Başarılar ve rozetler
PlayerRatingHistory	/players/:id/ratings	Rating geçmişi (grafik)
PlayerCompare	/players/compare	Oyuncu karşılaştırma
SearchPlayers	/players/search	Oyuncu ara
InvitePlayer	/players/invite	Oyuncu davet et
BlockedPlayers	/players/blocked	Engellenen oyuncular

👤 PROFILE SCREENS (11)

Ekran	Route	Açıklama
Profile	/profile	Profilim
EditProfile	/profile/edit	Profil düzenle
MyStats	/profile/stats	İstatistiklerim
MyMatches	/profile/matches	Maçlarım
MyLeagues	/profile/leagues	Ligallerim
MyAchievements	/profile/achievements	Başarılarım
Settings	/settings	Ayarlar
NotificationSettings	/settings/notifications	Bildirim ayarları
PrivacySettings	/settings/privacy	Gizlilik ayarları
AppearanceSettings	/settings/appearance	Görünüm (tema, dil)
AccountSettings	/settings/account	Hesap (şifre değiştir, hesap sil)

ℹ INFO SCREENS (6)

Ekran	Route	Açıklama
Help	/help	Yardım merkezi
FAQ	/faq	Sıkça sorulan sorular
Contact	/contact	İletişim formu
Feedback	/feedback	Geri bildirim gönder
About	/about	Uygulama hakkında

Ecran	Route	Açıklama
Terms	/terms	Kullanım şartları

TOPLAM: 66 Ecran**7.3 Tab Bar Yapısı**

```

<Tab.Navigator
  screenOptions={{
    tabBarActiveTintColor: '#16a34a',
    tabBarInactiveTintColor: '#6b7280',
    tabBarStyle: {
      height: 60,
      paddingBottom: 8,
      paddingTop: 8,
    }
  }}
>
  <Tab.Screen
    name="Home"
    component={HomeNavigator}
    options={{
      tabBarIcon: ({ color, size }) => <Home color={color} size={size} />,
      tabBarLabel: 'Ana Sayfa',
    }}
  />

  <Tab.Screen
    name="Leagues"
    component={LeagueNavigator}
    options={{
      tabBarIcon: ({ color, size }) => <Trophy color={color} size={size} />,
      tabBarLabel: 'Ligler',
    }}
  />

  <Tab.Screen
    name="Matches"
    component={MatchNavigator}
    options={{
      tabBarIcon: ({ color, size }) => <Calendar color={color} size={size} />,
      tabBarLabel: 'Maçlar',
      tabBarBadge: upcomingMatchCount > 0 ? upcomingMatchCount : undefined,
    }}
  />

  <Tab.Screen
    name="Players"
    component={PlayersNavigator}
    options={{
      tabBarIcon: ({ color, size }) => <Users color={color} size={size} />,
    }}
  />

```

```
        tabBarLabel: 'Oyuncular',
    )}
/>

<Tab.Screen
  name="Profile"
  component={ProfileNavigator}
  options={{
    tabBarIcon: ({ color, size }) => <User color={color} size={size} />,
    tabBarLabel: 'Profil',
    tabBarBadge: unreadNotifications > 0 ? unreadNotifications : undefined,
  }}
/>
</Tab.Navigator>
```

8. Kullanıcı Senaryoları

8.1 Senaryo 1: Yeni Kullanıcı (İlk Kurulum)

- █ ADIM 1: UYGULAMAYI İNDİR VE AÇ
 - Onboarding ekranları göster (3 sayfa)
 - |→ Sayfa 1: "Spor liglerinizi yönetin"
 - |→ Sayfa 2: "Maçları organize edin"
 - |→ Sayfa 3: "İstatistiklerinizi takip edin"
 - "Başla" butonu → Kayıt/Giriş seçimi
- █ ADIM 2: KAYIT OL
 - Email ve şifre gir
 - Ad, soyad bilgileri
 - Favori sporları seç (Futbol, Basketbol...)
 - Tercih edilen pozisyonları seç
 - Profil fotoğrafı yükle (opsiyonel)
- █ ADIM 3: BİLDİRİM İZİNLERİ
 - Push notification izni iste
 - FCM token'ı kaydet
- █ ADIM 4: İLK LİG'E KATIL
 - Dashboard açılır
 - "Lig Ara" butonu
 - Yaklaşan/popüler ligleri göster
 - Bir lige tıkla → Detayları gör
 - "Katıl" butonu
 - Admin onayı bekle → Bildirim gelir
- █ ADIM 5: İLK MAÇA KAYIT OL
 - "Maçlar" sekmesine git
 - Yaklaşan maçları gör
 - Bir maça tıkla
 - "Kayıt Ol" butonu

- ↳ Tercih edilen pozisyon seç (opsiyonel)
- ↳ Onay: "Kayıt başarılı! 9. sıradasınız"

TAMAMLANDI

8.2 Senaryo 2: Lig Yöneticisi (Lig Oluşturma)

ADIM 1: YENİ LİG OLUŞTUR

- ↳ Dashboard → "+" butonu → "Lig Oluştur"
- ↳ Lig bilgilerini gir
 - ↳ Lig adı: "Architect Halı Saha Ligi"
 - ↳ Spor türü: Futbol
 - ↳ Sezon süresi: 6 ay
 - ↳ Başlangıç tarihi: 15 Ekim 2025
 - ↳ Varsayılan ayarlar (kadro sayısı, ücret vb.)
- ↳ "Oluştur" butonu

ADIM 2: ÜYE DAVET ET

- ↳ Lig detay → "Üye Davet Et"
- ↳ Seçenekler:
 - ↳ Telefon rehberinden seç
 - ↳ Email ile davet
 - ↳ Davet linki paylaş (WhatsApp, SMS)
- ↳ Davetler gönderildi

ADIM 3: İLK FİKSTÜR OLUŞTUR

- ↳ Lig detay → "Fikstürler" sekmesi
- ↳ "+" butonu → "Fikstür Ekle"
- ↳ Fikstür bilgileri
 - ↳ İsim: "Salı Maçı"
 - ↳ Gün: Salı
 - ↳ Saat: 19:00
 - ↳ Periyot: Her hafta
 - ↳ Kadro sayısı: 10 kişi
 - ↳ Yedek: 2 kişi
 - ↳ Saha: Architect Halısaха
 - ↳ Ücret: 100 TL/kİŞİ
- ↳ "Oluştur" butonu

ADIM 4: İLK MAÇ OLUŞTUR

- ↳ Fikstür detay → "Maç Oluştur"
- ↳ Tarih seç: 15 Ekim 2025, 19:00
- ↳ Maç oluşturuldu (status: 'created')
- ↳ Otomatik olarak 'registration_open' durumuna geçer
- ↳ Tüm lig üyelerine bildirim gönderilir

TAMAMLANDI

8.3 Senaryo 3: Maç Organizatörü (Maç Günü)

- ⌚ MAÇ GÜNÜ SABAHI (08:00)
 - Bildirim gelir: "Bugün 19:00'da maç var!"
 - Uygulamayı aç
 - Dashboard'da "Bugünün Maçı" kartı
 - 12 oyuncu kayıtlı (10 kadro + 2 yedek) ✓
- ⌚ KAYITLARI KAPAT (17:00 - 2 saat önce)
 - Maç detay → "Kayıtları Kapat" butonu
 - Status: 'registration_closed'
 - Tüm kayıtlılara bildirim: "Kayıtlar kapandı"
- ⌚ TAKIMLARI KUR (18:00 - 1 saat önce)
 - Maç detay → "Takım Kur" butonu
 - Algoritma seç:
 - Otomatik (Rating bazlı) ← Seçildi
 - Rastgele
 - Manuel
 - Takımlar oluşturuldu:
 - Takım 1: 6 oyuncu (rating ortalaması: 4.3)
 - Takım 2: 6 oyuncu (rating ortalaması: 4.2)
 - Pozisyonları ata (opsiyonel)
 - Status: 'teams_set'
 - Tüm oyunculara bildirim: "Takımlar belirlendi"
- ⚽ MAÇ BAŞLADI (19:00)
 - Maç detay → "Maçı Başlat" butonu
 - Status: 'in_progress'
 - Canlı skor takibi (opsiyonel)
- 🏁 MAÇ BİTTİ (20:00)
 - "Skoru Gir" butonu
 - Skor gir:
 - Takım 1: 5
 - Takım 2: 3
 - Gol atan oyuncuları seç:
 - Ahmet: 2 gol, 1 asist
 - Mehmet: 1 gol, 1 asist
 - Ali: 2 gol
 - "Skoru Kaydet" butonu
- ☑ MAÇI TAMAMLA
 - "Maçı Tamamla" butonu
 - Otomatik işlemler başlar:
 - Puan durumu güncellenir
 - Oyuncu istatistikleri güncellenir
 - MVP otomatik hesaplanır (Ahmet - 2 gol, 4.8 rating)
 - Rating istekleri gönderilir (2 saat sonra)
 - Status: 'completed'
 - Tüm oyunculara bildirim: "Maç tamamlandı"
- ☑ TAMAMLANDI

8.4 Senaryo 4: Oyuncu (Maça Katılım)

⌚ BİLDİRİM GELDİ → Push notification: "Salı maçı için kayıtlar açıldı!" → Uygulamayı aç → Bildirime tıkla → Maç detayı ⚽ MAÇA KAYIT OL → Maç detayını incele: | | Tarih: 15 Ekim, 19:00 | | Saha: Architect Halısaşa | | Ücret: 100 TL | | Kayıtlı: 8/10 | | Yedek: 0/2 → "Kayıt Ol" butonu → Tercih edilen pozisyon seç: "Forvet" → Onay: "Kayıt başarılı! 9. sıradasınız" 📊 MAÇ GÜNÜ → Hatırlatma bildirimi (2 saat önce): "17:00'da maç!" → Takımlar açıklandı bildirimi → Takım 2'desin, pozisyon: Forvet → Sahaya git ⚽ MAÇ OYNANDI → 2 gol attın, 1 asist yaptı → Maç bitti: 3-5 (Kaybettik) ★ MAÇ SONRASI (2 saat sonra) → Bildirim: "Maç bitti! Arkadaşlarını değerlendir" → Rating ekranı açılır → 10 oyuncuyu değerlendir (1-5 yıldız) | | Ahmet: 5 yıldız (MVP!) | | | | Beceri: 5 | | | | Takım çalışması: 5 | | | | Sportmenlik: 5 | | Mehmet: 4 yıldız | | Diğer oyuncular... → Yorum yaz (opsiyonel): "Harika maç oldu!" → "Gönder" butonu 📉 İSTATİSTİKLERİMİ GÖR → Profil → "İstatistiklerim" → Bu maç: | | 2 gol, 1 asist | | Rating aldın: 4.2 | | Sonuç: Kaybettik (3-5) → Sezon toplam: | | 12 maç, 8 gol, 5 asist | | Ortalama rating: 4.3 | | Puan durumunda 3. sıradasınız | | 2 kez MVP seçildiniz ✅ TAMAMLANDI

8.5 Senaryo 5: Dostluk Maçı Organizasyonu

⚽ DOSTLUK MAÇI OLUŞTUR

- "Maçlar" → "+" butonu
- "Dostluk Maçı" seç
- Maç bilgileri:
 - | İsim: "Cumartesi Maçı"
 - | Tarih: 18 Ekim, 10:00
 - | Saha: Park Halısaşa
 - | Kadro: 12 kişi
 - | Yedek: 2 kişi
 - | Ücret: 80 TL
 - | Görünürlük: Herkes görebilir
- Ayarlar:
 - | İstatistikleri etkiler: Evet
 - | Puan durumunu etkiler: Hayır
- "Oluştur" butonu

👥 OYUNCU DAVET ET

- "Oyuncu Davet Et" butonu
- Seçenekler:
 - | Favorilerimden seç (sık oynadıklarım)
 - | Oyuncu ara
 - | Davet linki paylaş
- 15 oyuncu seç
- Davetler gönderildi

📊 KAYITLARI TAKİP ET

- Maç detay → "Davetler" sekmesi
- Durum:
 - | Kabul eden: 10
 - | Reddeden: 2
 - | Bekleyen: 3
- Yeterli oyuncu var ✓

⚽ MAÇ GÜNÜ

- ↳ Normal maç akışı gibi devam eder
- ↳ Takım kur, oyna, skor gir
- ↳ Fark: Puan durumunu etkilemez
- ↳ Ama istatistikler güncellenir

TAMAMLANDI

9. Özellik Bayrakları (Feature Flags)

9.1 Global Özellikler (app_config)

```
features: {  
    friendlyMatches: true,           // Dostluk maçları aktif  
    ratingSystem: true,             // Değerlendirme sistemi  
    commentSystem: true,            // Yorum sistemi  
    paymentTracking: true,          // Ödeme takibi  
    mvpSystem: true,                // MVP sistemi  
    notifications: true,            // Bildirimler  
    invitations: true,              // Davet sistemi  
    multiLeague: true,              // Çoklu lig desteği  
}
```

9.2 Lig Bazlı Özellikler (league_settings)

```
rules: {  
    lateArrivalPenalty: 10,          // Geç gelme cezası (TL)  
    absentWithoutNoticePenalty: 50,   // Haber vermeden gelmeme cezası  
    yellowCardFine: 20,              // Sarı kart cezası  
    redCardFine: 50,                // Kırmızı kart cezası  
    minAttendanceRate: 75,           // Min katılım oranı (%)  
}  
  
matchRules: {  
    allowGuestPlayers: true,  
    maxGuestPlayersPerMatch: 2,  
    guestPlayerPriceMultiplier: 1.5, // Misafir %50 fazla öder  
    autoAssignTeams: true,  
    teamBalanceAlgorithm: 'rating', // 'random' | 'rating' | 'position'  
}  
  
rating: {  
    enabled: true,                  // Zorunlu değil  
    mandatory: false,                // Anonim değerlendirme  
    anonymous: true,                 // 48 saat içinde  
    ratingDeadlineHours: 48,         // MVP için min 5 rating  
    minRatingsForMVP: 5,              // Kategorik değerlendirme  
    allowCategoryRating: true,  
}
```

10. Kurulum

10.1 Ön Gereksinimler

```
# Node.js (v18+)
node --version

# npm veya yarn
npm --version
yarn --version

# React Native CLI
npm install -g react-native-cli

# iOS için (macOS)
xcode-select --install
sudo gem install cocoapods

# Android için
# Android Studio ve SDK kurulu olmalı
```

10.2 Projeyi Başlatma

```
# 1. Repo'yu clone'la
git clone https://github.com/your-repo/sport-management-app.git
cd sport-management-app

# 2. Bağımlılıkları yükle
npm install
# veya
yarn install

# 3. iOS için pod install (macOS)
cd ios
pod install
cd ..

# 4. Environment variables
cp .env.example .env
# .env dosyasını düzenle (Firebase credentials)

# 5. Uygulamayı başlat
npm run android
# veya
npm run ios
```

10.3 Environment Variables (.env)

```
# Firebase Configuration
FIREBASE_API_KEY=your_api_key_here
FIREBASE_AUTH_DOMAIN=your_project.firebaseio.com
FIREBASE_PROJECT_ID=your_project_id
FIREBASE_STORAGE_BUCKET=your_project.appspot.com
FIREBASE_MESSAGING_SENDER_ID=123456789
FIREBASE_APP_ID=1:123456789:android:abc123

# API Configuration
API_BASE_URL=https://your-api.com
API_TIMEOUT=30000

# Environment
ENVIRONMENT=development

# Feature Flags (Override için)
ENABLE_ANALYTICS=true
ENABLE_CRASHLYTICS=false
```

11. Geliştirme Rehberi

11.1 Coding Standards

Naming Conventions

```
// Dosya isimleri
Components: PascalCase (PlayerCard.tsx)
Screens: PascalCase + Screen suffix (LeagueDetailScreen.tsx)
Services: PascalCase + Service suffix (MatchService.ts)
Utils: camelCase (formatDate.ts)
Constants: UPPER_SNAKE_CASE (API_ENDPOINTS.ts)

// Değişken isimleri
const playerName = 'Ahmet'; // camelCase
const MAX_PLAYERS = 10; // UPPER_SNAKE_CASE (constants)
interface IPlayer {...} // PascalCase + I prefix
type PlayerType = ... // PascalCase + Type suffix
enum MatchStatus {...} // PascalCase
```

Component Yapısı

```
import React from 'react';
import { View, Text, StyleSheet, TouchableOpacity } from 'react-native';
```

```
// =====
// TYPES
// =====
interface MatchCardProps {
  match: IMatch;
  onPress?: () => void;
}

// =====
// COMPONENT
// =====
export const MatchCard: React.FC<MatchCardProps> = ({ match, onPress }) => {
  return (
    <TouchableOpacity style={styles.container} onPress={onPress}>
      <Text style={styles.title}>{match.title}</Text>
    </TouchableOpacity>
  );
};

// =====
// STYLES
// =====
const styles = StyleSheet.create({
  container: {
    padding: 16,
    backgroundColor: '#fff',
    borderRadius: 8,
  },
  title: {
    fontSize: 18,
    fontWeight: 'bold',
  },
});
```

11.2 Git Workflow

```
# Branch isimleri
feature/player-rating-system
bugfix/match-registration-error
hotfix/payment-calculation
release/v1.0.0

# Commit mesajlari (Conventional Commits)
feat: Add player rating screen
fix: Fix match registration bug
docs: Update README
style: Format code with prettier
refactor: Refactor MatchService
test: Add unit tests for RatingService
chore: Update dependencies
perf: Optimize standings calculation
```

12. Testing

12.1 Unit Tests (Jest)

```
# Tüm testleri çalıştır
npm test

# Watch modunda
npm test -- --watch

# Coverage raporu
npm test -- --coverage
```

Örnek Test:

```
// __tests__/utils/formatDate.test.ts
import { formatDate } from '@/utils/date';

describe('formatDate', () => {
  it('should format date correctly', () => {
    const date = new Date('2025-10-15T19:00:00');
    expect(formatDate(date)).toBe('15 Ekim 2025');
  });

  it('should handle invalid date', () => {
    expect(formatDate(null)).toBe('-');
  });
});
```

12.2 Component Tests

```
// __tests__/components/MatchCard.test.tsx
import React from 'react';
import { render, fireEvent } from '@testing-library/react-native';
import { MatchCard } from '@/components/match/MatchCard';

describe('MatchCard', () => {
  it('should render match details', () => {
    const { getByText } = render(<MatchCard match={mockMatch} />);
    expect(getByText('Salı Maçı')).toBeInTheDocument();
  });

  it('should call onPress when pressed', () => {
    const onPress = jest.fn();
    const { getByTestId } = render(
      <MatchCard match={mockMatch} onPress={onPress} />
    );
    const card = getByTestId('match-card');
    fireEvent.press(card);
    expect(onPress).toHaveBeenCalled();
  });
});
```

```
<MatchCard match={mockMatch} onPress={onPress} testID="match-card" />
);

fireEvent.press(getByTestId('match-card'));
expect(onPress).toHaveBeenCalledTimes(1);
});
});
```

13. Deployment

13.1 Android Build

```
# Debug APK
npm run android

# Release APK
cd android
./gradlew assembleRelease

# APK yolu:
# android/app/build/outputs/apk/release/app-release.apk

# Bundle (AAB) oluştur (Google Play için)
./gradlew bundleRelease

# AAB yolu:
# android/app/build/outputs/bundle/release/app-release.aab
```

13.2 iOS Build

```
# Debug build
npm run ios

# Release build (Xcode ile)
# 1. Xcode'u aç
# 2. Product → Archive
# 3. Distribute App → App Store Connect
```

13.3 Firebase Deployment

```
# Firestore rules
firebase deploy --only firestore:rules

# Firestore indexes
firebase deploy --only firestore:indexes
```

```
# Cloud Functions
firebase deploy --only functions

# Tümü
firebase deploy
```

14. Push Notifications (FCM)

14.1 FCM Setup

```
// src/services/NotificationService.ts
import messaging from '@react-native-firebase/messaging';

export class NotificationService {
    static async requestPermission(): Promise<boolean> {
        const authStatus = await messaging().requestPermission();
        return authStatus === messaging.AuthorizationStatus.AUTHORIZED;
    }

    static async getToken(): Promise<string> {
        return await messaging().getToken();
    }

    static setupListeners() {
        // Foreground
        messaging().onMessage(async (remoteMessage) => {
            console.log('Foreground notification:', remoteMessage);
        });

        // Background
        messaging().setBackgroundMessageHandler(async (remoteMessage) => {
            console.log('Background notification:', remoteMessage);
        });

        // Notification tıklama
        messaging().onNotificationOpenedApp((remoteMessage) => {
            console.log('Notification opened app:', remoteMessage);
        });
    }
}
```

15. State Management (Redux Toolkit)

15.1 Store Setup

```
// src/store/store.ts
import { configureStore } from '@reduxjs/toolkit';
import authReducer from './slices/authSlice';
import leagueReducer from './slices/leagueSlice';
import matchReducer from './slices/matchSlice';

export const store = configureStore({
  reducer: {
    auth: authReducer,
    league: leagueReducer,
    match: matchReducer,
  },
});

export type RootState = ReturnType<typeof store.getState>;
export type AppDispatch = typeof store.dispatch;
```

15.2 Slice Örneği

```
// src/store/slices/authSlice.ts
import { createSlice, createAsyncThunk } from '@reduxjs/toolkit';

export const login = createAsyncThunk(
  'auth/login',
  async ({ email, password }) => {
    return await AuthService.login(email, password);
  }
);

const authSlice = createSlice({
  name: 'auth',
  initialState: {
    user: null,
    loading: false,
    error: null,
  },
  reducers: {
    logout: (state) => {
      state.user = null;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(login.pending, (state) => {
        state.loading = true;
      })
      .addCase(login.fulfilled, (state, action) => {
        state.user = action.payload;
        state.loading = false;
      })
  }
});
```

```
.addCase(login.rejected, (state, action) => {
  state.error = action.error.message;
  state.loading = false;
});
},
});

export default authSlice.reducer;
```

16. Güvenlik

16.1 Firestore Security Rules

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    function isAuthenticated() {
      return request.auth != null;
    }

    function isOwner(userId) {
      return isAuthenticated() && request.auth.uid == userId;
    }

    // Users
    match /users/{userId} {
      allow read: if isAuthenticated();
      allow write: if isOwner(userId);
    }

    // Leagues
    match /leagues/{leagueId} {
      allow read: if isAuthenticated();
      allow create: if isAuthenticated();
      allow update, delete: if isLeagueAdmin(leagueId);
    }

    // Matches
    match /matches/{matchId} {
      allow read: if isAuthenticated();
      allow create: if isAuthenticated();
      allow update: if isMatchOrganizer(matchId);
    }
  }
}
```

17. Performans Optimizasyonu

17.1 React Native Optimizasyon

```
// useMemo ve useCallback kullan
const memoizedValue = useMemo(() => calculateExpensiveValue(data), [data]);
const memoizedCallback = useCallback(() => handlePress(), []);

// FlatList optimize et
<FlatList
  data={items}
  renderItem={renderItem}
  keyExtractor={(item) => item.id}
  initialNumToRender={10}
  maxToRenderPerBatch={10}
  windowSize={5}
  removeClippedSubviews={true}
/>

// React.memo kullan
export const MatchCard = React.memo(({ match }) => {
  return <View>...</View>;
});
```

18. Sık Kullanılan Komutlar

```
# Development
npm run android          # Android'de çalıştır
npm run ios               # iOS'ta çalıştır
npm start                 # Metro bundler başlat
npm start -- --reset-cache # Cache temizle

# Testing
npm test                  # Testleri çalıştır
npm test -- --coverage    # Coverage raporu

# Build
cd android && ./gradlew assembleRelease # Android release
cd ios && pod install                      # iOS pod install

# Firebase
firebase deploy --only firestore:rules   # Rules deploy
firebase deploy --only firestore:indexes # Indexes deploy

# Git
git checkout -b feature/new-feature      # Yeni branch
git commit -m "feat: Add new feature"    # Commit
git push origin feature/new-feature       # Push

# Dependency
npm install package-name                # Paket ekle
```

```
npm update                                # Paketleri güncelle
cd ios && pod install && cd ..          # iOS pod güncelle
```

19. Troubleshooting

19.1 Sık Karşılaşılan Hatalar

Metro Bundler Hatası

```
# Çözüm:
npm start -- --reset-cache
watchman watch-del-all
rm -rf node_modules && npm install
```

Android Build Hatası

```
# Çözüm:
cd android
./gradlew clean
cd ..
npm run android
```

iOS Build Hatası

```
# Çözüm:
cd ios
rm -rf Pods Podfile.lock
pod install --repo-update
cd ..
npm run ios
```

Firebase Connection Hatası

```
# Android: android/app/google-services.json kontrol et
# iOS: ios/GoogleService-Info.plist kontrol et
# .env dosyasındaki Firebase credentials'ı kontrol et
```

20. Lisans & Katkı

20.1 Lisans

MIT License

Copyright (c) 2025 Sport Management App

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction...

20.2 Katkıda Bulunma

1. Fork the repository
 2. Create your feature branch (`git checkout -b feature/amazing-feature`)
 3. Commit your changes (`git commit -m 'feat: Add amazing feature'`)
 4. Push to the branch (`git push origin feature/amazing-feature`)
 5. Open a Pull Request
-

21. İletişim

- **Email:** support@sportmanagement.app
 - **Website:** <https://sportmanagement.app>
 - **GitHub:** <https://github.com/sportmanagement/app>
 - **Discord:** <https://discord.gg/sportmanagement>
-

Sonuç

Bu dokümantasyon, **React Native** ile geliştirilen **Spor Yönetim Uygulaması**'nın kapsamlı bir rehberidir.

Kapsanan Konular:

-  66 Ekran detaylı tanımı
-  Mimari yapı ve klasör organizasyonu
-  24 Collection veritabanı şeması
-  22+ API class ve metodları
-  10+ Servis katmanı
-  Navigation akışı (React Navigation)
-  State management (Redux Toolkit)
-  Push bildirimler (FCM)
-  Güvenlik kuralları (Firestore Rules)
-  Test stratejisi (Jest, Detox)
-  Deployment rehberi (Android/iOS)
-  5 Detaylı kullanıcı senaryosu

Başlangıç Adımları:

```
# 1. Projeyi clone'la  
git clone https://github.com/your-repo/sport-management-app.git  
  
# 2. Bağımlılıkları yükle  
cd sport-management-app  
npm install  
  
# 3. Firebase'i yapılandırır  
cp .env.example .env  
# .env dosyasını düzenle  
  
# 4. iOS için pod install (macOS)  
cd ios && pod install && cd ..  
  
# 5. Uygulamayı başlat  
npm run android # veya npm run ios
```

Başarılı! 🏆 🎉 ⚽

Dokümantasyon Versiyonu: 1.0

Son Güncelleme: Ocak 2025

Yazar: Development Team