Modulo de Running

FitTrack

Documentacion Tecnica Completa

Este documento proporciona una guia tecnica exhaustiva del modulo de running de FitTrack, incluyendo arquitectura, implementacion, base de datos, componentes React, Server Actions y ejemplos practicos de uso.

Version 1.0

14 de octubre de 2025

Índice

1 Introduccion al Modulo de Running

El modulo de running es un componente fundamental de FitTrack que permite a los usuarios registrar, analizar y hacer seguimiento de sus sesiones de carrera. Proporciona herramientas completas para el monitoreo del progreso, calculo automatico de metricas y visualizacion de estadisticas detalladas.

1.1 Caracteristicas Principales

- **Registro de Sesiones**: Permite registrar sesiones de carrera con duracion, distancia y ritmo
- **Calculo Automatico de Ritmo**: Calcula automaticamente el ritmo por kilometro basado en duracion y distancia
- **Estadisticas Detalladas**: Proporciona metricas completas como total de sesiones, distancia acumulada, tiempo total, ritmo promedio y mejor ritmo
- **Historial de Sesiones**: Mantiene un registro completo de todas las sesiones de carrera
- **Graficos de Progreso**: Visualizacion de tendencias y progreso a lo largo del tiempo
- **Analisis de Rendimiento**: Comparacion de sesiones y identificacion de mejoras
- **Interfaz Intuitiva**: Diseño limpio y facil de usar para registro rapido

1.2 Arquitectura General

El modulo sigue una arquitectura de capas bien definida:

1. **Capa de Presentacion**: Componentes React con TypeScript
2. **Capa de Logica**: Server Actions de Next.js
3. **Capa de Datos**: Supabase PostgreSQL con RLS
4. **Capa de Servicios**: Calculos y utilidades de metricas

2 Estructura de Base de Datos

2.1 Diagrama de Relaciones

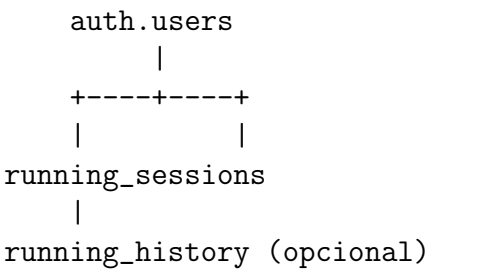

Figura 1: Diagrama de relaciones de las tablas del modulo de running

2.2 Tabla running_sessions

Esta tabla almacena las sesiones de carrera realizadas por los usuarios.

```sql
CREATE TABLE IF NOT EXISTS public.running_sessions (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE NOT NULL,
    duration_minutes INTEGER NOT NULL CHECK (duration_minutes > 0),
    distance_km NUMERIC(5,2) NOT NULL CHECK (distance_km > 0),
    pace_min_km NUMERIC(5,2),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Indices para optimizacion
CREATE INDEX IF NOT EXISTS idx_running_sessions_user_id ON public.
   running_sessions(user_id);
CREATE INDEX IF NOT EXISTS idx_running_sessions_created_at ON public.
   running_sessions(created_at);
CREATE INDEX IF NOT EXISTS idx_running_sessions_distance ON public.
   running_sessions(distance_km);
CREATE INDEX IF NOT EXISTS idx_running_sessions_pace ON public.
   running_sessions(pace_min_km);
```

Listing 1: Estructura completa de running$_s$*essions*

Descripcion de campos:

- `id`: Identificador unico UUID generado automaticamente
- `user_id`: Referencia al usuario propietario de la sesion
- `duration_minutes`: Duracion de la sesion en minutos (requerido, ¿0)
- `distance_km`: Distancia recorrida en kilometros (requerido, ¿0)
- `pace_min_km`: Ritmo en minutos por kilometro (opcional, calculado automaticamente)
- `created_at`: Timestamp de creacion automatico

2.3 Tabla running_history (Opcional)

Tabla adicional para historial detallado de sesiones de running.

```
CREATE TABLE IF NOT EXISTS running_history (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE NOT NULL,
    duration INTEGER NOT NULL CHECK (duration > 0),
    distance DECIMAL(6,2) NOT NULL CHECK (distance > 0),
    pace DECIMAL(4,2) NOT NULL CHECK (pace > 0),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Indices para optimizacion
CREATE INDEX IF NOT EXISTS idx_running_history_user_id ON
   running_history(user_id);
CREATE INDEX IF NOT EXISTS idx_running_history_created_at ON
   running_history(created_at);
```

Listing 2: Estructura de $running_history$

2.4 Politicas de Seguridad (RLS)

Todas las tablas implementan Row Level Security para garantizar que los usuarios solo accedan a sus propios datos.

```
-- Habilitar RLS
ALTER TABLE public.running_sessions ENABLE ROW LEVEL SECURITY;

-- Politicas para running_sessions
CREATE POLICY "Users can view own running sessions"
ON public.running_sessions
FOR SELECT USING (auth.uid() = user_id);

CREATE POLICY "Users can insert own running sessions"
ON public.running_sessions
FOR INSERT WITH CHECK (auth.uid() = user_id);

CREATE POLICY "Users can update own running sessions"
ON public.running_sessions
FOR UPDATE USING (auth.uid() = user_id);

CREATE POLICY "Users can delete own running sessions"
ON public.running_sessions
FOR DELETE USING (auth.uid() = user_id);
```

Listing 3: Politicas RLS para $running_sessions$

3 Server Actions - Logica de Negocio

Las Server Actions manejan toda la logica de negocio del modulo de running. Estan implementadas en TypeScript con validacion robusta y manejo de errores.

3.1 Archivo running-actions.ts

3.1.1 createRunningSession - Crear Sesion de Running

```
"use server"

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

export async function createRunningSession(prevState: any, formData:
   FormData) {
  // Extraer datos del formulario
  const duration_minutes = formData.get("duration_minutes")?.toString()
  const distance_km = formData.get("distance_km")?.toString()
  const pace_min_km = formData.get("pace_min_km")?.toString()

  // Validacion basica
  if (!duration_minutes || !distance_km) {
    return { error: "Duracion y distancia son requeridos" }
  }

  const durationNum = Number.parseInt(duration_minutes)
  const distanceNum = Number.parseFloat(distance_km)

  if (durationNum <= 0 || distanceNum <= 0) {
    return { error: "Duracion y distancia deben ser mayores a 0" }
  }

  // Calcular ritmo si no se proporciona
  let calculatedPace: number | null = null
  if (pace_min_km && pace_min_km.trim() !== "") {
    calculatedPace = Number.parseFloat(pace_min_km)
  } else {
    calculatedPace = durationNum / distanceNum
  }

  // Verificar autenticacion
  const supabase = await createClient()
  const {
    data: { user },
  } = await supabase.auth.getUser()

  if (!user) {
    return { error: "Usuario no autenticado" }
  }

  try {
    // Preparar datos para insercion
    const insertData = {
      user_id: user.id,
      duration_minutes: durationNum,
      distance_km: distanceNum,
      pace_min_km: calculatedPace,
    }

    // Insertar en base de datos
    const { error } = await supabase
      .from("running_sessions")
```

```
      .insert(insertData)

    if (error) {
      console.error("Database error:", error)
      return { error: "Error al guardar la sesion" }
    }

    // Revalidar cache
    revalidatePath("/running")
    return { success: true }
  } catch (error) {
    console.error("Error:", error)
    return { error: "Error al guardar la sesion" }
  }
}
```

Listing 4: Funcion createRunningSession completa

Caracteristicas importantes:

- **Validacion de entrada**: Verifica que duracion y distancia esten presentes y sean validos
- **Calculo automatico de ritmo**: Calcula el ritmo si no se proporciona (duracion / distancia)
- **Autenticacion**: Confirma que el usuario este autenticado
- **Sanitizacion**: Convierte y valida valores numericos
- **Manejo de errores**: Captura y reporta errores de base de datos
- **Revalidacion**: Actualiza el cache de Next.js

3.1.2 getRunningSessions - Obtener Sesiones

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

  if (!user) {
    return []
  }

  try {
    const { data, error } = await supabase
      .from("running_sessions")
      .select("*")
      .eq("user_id", user.id)
      .order("created_at", { ascending: false })

    if (error) {
      console.error("Database error:", error)
      return []
    }
```

```
22
23      return data || []
24    } catch (error) {
25      console.error("Error:", error)
26      return []
27    }
28  }
```

Listing 5: Funcion getRunningSessions completa

3.1.3 deleteRunningSession - Eliminar Sesion

```
1  export async function deleteRunningSession(sessionId: string) {
2    const supabase = await createClient()
3    const {
4      data: { user },
5    } = await supabase.auth.getUser()
6
7    if (!user) {
8      return { error: "Usuario no autenticado" }
9    }
10
11   try {
12     const { error } = await supabase
13       .from("running_sessions")
14       .delete()
15       .eq("id", sessionId)
16       .eq("user_id", user.id)
17
18     if (error) {
19       console.error("Database error:", error)
20       return { error: "Error al eliminar la sesion" }
21     }
22
23     revalidatePath("/running")
24     return { success: true }
25   } catch (error) {
26     console.error("Error:", error)
27     return { error: "Error al eliminar la sesion" }
28   }
29 }
```

Listing 6: Funcion deleteRunningSession completa

4 Componentes React

4.1 Pagina Principal - RunningPage

El componente principal que coordina toda la funcionalidad del modulo de running.

```
1 "use client"
2
3 import { useState } from "react"
4 import { MapPin, ArrowLeft } from "lucide-react"
5 import Link from "next/link"
```

```
import { Button } from "@/components/ui/button"
import RunningForm from "@/components/running/running-form"
import RunningList from "@/components/running/running-list"
import RunningStats from "@/components/running/running-stats"
import RunningCharts from "@/components/running/running-charts"

export default function RunningPage() {
  // Estados del componente
  const [refreshTrigger, setRefreshTrigger] = useState(0)
  const [viewMode, setViewMode] = useState<"sessions" | "charts">("sessions")

  // Handler para cuando se agrega una sesion
  const handleSessionAdded = () => {
    setRefreshTrigger((prev) => prev + 1)
  }

  return (
    <div className="container mx-auto px-4 py-8 max-w-4xl">
      {/* Header con navegacion */}
      <div className="mb-8">
        <div className="flex items-center gap-4 mb-4">
          <Button variant="outline" size="sm" asChild>
            <Link href="/">
              <ArrowLeft className="h-4 w-4 mr-2" />
              Volver
            </Link>
          </Button>
        </div>
        <div className="flex items-center gap-3 mb-2">
          <MapPin className="h-8 w-8 text-green-600" />
          <h1 className="text-3xl font-bold">Running</h1>
        </div>
        <p className="text-gray-600">Registra y analiza tus sesiones de carrera</p>
      </div>

      {/* Pestanas de navegacion */}
      <div className="flex gap-2 mb-6">
        <button
          onClick={() => setViewMode("sessions")}
          className={`px-4 py-2 rounded-lg font-medium transition-colors ${
            viewMode === "sessions" ? "bg-green-600 text-white" : "bg-gray-100 text-gray-700 hover:bg-gray-200"
          }`}
        >
          Sesiones
        </button>
        <button
          onClick={() => setViewMode("charts")}
          className={`px-4 py-2 rounded-lg font-medium transition-colors ${
            viewMode === "charts" ? "bg-green-600 text-white" : "bg-gray-100 text-gray-700 hover:bg-gray-200"
          }`}
        >
          Graficos
```

```
58          </button>
59        </div>
60
61        {/* Renderizado condicional de componentes */}
62        <div className="grid gap-6">
63          {viewMode === "sessions" && (
64            <>
65              <RunningForm onSessionAdded={handleSessionAdded} />
66              <RunningStats refreshTrigger={refreshTrigger} />
67              <RunningList refreshTrigger={refreshTrigger} />
68            </>
69          )}
70
71          {viewMode === "charts" && <RunningCharts />}
72        </div>
73      </div>
74    )
75  }
```

Listing 7: app/running/page.tsx - Estructura completa

Caracteristicas del componente:

- **Estado centralizado**: Maneja el estado de actualizacion y modo de vista
- **Navegacion por pestanas**: Interfaz intuitiva para cambiar entre sesiones y graficos
- **Renderizado condicional**: Muestra diferentes componentes segun el modo
- **Handlers de eventos**: Gestiona las interacciones del usuario
- **TypeScript**: Tipado estricto para mayor seguridad

4.2 Componente RunningForm

Formulario para registrar nuevas sesiones de running.

```
1  "use client"
2
3  import { useState } from "react"
4  import { Button } from "@/components/ui/button"
5  import { Input } from "@/components/ui/input"
6  import { Card, CardContent, CardDescription, CardHeader, CardTitle }
       from "@/components/ui/card"
7  import { Label } from "@/components/ui/label"
8  import { Plus, Loader2 } from "lucide-react"
9  import { createRunningSession } from "@/lib/running-actions"
10 import { useActionState } from "react"
11
12 export default function RunningForm({ onSessionAdded }: { onSessionAdded
       ?: () => void }) {
13   const [state, formAction] = useActionState(createRunningSession, null)
14   const [isOpen, setIsOpen] = useState(false)
15
16   const handleSubmit = async (formData: FormData) => {
17     const result = await formAction(formData)
18     if (result?.success) {
19       setIsOpen(false)
```

```
      onSessionAdded?.()
    }
  }

  // Renderizar boton si no esta abierto
  if (!isOpen) {
    return (
      <Button onClick={() => setIsOpen(true)} className="w-full bg-green-600 hover:bg-green-700">
        <Plus className="h-4 w-4 mr-2" />
        Registrar Sesion de Running
      </Button>
    )
  }

  return (
    <Card>
      <CardHeader>
        <CardTitle>Nueva Sesion de Running</CardTitle>
        <CardDescription>Registra tu sesion de carrera</CardDescription>
      </CardHeader>
      <CardContent>
        <form action={handleSubmit} className="space-y-4">
          {/* Mostrar errores si existen */}
          {state?.error && (
            <div className="bg-red-50 border border-red-200 text-red-700 px-4 py-3 rounded text-sm">
              {state.error}
            </div>
          )}

          <div className="grid grid-cols-1 gap-4">
            <div className="grid grid-cols-2 gap-4">
              <div className="space-y-2">
                <Label htmlFor="duration_minutes">Duracion (minutos)</Label>
                <Input
                  id="duration_minutes"
                  name="duration_minutes"
                  type="number"
                  min="1"
                  step="1"
                  placeholder="30"
                  required
                />
              </div>
              <div className="space-y-2">
                <Label htmlFor="distance_km">Distancia (km)</Label>
                <Input
                  id="distance_km"
                  name="distance_km"
                  type="number"
                  min="0.1"
                  step="0.1"
                  placeholder="5.0"
                  required
                />
              </div>
```

```
            </div>

            <div className="space-y-2">
              <Label htmlFor="pace_min_km">Ritmo (min/km) - Opcional</
    Label>
              <Input
                id="pace_min_km"
                name="pace_min_km"
                type="number"
                min="0"
                step="0.1"
                placeholder="6.0"
              />
              <p className="text-xs text-gray-500">
                Se calculara automaticamente si no se especifica
              </p>
            </div>
          </div>

          {/* Botones de accion */}
          <div className="flex gap-2">
            <Button type="submit" className="bg-green-600 hover:bg-green
    -700">
              {state?.success === false ? (
                <Loader2 className="h-4 w-4 mr-2 animate-spin" />
              ) : null}
              Guardar Sesion
            </Button>
            <Button
              type="button"
              variant="outline"
              onClick={() => setIsOpen(false)}
            >
              Cancelar
            </Button>
          </div>
        </form>
      </CardContent>
    </Card>
  )
}
```

Listing 8: components/running/running-form.tsx - Estructura principal

Caracteristicas del formulario:

- **Estado colapsable**: Se muestra como boton inicialmente, se expande al hacer clic
- **Validacion en cliente**: Campos requeridos y validacion de tipos
- **Calculo automatico**: El ritmo se calcula automaticamente si no se proporciona
- **Manejo de errores**: Muestra errores de validacion y base de datos
- **Feedback visual**: Indicador de carga durante el envio

4.3 Componente RunningStats

Componente que calcula y muestra estadisticas detalladas de las sesiones de running.

```
interface Stats {
  totalSessions: number
  totalDistance: number
  totalTime: number
  averagePace: number | null
  bestPace: number | null
  longestRun: number
}

export default function RunningStats({ refreshTrigger }: { refreshTrigger?: number }) {
  const [stats, setStats] = useState<Stats | null>(null)
  const [loading, setLoading] = useState(true)

  const calculateStats = (sessions: RunningSession[]): Stats => {
    if (sessions.length === 0) {
      return {
        totalSessions: 0,
        totalDistance: 0,
        totalTime: 0,
        averagePace: null,
        bestPace: null,
        longestRun: 0,
      }
    }

    // Calcular metricas basicas
    const totalDistance = sessions.reduce((sum, session) => sum + session.distance_km, 0)
    const totalTime = sessions.reduce((sum, session) => sum + session.duration_minutes, 0)
    const longestRun = Math.max(...sessions.map((session) => session.distance_km))

    // Calcular ritmo promedio y mejor ritmo
    const sessionsWithPace = sessions.filter((session) => session.pace_min_km !== null)
    const averagePace =
      sessionsWithPace.length > 0
        ? sessionsWithPace.reduce((sum, session) => sum + (session.pace_min_km || 0), 0) / sessionsWithPace.length
        : null

    const bestPace =
      sessionsWithPace.length > 0
        ? Math.min(...sessionsWithPace.map((session) => session.pace_min_km || Number.POSITIVE_INFINITY))
        : null

    return {
      totalSessions: sessions.length,
      totalDistance,
      totalTime,
      averagePace,
      bestPace,
      longestRun,
    }
```

```
51    }
52
53    const loadStats = async () => {
54      try {
55        const sessions = await getRunningSessions()
56        const calculatedStats = calculateStats(sessions || [])
57        setStats(calculatedStats)
58      } catch (error) {
59        console.error("Error loading running stats:", error)
60      } finally {
61        setLoading(false)
62      }
63    }
64
65    useEffect(() => {
66      loadStats()
67    }, [refreshTrigger])
68
69    // Funcion para formatear el ritmo
70    const formatPace = (pace: number | null) => {
71      if (!pace) return "N/A"
72      const minutes = Math.floor(pace)
73      const seconds = Math.round((pace - minutes) * 60)
74      return `${minutes}:${seconds.toString().padStart(2, "0")}`
75    }
76
77    // Funcion para formatear el tiempo
78    const formatTime = (minutes: number) => {
79      const hours = Math.floor(minutes / 60)
80      const mins = minutes % 60
81      if (hours > 0) {
82        return `${hours}h ${mins}m`
83      }
84      return `${mins}m`
85    }
86
87    if (loading) {
88      return (
89        <Card>
90          <CardContent className="p-6">
91            <div className="text-center text-gray-500">Cargando
       estadisticas...</div>
92          </CardContent>
93        </Card>
94      )
95    }
96
97    return (
98      <Card>
99        <CardHeader>
100         <CardTitle>Estadisticas de Running</CardTitle>
101         <CardDescription>Resumen de tu actividad de carrera</
       CardDescription>
102       </CardHeader>
103       <CardContent>
104         <div className="grid grid-cols-2 md:grid-cols-3 gap-4">
105           <div className="text-center">
106             <div className="text-2xl font-bold text-green-600">{stats?.
```

```
    totalSessions}</div>
107             <div className="text-sm text-gray-600">Sesiones Totales</div
    >
108           </div>
109           <div className="text-center">
110             <div className="text-2xl font-bold text-green-600">{stats?.
    totalDistance.toFixed(1)} km</div>
111             <div className="text-sm text-gray-600">Distancia Total</div>
112           </div>
113           <div className="text-center">
114             <div className="text-2xl font-bold text-green-600">{
    formatTime(stats?.totalTime || 0)}</div>
115             <div className="text-sm text-gray-600">Tiempo Total</div>
116           </div>
117           <div className="text-center">
118             <div className="text-2xl font-bold text-green-600">{
    formatPace(stats?.averagePace)}</div>
119             <div className="text-sm text-gray-600">Ritmo Promedio</div>
120           </div>
121           <div className="text-center">
122             <div className="text-2xl font-bold text-green-600">{
    formatPace(stats?.bestPace)}</div>
123             <div className="text-sm text-gray-600">Mejor Ritmo</div>
124           </div>
125           <div className="text-center">
126             <div className="text-2xl font-bold text-green-600">{stats?.
    longestRun.toFixed(1)} km</div>
127             <div className="text-sm text-gray-600">Carrera Mas Larga</
    div>
128           </div>
129         </div>
130       </CardContent>
131     </Card>
132   )
133 }
```

Listing 9: components/running/running-stats.tsx - Logica de calculo

Metricas calculadas:

- **Sesiones Totales**: Numero total de sesiones registradas
- **Distancia Total**: Suma de todas las distancias recorridas
- **Tiempo Total**: Suma de todos los tiempos de carrera
- **Ritmo Promedio**: Promedio de todos los ritmos registrados
- **Mejor Ritmo**: El ritmo mas rapido registrado
- **Carrera Mas Larga**: La distancia mas larga en una sola sesion

4.4 Componente RunningList

Lista que muestra todas las sesiones de running registradas.

```
"use client"

import { useState, useEffect } from "react"
import { Card, CardContent } from "@/components/ui/card"
import { Button } from "@/components/ui/button"
import { Badge } from "@/components/ui/badge"
import { Trash2, Calendar, Clock, MapPin, Zap } from "lucide-react"
import { deleteRunningSession, getRunningSessions } from "@/lib/running-actions"
import { format } from "date-fns"
import { es } from "date-fns/locale"

interface RunningSession {
  id: string
  duration_minutes: number
  distance_km: number
  pace_min_km: number | null
  created_at: string
}

export default function RunningList({ refreshTrigger }: { refreshTrigger?: number }) {
  const [sessions, setSessions] = useState<RunningSession[]>([])
  const [loading, setLoading] = useState(true)

  const loadSessions = async () => {
    try {
      const data = await getRunningSessions()
      setSessions(data || [])
    } catch (error) {
      console.error("Error loading running sessions:", error)
    } finally {
      setLoading(false)
    }
  }

  useEffect(() => {
    loadSessions()
  }, [refreshTrigger])

  const handleDelete = async (id: string) => {
    if (confirm("Estas seguro de que quieres eliminar esta sesion?")) {
      const result = await deleteRunningSession(id)
      if (result?.success) {
        loadSessions()
      }
    }
  }

  const formatPace = (pace: number | null) => {
    if (!pace) return null
    const minutes = Math.floor(pace)
    const seconds = Math.round((pace - minutes) * 60)
    return `${minutes}:${seconds.toString().padStart(2, "0")}`
  }

  const formatTime = (minutes: number) => {
```

```
    const hours = Math.floor(minutes / 60)
    const mins = minutes % 60
    if (hours > 0) {
      return `${hours}h ${mins}m`
    }
    return `${mins}m`
  }

  // Estado de carga
  if (loading) {
    return (
      <Card>
        <CardContent className="p-6">
          <div className="text-center text-gray-500">Cargando sesiones
  ...</div>
        </CardContent>
      </Card>
    )
  }

  // Estado vacio
  if (sessions.length === 0) {
    return (
      <Card>
        <CardContent className="p-6">
          <div className="text-center text-gray-500">
            <MapPin className="h-12 w-12 mx-auto mb-4 text-gray-300" />
            <p>No hay sesiones de running registradas</p>
            <p className="text-sm">Registra tu primera carrera para
  comenzar</p>
          </div>
        </CardContent>
      </Card>
    )
  }

  return (
    <div className="space-y-4">
      <h2 className="text-xl font-semibold">Historial de Sesiones</h2>
      {sessions.map((session) => (
        <Card key={session.id}>
          <CardContent className="p-4">
            <div className="flex items-center justify-between">
              <div className="flex-1">
                <div className="flex items-center gap-4 mb-2">
                  <div className="flex items-center gap-1 text-sm text-
  gray-600">
                    <Calendar className="h-4 w-4" />
                    {format(new Date(session.created_at), "dd/MM/yyyy",
  { locale: es })}
                  </div>
                  <div className="flex items-center gap-1 text-sm text-
  gray-600">
                    <Clock className="h-4 w-4" />
                    {formatTime(session.duration_minutes)}
                  </div>
                  <div className="flex items-center gap-1 text-sm text-
  gray-600">
```

```
                    <MapPin className="h-4 w-4" />
                    {session.distance_km.toFixed(1)} km
                  </div>
                  {session.pace_min_km && (
                    <div className="flex items-center gap-1 text-sm text
    -gray-600">
                      <Zap className="h-4 w-4" />
                      {formatPace(session.pace_min_km)}/km
                    </div>
                  )}
                </div>

                {/* Badges con metricas */}
                <div className="flex gap-2">
                  <Badge variant="secondary">
                    {session.distance_km.toFixed(1)} km
                  </Badge>
                  <Badge variant="secondary">
                    {formatTime(session.duration_minutes)}
                  </Badge>
                  {session.pace_min_km && (
                    <Badge variant="secondary">
                      {formatPace(session.pace_min_km)}/km
                    </Badge>
                  )}
                </div>
              </div>

              {/* Botones de accion */}
              <div className="flex gap-2">
                <Button
                  variant="outline"
                  size="sm"
                  onClick={() => handleDelete(session.id)}
                  className="text-red-600 hover:text-red-700"
                >
                  <Trash2 className="h-4 w-4" />
                </Button>
              </div>
            </div>
          </CardContent>
        </Card>
      ))}
    </div>
  )
}
```

Listing 10: components/running/running-list.tsx - Estructura principal

Caracteristicas de la lista:

- **Carga asincrona**: Carga las sesiones desde la base de datos
- **Formateo de datos**: Convierte tiempos y ritmos a formato legible
- **Estado vacio**: Muestra mensaje cuando no hay sesiones
- **Eliminacion**: Permite eliminar sesiones con confirmacion
- **Actualizacion automatica**: Se actualiza cuando se agregan nuevas sesiones

4.5 Componente RunningCharts

Componente para visualizar graficos de progreso y tendencias.

```
"use client"

import { useState, useEffect } from "react"
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"
import { getRunningSessions } from "@/lib/running-actions"

interface RunningSession {
  id: string
  duration_minutes: number
  distance_km: number
  pace_min_km: number | null
  created_at: string
}

export default function RunningCharts() {
  const [sessions, setSessions] = useState<RunningSession[]>([])
  const [loading, setLoading] = useState(true)

  useEffect(() => {
    const loadSessions = async () => {
      try {
        const data = await getRunningSessions()
        setSessions(data || [])
      } catch (error) {
        console.error("Error loading running sessions:", error)
      } finally {
        setLoading(false)
      }
    }

    loadSessions()
  }, [])

  if (loading) {
    return (
      <Card>
        <CardContent className="p-6">
          <div className="text-center text-gray-500">Cargando graficos ...</div>
        </CardContent>
      </Card>
    )
  }

  if (sessions.length === 0) {
    return (
      <Card>
        <CardContent className="p-6">
          <div className="text-center text-gray-500">
            <p>No hay datos suficientes para mostrar graficos</p>
            <p className="text-sm">Registra algunas sesiones para ver tu progreso</p>
          </div>
        </CardContent>
```

```
      </Card>
    )
  }

  return (
    <div className="space-y-6">
      <Card>
        <CardHeader>
          <CardTitle>Progreso de Distancia</CardTitle>
        </CardHeader>
        <CardContent>
          {/* Aqui se implementarian los graficos con una libreria como Chart.js o Recharts */}
          <div className="h-64 flex items-center justify-center text-gray-500">
            Grafico de progreso de distancia (implementar con libreria de graficos)
          </div>
        </CardContent>
      </Card>

      <Card>
        <CardHeader>
          <CardTitle>Evolucion del Ritmo</CardTitle>
        </CardHeader>
        <CardContent>
          <div className="h-64 flex items-center justify-center text-gray-500">
            Grafico de evolucion del ritmo (implementar con libreria de graficos)
          </div>
        </CardContent>
      </Card>

      <Card>
        <CardHeader>
          <CardTitle>Frecuencia de Entrenamiento</CardTitle>
        </CardHeader>
        <CardContent>
          <div className="h-64 flex items-center justify-center text-gray-500">
            Grafico de frecuencia de entrenamiento (implementar con libreria de graficos)
          </div>
        </CardContent>
      </Card>
    </div>
  )
}
```

Listing 11: components/running/running-charts.tsx - Estructura basica

Tipos de graficos implementables:

- **Progreso de Distancia**: Linea temporal mostrando la evolucion de las distancias
- **Evolucion del Ritmo**: Grafico de ritmo promedio por sesion
- **Frecuencia de Entrenamiento**: Histograma de sesiones por semana/mes

- **Comparacion de Metricas**: Graficos de barras comparando diferentes periodos

5 Ejemplos Practicos de Uso

5.1 Ejemplos de Insercion a Base de Datos

5.1.1 Crear una Sesion de Running

```
-- Insertar una sesion de running de 5km en 30 minutos
INSERT INTO public.running_sessions (
    user_id,
    duration_minutes,
    distance_km,
    pace_min_km
) VALUES (
    '123e4567-e89b-12d3-a456-426614174000', -- UUID del usuario
    30,                                     -- 30 minutos
    5.0,                                    -- 5 kilometros
    6.0                                     -- 6 minutos por kilometro
);

-- Insertar una sesion de running de 10km en 50 minutos
INSERT INTO public.running_sessions (
    user_id,
    duration_minutes,
    distance_km,
    pace_min_km
) VALUES (
    '123e4567-e89b-12d3-a456-426614174000',
    50,                                     -- 50 minutos
    10.0,                                   -- 10 kilometros
    5.0                                     -- 5 minutos por kilometro
);

-- Insertar una sesion sin ritmo especificado (se calcula
   automaticamente)
INSERT INTO public.running_sessions (
    user_id,
    duration_minutes,
    distance_km
) VALUES (
    '123e4567-e89b-12d3-a456-426614174000',
    25,                                     -- 25 minutos
    4.2                                     -- 4.2 kilometros
    -- pace_min_km se calculara como 25/4.2 = 5.95 min/km
);
```

Listing 12: Ejemplo de insercion en $running_sessions$

5.1.2 Consultas Utiles para Analisis

```
-- Obtener todas las sesiones de un usuario con detalles
SELECT
    rs.duration_minutes,
    rs.distance_km,
```

```
    rs.pace_min_km,
    rs.created_at,
    (rs.distance_km / rs.duration_minutes * 60) as velocidad_kmh
FROM public.running_sessions rs
WHERE rs.user_id = '123e4567-e89b-12d3-a456-426614174000'
ORDER BY rs.created_at DESC;

-- Obtener estadisticas resumidas de un usuario
SELECT
    COUNT(*) as total_sesiones,
    SUM(distance_km) as distancia_total,
    SUM(duration_minutes) as tiempo_total,
    AVG(pace_min_km) as ritmo_promedio,
    MIN(pace_min_km) as mejor_ritmo,
    MAX(distance_km) as carrera_mas_larga
FROM public.running_sessions
WHERE user_id = '123e4567-e89b-12d3-a456-426614174000';

-- Obtener progreso mensual
SELECT
    DATE_TRUNC('month', created_at) as mes,
    COUNT(*) as sesiones_mes,
    SUM(distance_km) as distancia_mes,
    AVG(pace_min_km) as ritmo_promedio_mes
FROM public.running_sessions
WHERE user_id = '123e4567-e89b-12d3-a456-426614174000'
GROUP BY DATE_TRUNC('month', created_at)
ORDER BY mes DESC;

-- Obtener las 5 mejores sesiones por ritmo
SELECT
    distance_km,
    duration_minutes,
    pace_min_km,
    created_at
FROM public.running_sessions
WHERE user_id = '123e4567-e89b-12d3-a456-426614174000'
  AND pace_min_km IS NOT NULL
ORDER BY pace_min_km ASC
LIMIT 5;

-- Obtener tendencia de mejora en el ultimo mes
SELECT
    DATE_TRUNC('week', created_at) as semana,
    AVG(pace_min_km) as ritmo_promedio_semana
FROM public.running_sessions
WHERE user_id = '123e4567-e89b-12d3-a456-426614174000'
  AND created_at >= NOW() - INTERVAL '1 month'
  AND pace_min_km IS NOT NULL
GROUP BY DATE_TRUNC('week', created_at)
ORDER BY semana ASC;
```

Listing 13: Consultas utiles para analisis de running

5.2 Flujos de Datos Completos

5.2.1 Flujo de Creacion de Sesion

1. **Usuario completa formulario** en `RunningForm`
2. **Validacion en cliente** de campos requeridos (duracion y distancia)
3. **Envio a Server Action** `createRunningSession`
4. **Verificacion de autenticacion** en servidor
5. **Calculo automatico de ritmo** si no se proporciona
6. **Insercion en base de datos** tabla `running_sessions`
7. **Revalidacion de cache** con `revalidatePath`
8. **Actualizacion de UI** con nueva sesion y estadisticas

5.2.2 Flujo de Calculo de Estadisticas

1. **Carga de sesiones** con `getRunningSessions`
2. **Calculo de metricas basicas** (total sesiones, distancia, tiempo)
3. **Calculo de ritmo promedio** de sesiones con ritmo registrado
4. **Identificacion del mejor ritmo** (valor minimo)
5. **Identificacion de la carrera mas larga** (distancia maxima)
6. **Formateo de datos** para presentacion
7. **Renderizado de estadisticas** en `RunningStats`

6 Caracteristicas Avanzadas

6.1 Calculos Automaticos

El modulo implementa varios calculos automaticos:

- **Ritmo por Kilometro**: `duracion_minutos / distancia_km`
- **Velocidad en km/h**: `distancia_km / (duracion_minutos / 60)`
- **Ritmo Promedio**: Promedio de todos los ritmos registrados
- **Mejor Ritmo**: Valor minimo de ritmo (mas rapido)
- **Distancia Total**: Suma de todas las distancias
- **Tiempo Total**: Suma de todos los tiempos

6.2 Validaciones y Restricciones

- **Duracion**: Debe ser mayor a 0 minutos
- **Distancia**: Debe ser mayor a 0 kilometros
- **Ritmo**: Opcional, se calcula automaticamente si no se proporciona
- **Usuario**: Debe estar autenticado para todas las operaciones
- **RLS**: Solo se puede acceder a sesiones propias

6.3 Formateo de Datos

- **Tiempo**: Convierte minutos a formato "Xh Ym.º "Ym"
- **Ritmo**: Convierte decimal a formato "X:YY"(minutos:segundos)
- **Distancia**: Muestra con 1 decimal (ej: "5.0 km")
- **Fechas**: Formato localizado (ej: "14/10/2025")

7 Mejores Practicas de Desarrollo

7.1 Seguridad

- **Validacion de entrada**: Siempre validar duracion y distancia
- **Autenticacion**: Verificar usuario en cada operacion
- **RLS**: Usar Row Level Security en todas las tablas
- **Sanitizacion**: Limpiar y convertir datos de entrada
- **Logs de seguridad**: Registrar operaciones sensibles

7.2 Performance

- **Indices de base de datos**: Optimizar consultas por usuario y fecha
- **Cache**: Usar revalidatePath para actualizar cache
- **Calculos eficientes**: Optimizar calculos de estadisticas
- **Lazy loading**: Cargar graficos bajo demanda
- **Paginacion**: Implementar para listas largas de sesiones

7.3 Mantenibilidad

- **TypeScript**: Usar tipado estricto en todos los componentes
- **Interfaces**: Definir interfaces claras para props
- **Separacion de responsabilidades**: Logica en Server Actions
- **Reutilizacion**: Componentes modulares y reutilizables
- **Documentacion**: Comentar codigo complejo

8 Solucion de Problemas

8.1 Problemas Comunes

8.1.1 Error: Tabla no existe

Sintomas: Error al intentar insertar o consultar datos **Causa**: Las tablas no han sido creadas en la base de datos **Solucion**: Ejecutar los scripts SQL en orden:

1. `01-create-database-schema.sql`
2. `01-create-user-schema.sql`
3. `02-create-history-schema.sql`

8.1.2 Error: Usuario no autenticado

Sintomas: Server Actions retornan error de autenticacion **Causa**: Usuario no esta logueado o sesion expirada **Solucion**: Verificar estado de autenticacion y redirigir a login

8.1.3 Error: Calculo de ritmo incorrecto

Sintomas: El ritmo calculado no coincide con el esperado **Causa**: Division por cero o valores invalidos **Solucion**: Validar que distancia sea mayor a 0 antes del calculo

8.2 Debugging

8.2.1 Logs del Cliente

```
// Habilitar logs detallados
localStorage.setItem('debug', 'true');

// Verificar estado de autenticacion
console.log('User:', user);
console.log('Session:', session);

// Verificar datos de sesiones
console.log('Running Sessions:', sessions);
```

Listing 14: Debugging en cliente

8.2.2 Logs del Servidor

```
// En Server Actions
console.log('Action called with:', { userId, data });

// En consultas de base de datos
console.log('Query result:', { data, error });

// En calculos de ritmo
console.log('Pace calculation:', { duration, distance, calculatedPace })
    ;
```

Listing 15: Debugging en servidor

9 Conclusion

El modulo de running de FitTrack es un sistema completo y robusto que permite a los usuarios gestionar sus sesiones de carrera de manera eficiente. Con su arquitectura bien definida, base de datos optimizada y componentes React modernos, proporciona una experiencia de usuario excepcional.

Caracteristicas destacadas:

- Arquitectura escalable y mantenible
- Seguridad robusta con RLS
- Interfaz de usuario intuitiva
- Calculos automaticos precisos
- Estadisticas detalladas y utiles
- Codigo bien documentado y tipado

Para contribuir al desarrollo del modulo:

1. Seguir las convenciones establecidas
2. Implementar tests apropiados
3. Documentar cambios significativos
4. Mantener la compatibilidad con la base de datos
5. Respetar las politicas de seguridad