🏗️ ARCHITETTURA TECNICA KORA - DEVELOPER HANDOFF

1. TECH STACK OVERVIEW

1.1 Frontend Stack

```javascript
// Web App (Progressive Web App)
{
  "framework": "Next.js 14+ (App Router)",
  "ui_library": "React 18+",
  "styling": "Tailwind CSS 3.4+",
  "state_management": "Zustand",
  "forms": "React Hook Form + Zod",
  "api_client": "TanStack Query (React Query)",
  "animations": "Framer Motion",
  "pwa": "next-pwa",
  "typescript": "5.0+"
}

// Mobile App (Future - React Native)
{
  "framework": "React Native 0.73+",
  "navigation": "React Navigation 6",
  "state": "Zustand",
  "ui": "NativeWind (Tailwind for RN)",
  "expo": "SDK 50+"
}
```

1.2 Backend Stack

```javascript
```

```
{
  "database": "Supabase (PostgreSQL)",
  "orm": "Prisma ORM",
  "auth": "Supabase Auth",
  "storage": "Supabase Storage",
  "realtime": "Supabase Realtime",
  "edge_functions": "Supabase Edge Functions",
  "ai": "OpenAI GPT-4o mini (fallback: Claude 3 Haiku)",
  "email": "Resend",
  "monitoring": "Sentry",
  "deployment": "Vercel (frontend) + Supabase (backend)"
}
```

---

2. PROJECT STRUCTURE

2.1 Simplified Structure (Vercel + Supabase)

bash

```
kora/
├── app/                    # Next.js App Router
│   ├── (auth)/             # Auth pages
│   ├── (dashboard)/        # Protected pages
│   ├── api/                # API routes (if needed)
│   └── layout.tsx          # Root layout
│
├── components/             # React components
│   ├── ui/                 # Base UI components
│   ├── features/           # Feature components
│   └── layouts/            # Layout components
│
├── lib/                    # Utilities
│   ├── supabase/           # Supabase client
│   ├── openai/             # GPT-4o mini client
│   └── utils/              # Helpers
│
├── hooks/                  # Custom React hooks
├── stores/                 # Zustand stores
├── types/                  # TypeScript types
├── prisma/                 # Database schema
│   └── schema.prisma
│
├── public/                 # Static assets
├── styles/                 # Global styles
│
├── .env.local              # Environment variables
├── next.config.js          # Next.js config
├── tailwind.config.js      # Tailwind config
└── package.json
```

---

3. DATABASE SCHEMA (PRISMA)

3.1 Core Models

```
prisma
```

```
// prisma/schema.prisma

generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

// User Management
model User {
  id               String    @id @default(cuid())
  email            String    @unique
  password         String
  firstName        String
  lastName         String
  role             UserRole  @default(COACH)
  subscriptionTier Tier      @default(FREE)
  emailVerified    Boolean   @default(false)
  createdAt        DateTime  @default(now())
  updatedAt        DateTime  @updatedAt

  // Relations
  teams            Team[]
  conversations    Conversation[]
  subscription     Subscription?
}

enum UserRole {
  COACH
  ASSISTANT
  ADMIN
}

enum Tier {
  FREE
  LEVEL1
  PREMIUM
}

// Team Management
model Team {
  id          String    @id @default(cuid())
  name        String
  sport      Sport
  category   String   // "U14", "Prima Squadra", etc
  season     String   // "2024/2025"
  homeField  String?
  colors     Json?    // { primary: "#color", secondary: "#color" }
  logo       String?  // S3 URL
  createdAt  DateTime @default(now())
  updatedAt  DateTime @updatedAt

  // Relations
  coachId    String
  coach      User     @relation(fields: [coachId], references: [id])
  players    Player[]
  trainings  Training[]
  matches    Match[]

  @@index([coachId])
}

enum Sport {
  SOCCER
  BASKETBALL
  VOLLEYBALL
  RUGBY
  TENNIS
  OTHER
}

// Player Management
model Player {
  id            String   @id @default(cuid())
  firstName     String
  lastName      String
  dateOfBirth   DateTime
  position      String
  jerseyNumber  Int?
  photo         String?  // S3 URL
  medicalNotes  String?
  parentEmail   String?
  parentPhone   String?
  isActive      Boolean  @default(true)
  createdAt     DateTime @default(now())
  updatedAt     DateTime @updatedAt

  // Relations
  teamId        String
  team          Team     @relation(fields: [teamId], references: [id])
  attendances    Attendance[]
  stats          PlayerStats[]

  @@index([teamId])
}

// Training Management
model Training {
  id             String    @id @default(cuid())
  date           DateTime
  duration       Int       // minutes
  location       String
  type           TrainingType
  focus          String[]  // ["tecnica", "tattica", "fisico"]
  plan           Json      // Structured training plan
  notes          String?
  completed      Boolean   @default(false)
  createdAt      DateTime  @default(now())
  updatedAt      DateTime  @updatedAt

  // Relations
  teamId         String
  team           Team      @relation(fields: [teamId], references: [id])
  attendances    Attendance[]

  @@index([teamId, date])
}

enum TrainingType {
  REGULAR
  MATCH_PREP
  RECOVERY
  TACTICAL
  TECHNICAL
  PHYSICAL
}

// Attendance Tracking
model Attendance {
  id             String    @id @default(cuid())
  status         AttendanceStatus
  arrivedLate    Boolean   @default(false)
  leftEarly      Boolean   @default(false)
  notes          String?

  // Relations
  playerId       String
  player         Player    @relation(fields: [playerId], references: [id])
  trainingId     String
  training       Training  @relation(fields: [trainingId], references: [id])

  @@unique([playerId, trainingId])
  @@index([trainingId])
}

enum AttendanceStatus {
  PRESENT
  ABSENT_JUSTIFIED
  ABSENT_UNJUSTIFIED
  INJURED
}

// AI Integration
model Conversation {
  id            String    @id @default(cuid())
  title         String?
  context       Json      // Team info, recent performance, etc
  messages      Json      // Array of messages
  createdAt     DateTime  @default(now())
  updatedAt     DateTime  @updatedAt

  // Relations
  userId        String
  user          User      @relation(fields: [userId], references: [id])

  @@index([userId])
}

// Subscription Management
model Subscription {
  id            String    @id @default(cuid())
  stripeCustomerId String?
  stripePriceId   String?
  status        SubscriptionStatus
  currentPeriodEnd DateTime?
  cancelAtPeriodEnd Boolean @default(false)
  createdAt     DateTime  @default(now())
  updatedAt     DateTime  @updatedAt

  // Relations
  userId        String    @unique
  user          User      @relation(fields: [userId], references: [id])
}
```

```
enum SubscriptionStatus {
  ACTIVE
  CANCELED
  PAST_DUE
  UNPAID
  TRIALING
}
```

---

4. API IMPLEMENTATION WITH SUPABASE

4.1 Supabase Client Setup

typescript

```typescript
// lib/supabase/server.ts
import { createServerClient, type CookieOptions } from '@supabase/ssr'
import { cookies } from 'next/headers'

export async function createClient() {
  const cookieStore = cookies()

  return createServerClient(
    process.env.NEXT_PUBLIC_SUPABASE_URL!,
    process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!,
    {
      cookies: {
        get(name: string) {
          return cookieStore.get(name)?.value
        },
        set(name: string, value: string, options: CookieOptions) {
          cookieStore.set({ name, value, ...options })
        },
        remove(name: string, options: CookieOptions) {
          cookieStore.delete({ name, ...options })
        },
      },
    }
  )
}
```

4.2 API Routes with Next.js App Router

typescript

```typescript
// app/api/teams/route.ts
import { createClient } from '@/lib/supabase/server'
import { prisma } from '@/lib/prisma'

export async function GET() {
  const supabase = await createClient()
  const { data: { user } } = await supabase.auth.getUser()

  if (!user) {
    return Response.json({ error: 'Unauthorized' }, { status: 401 })
  }

  const teams = await prisma.team.findMany({
    where: { coachId: user.id },
    include: { players: true }
  })

  return Response.json({ teams })
}

export async function POST(request: Request) {
  const supabase = await createClient()
  const { data: { user } } = await supabase.auth.getUser()

  if (!user) {
    return Response.json({ error: 'Unauthorized' }, { status: 401 })
  }

  const body = await request.json()

  const team = await prisma.team.create({
    data: {
      ...body,
      coachId: user.id
    }
  })

  return Response.json({ team }, { status: 201 })
}
```

4.3 Server Actions (Alternative to API Routes)

typescript

```
// app/actions/teams.ts
'use server'

import { createClient } from '@/lib/supabase/server'
import { prisma } from '@/lib/prisma'
import { revalidatePath } from 'next/cache'

export async function createTeam(formData: FormData) {
  const supabase = await createClient()
  const { data: { user } } = await supabase.auth.getUser()

  if (!user) {
    return { error: 'Unauthorized' }
  }

  const team = await prisma.team.create({
    data: {
      name: formData.get('name') as string,
      sport: formData.get('sport') as Sport,
      category: formData.get('category') as string,
      season: formData.get('season') as string,
      coachId: user.id
    }
  })

  revalidatePath('/dashboard/teams')
  return { team }
}
```

5. FRONTEND IMPLEMENTATION GUIDE

5.1 Component Structure

typescript

```tsx
// components/ui/Button.tsx
interface ButtonProps {
  variant?: 'primary' | 'secondary' | 'ghost';
  size?: 'sm' | 'md' | 'lg';
  loading?: boolean;
  disabled?: boolean;
  fullWidth?: boolean;
  children: React.ReactNode;
  onClick?: () => void;
}

export function Button({
  variant = 'primary',
  size = 'md',
  loading = false,
  disabled = false,
  fullWidth = false,
  children,
  onClick
}: ButtonProps) {
  const baseClasses = "inline-flex items-center justify-center font-medium transition-colors focus:outline-none fo

  const variants = {
    primary: "bg-blue-600 text-white hover:bg-blue-700 focus:ring-blue-500",
    secondary: "bg-gray-200 text-gray-900 hover:bg-gray-300 focus:ring-gray-500",
    ghost: "bg-transparent hover:bg-gray-100 focus:ring-gray-500"
  };

  const sizes = {
    sm: "px-3 py-1.5 text-sm rounded-md",
    md: "px-4 py-2 text-base rounded-lg",
    lg: "px-6 py-3 text-lg rounded-lg"
  };

  return (
    <button
      onClick={onClick}
      disabled={disabled || loading}
      className={cn(
        baseClasses,
        variants[variant],
        sizes[size],
        fullWidth && "w-full",
        className
      )}
    >
```

```
      {loading && <Spinner className="mr-2" />}
      {children}
    </button>
  );
}
```

5.2 Page Implementation Example

```
typescript
```

```
// app/dashboard/page.tsx
export default async function DashboardPage() {
  return (
    <DashboardLayout>
      <div className="space-y-6">
        {/* Header */}
        <div className="flex justify-between items-center">
          <h1 className="text-2xl font-bold text-gray-900">
            Dashboard
          </h1>
          <Button onClick={() => router.push('/trainings/new')}>
            Nuovo Allenamento
          </Button>
        </div>

        {/* Stats Grid */}
        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4">
          <StatCard
            title="Prossimo Allenamento"
            value="Oggi, 18:30"
            icon={<CalendarIcon />}
            action={{ label: "Dettagli", href: "/trainings/next" }}
          />
          <StatCard
            title="Presenze Ultima"
            value="85%"
            subtitle="17/20 giocatori"
            icon={<UsersIcon />}
          />
          {/* More stats... */}
        </div>

        {/* AI Coach CTA */}
        <AiCoachCard />

        {/* Recent Activity */}
        <RecentActivityFeed />
      </div>
    </DashboardLayout>
  );
}
```

5.3 State Management Pattern

```
typescript
```

```
// stores/teamStore.ts (Zustand)
interface TeamStore {
  teams: Team[];
  currentTeam: Team | null;
  loading: boolean;
  error: string | null;

  // Actions
  fetchTeams: () => Promise<void>;
  setCurrentTeam: (teamId: string) => void;
  createTeam: (data: CreateTeamDTO) => Promise<Team>;
  updateTeam: (id: string, data: UpdateTeamDTO) => Promise<void>;
}

export const useTeamStore = create<TeamStore>((set, get) => ({
  teams: [],
  currentTeam: null,
  loading: false,
  error: null,

  fetchTeams: async () => {
    set({ loading: true, error: null });
    try {
      const teams = await api.teams.list();
      set({ teams, loading: false });
    } catch (error) {
      set({ error: error.message, loading: false });
    }
  },

  setCurrentTeam: (teamId) => {
    const team = get().teams.find(t => t.id === teamId);
    set({ currentTeam: team });
  },

  // ... other actions
}));
```

---

6. MOBILE-FIRST RESPONSIVE DESIGN

6.1 Tailwind Configuration

javascript

```
// tailwind.config.js
module.exports = {
  content: [
    './app/**/*.{js,ts,jsx,tsx,mdx}',
    './components/**/*.{js,ts,jsx,tsx,mdx}',
  ],
  theme: {
    extend: {
      colors: {
        primary: {
          50: '#eff6ff',
          500: '#3b82f6',
          600: '#2563eb',
          700: '#1d4ed8',
        },
        gray: {
          50: '#f9fafb',
          900: '#111827',
        }
      },
      fontFamily: {
        sans: ['Inter', 'system-ui', 'sans-serif'],
      },
      animation: {
        'slide-up': 'slideUp 0.3s ease-out',
        'fade-in': 'fadeIn 0.2s ease-out',
      }
    },
  },
  plugins: [
    require('@tailwindcss/forms'),
    require('@tailwindcss/typography'),
  ],
};
```

6.2 Responsive Component Example

```
typescript
```

```tsx
// components/PlayerCard.tsx
export function PlayerCard({ player }: { player: Player }) {
 return (
  <div className="bg-white rounded-lg shadow-sm border border-gray-200 p-4
          hover:shadow-md transition-shadow cursor-pointer
          // Responsive layout
          flex flex-col sm:flex-row sm:items-center sm:justify-between">

   {/* Player Info */}
   <div className="flex items-center space-x-3 mb-3 sm:mb-0">
    <div className="w-12 h-12 rounded-full bg-gray-200 flex items-center justify-center">
     {player.photo ? (
      <img src={player.photo} className="w-full h-full rounded-full object-cover" />
     ) : (
      <span className="text-lg font-medium text-gray-600">
       {player.firstName[0]}{player.lastName[0]}
      </span>
     )}
    </div>

    <div>
     <h3 className="font-medium text-gray-900">
      {player.firstName} {player.lastName}
     </h3>
     <p className="text-sm text-gray-500">
      #{player.jerseyNumber} • {player.position}
     </p>
    </div>
   </div>

   {/* Stats - Hidden on mobile, shown on tablet+ */}
   <div className="hidden sm:flex items-center space-x-6">
    <Stat label="Presenze" value={`${player.attendanceRate}%`} />
    <Stat label="Goal" value={player.goals} />
   </div>

   {/* Actions */}
   <div className="flex justify-end space-x-2">
    <Button variant="ghost" size="sm">
     <EditIcon className="w-4 h-4" />
    </Button>
    <Button variant="ghost" size="sm">
     <StatsIcon className="w-4 h-4" />
    </Button>
   </div>
  </div>
  );
}
```

7. PWA CONFIGURATION

7.1 Next.js PWA Setup

javascript

```
// next.config.js
const withPWA = require('next-pwa')({
  dest: 'public',
  register: true,
  skipWaiting: true,
  disable: process.env.NODE_ENV === 'development',
  runtimeCaching: [
    {
      urlPattern: /^https:\/\/api\.kora\.app\/api/,
      handler: 'NetworkFirst',
      options: {
        cacheName: 'api-cache',
        expiration: {
          maxEntries: 50,
          maxAgeSeconds: 60 * 5 // 5 minutes
        }
      }
    }
  ]
});

module.exports = withPWA({
  reactStrictMode: true,
  images: {
    domains: ['kora-assets.s3.amazonaws.com'],
  },
});

// public/manifest.json
{
  "name": "Kora - AI Coach",
  "short_name": "Kora",
  "description": "L'assistente AI per ogni allenatore",
  "start_url": "/",
  "display": "standalone",
  "theme_color": "#2563eb",
  "background_color": "#ffffff",
  "icons": [
    {
      "src": "/icon-192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/icon-512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

---

8. AI INTEGRATION IMPLEMENTATION

8.1 GPT-4o Mini Service (Cost-Optimized)

```
typescript
```

```
// services/ai/openai.service.ts
import OpenAI from 'openai';

export class AIService {
 private openai: OpenAI;
 private model = 'gpt-4o-mini'; // $0.15/1M input, $0.60/1M output

 constructor() {
  this.openai = new OpenAI({
   apiKey: process.env.OPENAI_API_KEY,
  });
 }

 async generateTrainingPlan(params: {
  team: Team;
  focus: string[];
  duration: number;
  recentTrainings?: Training[];
 }): Promise<TrainingPlan> {
  const prompt = this.buildTrainingPrompt(params);

  try {
   const completion = await this.openai.chat.completions.create({
    model: this.model,
    messages: [
     {
      role: 'system',
      content: `Sei un assistente AI esperto di allenamento ${params.team.sport}.
           Crea piani di allenamento dettagliati per la categoria ${params.team.category}.
           Rispondi SEMPRE in formato JSON valido senza markdown.`
     },
     {
      role: 'user',
      content: prompt
     }
    ],
    temperature: 0.7,
    max_tokens: 1500,
    response_format: { type: "json_object" }
   });

   return JSON.parse(completion.choices[0].message.content!);
  } catch (error) {
   // Fallback to Claude 3 Haiku if needed
   if (this.shouldFallbackToClaude(error)) {
    return this.generateWithClaude(params);
```

```
      }
      throw error;
    }
  }

  async chatWithCoach(params: {
    message: string;
    conversationId?: string;
    context: CoachContext;
  }): Promise<ChatResponse> {
    const messages = await this.buildConversationHistory(params);

    const completion = await this.openai.chat.completions.create({
      model: this.model,
      messages: [
        {
          role: 'system',
          content: this.getCoachSystemPrompt(params.context)
        },
        ...messages
      ],
      temperature: 0.8,
      max_tokens: 800
    });

    const response = completion.choices[0].message.content!;

    // Log usage for cost tracking
    await this.logUsage({
      model: this.model,
      promptTokens: completion.usage?.prompt_tokens || 0,
      completionTokens: completion.usage?.completion_tokens || 0,
      cost: this.calculateCost(completion.usage!)
    });

    return {
      message: response,
      suggestions: this.extractSuggestions(response)
    };
  }

  private calculateCost(usage: OpenAI.CompletionUsage): number {
    const inputCost = (usage.prompt_tokens / 1_000_000) * 0.15;
    const outputCost = (usage.completion_tokens / 1_000_000) * 0.60;
    return inputCost + outputCost;
  }
  // Claude 3 Haiku Fallback ($0.25/1M input, $1.25/1M output)
  private async generateWithClaude(params: any): Promise<TrainingPlan> {
    // Implementation for Claude fallback
    // Only used when GPT-4o mini quality is insufficient
  }
}
```

8.2 Caching Strategy for AI Responses

```typescript
// services/cache/ai-cache.service.ts
export class AICacheService {
  private redis: Redis;

  async getCachedResponse(key: string): Promise<any | null> {
    const cached = await this.redis.get(key);
    return cached ? JSON.parse(cached) : null;
  }

  async cacheResponse(key: string, data: any, ttl: number = 3600) {
    await this.redis.setex(key, ttl, JSON.stringify(data));
  }

  generateCacheKey(params: any): string {
    // Create deterministic cache key from parameters
    const normalized = {
      team: params.team.id,
      focus: params.focus.sort(),
      duration: params.duration,
      sport: params.team.sport,
      category: params.team.category
    };

    return `ai:training:${crypto
      .createHash('md5')
      .update(JSON.stringify(normalized))
      .digest('hex')}`;
  }
}
```

---

9. DEPLOYMENT & DEVOPS

9.1 Vercel Deployment Configuration

```javascript
```

```
// vercel.json
{
  "framework": "nextjs",
  "buildCommand": "prisma generate && next build",
  "env": {
    "DATABASE_URL": "@supabase_database_url",
    "DIRECT_URL": "@supabase_direct_url",
    "NEXT_PUBLIC_SUPABASE_URL": "@supabase_url",
    "NEXT_PUBLIC_SUPABASE_ANON_KEY": "@supabase_anon_key"
  },
  "functions": {
    "app/api/ai/chat/route.ts": {
      "maxDuration": 30
    }
  }
}

// next.config.js
/** @type {import('next').NextConfig} */
const nextConfig = {
  images: {
    remotePatterns: [
      {
        protocol: 'https',
        hostname: '**.supabase.co',
        pathname: '/storage/v1/object/public/**',
      },
    ],
  },
}

module.exports = nextConfig
```

9.2 Environment Variables

```bash
```

```
# .env.local

# Supabase
NEXT_PUBLIC_SUPABASE_URL=https://[PROJECT_ID].supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=eyJ...
SUPABASE_SERVICE_ROLE_KEY=eyJ...

# Database (Prisma)
DATABASE_URL=postgres://postgres.[PROJECT_ID]:6543/postgres?pgbouncer=true
DIRECT_URL=postgres://postgres.[PROJECT_ID]:5432/postgres

# AI Services
OPENAI_API_KEY=sk-...
ANTHROPIC_API_KEY=sk-ant-... # Fallback option

# Email
RESEND_API_KEY=re_...

# Monitoring
NEXT_PUBLIC_SENTRY_DSN=https://...
SENTRY_AUTH_TOKEN=...

# Analytics (Vercel)
NEXT_PUBLIC_ANALYTICS_ID=...
```

9.3 CI/CD Pipeline (GitHub Actions + Vercel)

yaml

```yaml
# .github/workflows/preview.yml
name: Preview Deployment

on:
  pull_request:
    types: [opened, synchronize]

jobs:
  deploy-preview:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '20'
          cache: 'npm'

      - name: Install dependencies
        run: npm ci

      - name: Run tests
        run: npm test

      - name: Run type check
        run: npm run type-check

      - name: Deploy to Vercel
        run: |
          npm i -g vercel
          vercel pull --yes --environment=preview --token=${{ secrets.VERCEL_TOKEN }}
          vercel build --token=${{ secrets.VERCEL_TOKEN }}
          vercel deploy --prebuilt --token=${{ secrets.VERCEL_TOKEN }}
```

---

10. SECURITY IMPLEMENTATION

10.1 Authentication Middleware

```
typescript
```

```
// middleware/auth.middleware.ts
export const authenticateToken = async (
  req: Request,
  res: Response,
  next: NextFunction
) => {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];

  if (!token) {
    return res.status(401).json({ error: 'Access token required' });
  }

  try {
    const payload = jwt.verify(token, process.env.JWT_SECRET) as JWTPayload;
    const user = await prisma.user.findUnique({
      where: { id: payload.userId },
      select: { id: true, email: true, tier: true }
    });

    if (!user) {
      return res.status(401).json({ error: 'User not found' });
    }

    req.user = user;
    next();
  } catch (error) {
    return res.status(403).json({ error: 'Invalid token' });
  }
};

// Rate limiting for AI endpoints
export const aiRateLimiter = rateLimit({
  windowMs: 60 * 1000, // 1 minute
  max: (req) => {
    // Different limits based on subscription tier
    const tierLimits = {
      FREE: 5,
      LEVEL1: 20,
      PREMIUM: 100
    };
    return tierLimits[req.user?.tier || 'FREE'];
  },
  message: 'Too many AI requests, please try again later'
});
```

10.2 Input Validation

typescript

```typescript
// validation/team.validation.ts
import { z } from 'zod';

export const createTeamSchema = z.object({
  name: z.string().min(2).max(50),
  sport: z.enum(['SOCCER', 'BASKETBALL', 'VOLLEYBALL', 'RUGBY', 'TENNIS', 'OTHER']),
  category: z.string().min(2).max(30),
  season: z.string().regex(/^\d{4}\/\d{4}$/),
  homeField: z.string().optional(),
  colors: z.object({
    primary: z.string().regex(/^#[0-9A-F]{6}$/i),
    secondary: z.string().regex(/^#[0-9A-F]{6}$/i)
  }).optional()
});

export const createPlayerSchema = z.object({
  firstName: z.string().min(2).max(30),
  lastName: z.string().min(2).max(30),
  dateOfBirth: z.string().datetime(),
  position: z.string().min(2).max(30),
  jerseyNumber: z.number().int().min(1).max(99).optional(),
  parentEmail: z.string().email().optional(),
  parentPhone: z.string().optional()
});
```

---

11. DEVELOPER SETUP INSTRUCTIONS

11.1 Initial Setup Commands

bash

```
# Clone repository
git clone https://github.com/your-org/kora.git
cd kora

# Install dependencies
npm install

# Setup database
docker-compose up -d postgres redis
npx prisma migrate dev
npx prisma db seed

# Setup environment
cp .env.example .env.local
# Edit .env.local with your values

# Run development
npm run dev

# Access:
# - Web: http://localhost:3000
# - API: http://localhost:3001
# - Prisma Studio: http://localhost:5555
```

11.2 Development Workflow

bash

```bash
# Create new feature branch
git checkout -b feature/player-stats

# Run development server
npm run dev

# Run Supabase locally (optional)
npx supabase start

# Check TypeScript
npm run type-check

# Run tests
npm run test

# Check linting
npm run lint

# Format code
npm run format

# Generate Prisma client after schema changes
npx prisma generate

# Push schema to Supabase
npx prisma db push

# Create migration
npx prisma migrate dev --name add_player_stats

# Deploy to Vercel (automatic on push to main)
git push origin main
```

---

12. CLAUDE CODE SPECIFIC INSTRUCTIONS

12.1 Project Setup Prompt for Claude Code

```
markdown
```

```
Create a new Next.js 14 project with TypeScript for Kora - an AI-powered sports team management app.

Tech stack:
- Next.js 14 with App Router
- TypeScript
- Tailwind CSS
- Prisma with PostgreSQL
- Zustand for state management
- React Hook Form + Zod for forms
- TanStack Query for API calls

Initialize with:
1. Monorepo structure using Turborepo
2. Separate packages for web app and API
3. Shared types package
4. Docker compose for local development
5. ESLint and Prettier configured
```

12.2 Component Generation Pattern

```markdown
Generate a responsive PlayerCard component that:
- Shows player photo, name, jersey number, and position
- Displays attendance percentage and key stats
- Has edit and view stats action buttons
- Is fully responsive (stack on mobile, horizontal on desktop)
- Uses Tailwind classes for styling
- Includes TypeScript interfaces
- Handles loading and error states
- Is accessible (ARIA labels, keyboard navigation)
```

12.3 API Endpoint Generation

```markdown
```

```
Create a REST API endpoint for managing team trainings that:
- Uses Express with TypeScript
- Implements CRUD operations
- Uses Prisma for database queries
- Includes proper error handling
- Has request validation using Zod
- Implements authentication middleware
- Returns consistent response format
- Includes rate limiting for AI features
- Has comprehensive error messages
```

CONCLUSION

Questa architettura tecnica fornisce:

1. **Scalabilità**: Monorepo structure con deployment indipendente
2. **Type Safety**: TypeScript end-to-end con Prisma
3. **Performance**: Caching strategico e PWA
4. **Security**: JWT auth, rate limiting, input validation
5. **Developer Experience**: Hot reload, type generation, testing

Il sistema è progettato per essere sviluppato iterativamente con Claude Code, con chiare convenzioni e pattern da seguire.