🎯 Guida Completa: ArmoniaWellness da Zero

📋 Indice

🎯 Panoramica del Progetto {#panoramica}

ArmoniaWellness è un'app React per gestione prenotazioni centro benessere.

Funzionalità:

- ✅ Sistema autenticazione (admin/admin)
- ✅ Dashboard con statistiche
- ✅ Gestione prenotazioni CRUD
- ✅ Viste: Giorno/Settimana/Mese
- ✅ Interfaccia responsive
- ✅ Gestione orari di lavoro

Tecnologie:

- React 19 + TypeScript
- Vite + Tailwind CSS 4
- Lucide React (icone)
- ESLint

🛠️ Setup Iniziale {#setup}

1. Creare Progetto con Vite (Raccomandato)

```
# Crea progetto direttamente con Vite
npm create vite@latest ArmoniaWellness -- --template react-ts

# Naviga nella cartella
cd ArmoniaWellness
```

2. Installare Dipendenze

```
# Installa dipendenze base (già incluse nel template)
npm install

# Aggiungi dipendenze aggiuntive
npm install lucide-react

# Installa Tailwind CSS
npm install -D tailwindcss @tailwindcss/vite
```

3. Configurare Tailwind CSS

tailwind.config.js:

```
export default {
  content: ["./index.html", "./src/**/*.{js,ts,jsx,tsx}"],
  theme: {
    extend: {
      colors: {
        primary: { 50: '#eef2ff', 600: '#4f46e5' },
        secondary: { 50: '#fdf2f8', 600: '#db2777' }
      }
    }
  }
}
```

4. Aggiornare CSS

src/index.css:

```
@import "tailwindcss";

@layer base {
  :root {
    --color-primary: 99 102 241;
    --color-secondary: 236 72 153;
    --color-accent: 168 85 247;
  }
}

@layer components {
  .btn-primary {
    @apply bg-indigo-600 hover:bg-indigo-700 text-white font-medium py-2
px-4 rounded-lg transition-colors duration-200;
  }

  .btn-outline {
    @apply border-2 border-indigo-600 text-indigo-600 hover:bg-indigo-600
hover:text-white font-medium py-2 px-4 rounded-lg transition-colors
```

```
duration-200;
  }

  .card {
    @apply bg-white rounded-xl shadow-lg border border-gray-100 p-6;
  }

  .input-field {
    @apply w-full px-3 py-2 border border-gray-300 rounded-lg
focus:outline-none focus:ring-2 focus:ring-indigo-500 focus:border-
transparent;
  }
}
```

📁 Struttura e Tipi {#struttura}

Struttura Cartelle

```
src/
├── components/     # Componenti React
├── types/         # Definizioni TypeScript
├── assets/        # Immagini/risorse
├── App.tsx        # Componente principale
├── main.tsx       # Entry point
└── index.css      # Stili globali
```

Tipi TypeScript

src/types/index.ts:

```
export interface Treatment {
  id: string;
  name: string;
  duration: number;
  price: number;
  category: 'massage' | 'facial' | 'body' | 'wellness';
}

export interface Appointment {
  id: string;
  clientName: string;
  clientPhone: string;
  clientEmail: string;
  treatmentId: string;
  treatment: Treatment;
  date: string;
  startTime: string;
  endTime: string;
```

```
  notes?: string;
  status: 'confirmed' | 'pending' | 'cancelled';
  createdAt: Date;
  updatedAt: Date;
}

export interface User {
  username: string;
  isAuthenticated: boolean;
}

export type ViewMode = 'day' | 'week' | 'month';
```

🧩 Componenti Base {#componenti-base}

1. Logo Component

src/components/Logo.tsx:

```
import React from 'react';
import { Flower } from 'lucide-react';

interface LogoProps {
  variant?: 'color' | 'white';
  size?: 'sm' | 'md' | 'lg' | 'xl';
  className?: string;
}

const Logo: React.FC<LogoProps> = ({
  variant = 'color',
  size = 'md',
  className = ''
}) => {
  const sizeClasses = {
    sm: 'w-6 h-6', md: 'w-8 h-8', lg: 'w-10 h-10', xl: 'w-12 h-12'
  };

  const iconColor = variant === 'white' ? 'text-white' : 'text-indigo-
600';
  const textColor = variant === 'white' ? 'text-white' : 'text-gray-900';

  return (
    <div className={`flex items-center gap-2 ${className}`}>
      <Flower className={`${sizeClasses[size]} ${iconColor}`} />
      <div className="flex flex-col">
        <span className={`font-bold text-lg leading-tight ${textColor}`}>
          Armonia
        </span>
        <span className={`font-light text-sm leading-tight ${textColor}`}>
          Wellness
```

```
          </span>
        </div>
      </div>
    );
  };

  export default Logo;
```

2. Login Component

src/components/Login.tsx:

```
import React, { useState } from 'react';
import { Eye, EyeOff, Lock, User } from 'lucide-react';
import Logo from './Logo';

interface LoginProps {
  onLogin: (username: string) => void;
}

const Login: React.FC<LoginProps> = ({ onLogin }) => {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [showPassword, setShowPassword] = useState(false);
  const [error, setError] = useState('');
  const [isLoading, setIsLoading] = useState(false);

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    setError('');
    setIsLoading(true);

    // Simulazione login
    setTimeout(() => {
      if (username === 'admin' && password === 'admin') {
        onLogin(username);
      } else {
        setError('Credenziali non valide. Usa admin/admin');
      }
      setIsLoading(false);
    }, 1000);
  };

  return (
    <div className="min-h-screen bg-gradient-to-br from-indigo-50 via-
fuchsia-50 to-purple-50 flex items-center justify-center p-4">
      <div className="card w-full max-w-md">
        <div className="text-center mb-8">
          <Logo size="xl" className="mx-auto mb-4" />
          <h1 className="text-2xl font-bold text-gray-900 mb-2">
            Benvenuto
```

```
            </h1>
            <p className="text-gray-600">
              Accedi al sistema di gestione prenotazioni
            </p>
          </div>

          <form onSubmit={handleSubmit} className="space-y-6">
            {/* Username Field */}
            <div>
              <label htmlFor="username" className="block text-sm font-medium
text-gray-700 mb-2">
                Username
              </label>
              <div className="relative">
                <User className="absolute left-3 top-1/2 transform -
translate-y-1/2 text-gray-400 w-5 h-5" />
                <input
                  id="username"
                  type="text"
                  value={username}
                  onChange={(e) => setUsername(e.target.value)}
                  className="input-field pl-10"
                  placeholder="Inserisci username"
                  required
                />
              </div>
            </div>

            {/* Password Field */}
            <div>
              <label htmlFor="password" className="block text-sm font-medium
text-gray-700 mb-2">
                Password
              </label>
              <div className="relative">
                <Lock className="absolute left-3 top-1/2 transform -
translate-y-1/2 text-gray-400 w-5 h-5" />
                <input
                  id="password"
                  type={showPassword ? 'text' : 'password'}
                  value={password}
                  onChange={(e) => setPassword(e.target.value)}
                  className="input-field pl-10 pr-10"
                  placeholder="Inserisci password"
                  required
                />
                <button
                  type="button"
                  onClick={() => setShowPassword(!showPassword)}
                  className="absolute right-3 top-1/2 transform -translate-
y-1/2 text-gray-400 hover:text-gray-600"
                >
                  {showPassword ? <EyeOff className="w-5 h-5" /> : <Eye
className="w-5 h-5" />}
```

```
              </button>
            </div>
          </div>

          {/* Error Display */}
          {error && (
            <div className="bg-red-50 border border-red-200 text-red-700
px-4 py-3 rounded-lg text-sm">
              {error}
            </div>
          )}

          {/* Submit Button */}
          <button
            type="submit"
            disabled={isLoading}
            className="btn-primary w-full flex items-center justify-center
gap-2 disabled:opacity-50 disabled:cursor-not-allowed"
          >
            {isLoading ? (
              <>
                <div className="w-4 h-4 border-2 border-white border-t-
transparent rounded-full animate-spin"></div>
                Accesso in corso...
              </>
            ) : (
              'Accedi'
            )}
          </button>
        </form>

        {/* Demo Credentials */}
        <div className="mt-6 p-4 bg-blue-50 border border-blue-200
rounded-lg">
          <p className="text-sm text-blue-800 text-center">
            <strong>Credenziali demo:</strong><br />
            Username: <code className="bg-blue-100 px-1
rounded">admin</code><br />
            Password: <code className="bg-blue-100 px-1
rounded">admin</code>
          </p>
        </div>
      </div>
    </div>
  );
};

export default Login;
```

3. Navbar Component

src/components/Navbar.tsx:

```
import React, { useState } from 'react';
import { LogOut, Menu, X } from 'lucide-react';
import Logo from './Logo';

interface NavbarProps {
  username: string;
  onLogout: () => void;
}

const Navbar: React.FC<NavbarProps> = ({ username, onLogout }) => {
  const [isMobileMenuOpen, setIsMobileMenuOpen] = useState(false);

  return (
    <nav className="bg-white shadow-lg border-b border-gray-100">
      <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
        <div className="flex justify-between items-center h-16">
          {/* Logo */}
          <div className="flex-shrink-0">
            <Logo size="lg" />
          </div>

          {/* Desktop Navigation */}
          <div className="hidden md:block">
            <div className="ml-10 flex items-baseline space-x-4">
              <span className="text-gray-700 px-3 py-2 rounded-md text-sm
font-medium">
                Benvenuto, <span className="text-indigo-600 font-
semibold">{username}</span>
              </span>
            </div>
          </div>

          {/* Logout Button */}
          <div className="hidden md:block">
            <button
              onClick={onLogout}
              className="btn-outline flex items-center gap-2"
            >
              <LogOut className="w-4 h-4" />
              Logout
            </button>
          </div>

          {/* Mobile menu button */}
          <div className="md:hidden">
            <button
              onClick={() => setIsMobileMenuOpen(!isMobileMenuOpen)}
              className="text-gray-700 hover:text-indigo-600 p-2"
            >
              {isMobileMenuOpen ? (
                <X className="w-6 h-6" />
              ) : (
                <Menu className="w-6 h-6" />
```

```
                )}
              </button>
            </div>
          </div>
        </div>

        {/* Mobile menu */}
        {isMobileMenuOpen && (
          <div className="md:hidden">
            <div className="px-2 pt-2 pb-3 space-y-1 sm:px-3 bg-gray-50
border-t border-gray-200">
              <div className="px-3 py-2 text-sm text-gray-700">
                Benvenuto, <span className="text-indigo-600 font-semibold">
{username}</span>
              </div>
              <button
                onClick={onLogout}
                className="w-full text-left px-3 py-2 text-sm text-gray-700
hover:text-indigo-600 hover:bg-gray-100 rounded-md flex items-center gap-
2"
              >
                <LogOut className="w-4 h-4" />
                Logout
              </button>
            </div>
          </div>
        )}
      </nav>
    );
  };

  export default Navbar;
```

📅 Viste Calendario {#viste-calendario}

1. ViewSelector Component

src/components/ViewSelector.tsx:

```
import React from 'react';
import { Calendar, CalendarDays, CalendarRange } from 'lucide-react';
import type { ViewMode } from '../types/index';

interface ViewSelectorProps {
  currentView: ViewMode;
  onViewChange: (view: ViewMode) => void;
}

const ViewSelector: React.FC<ViewSelectorProps> = ({ currentView,
onViewChange }) => {
```

```
  const views = [
    { id: 'day' as ViewMode, label: 'Giorno', icon: Calendar },
    { id: 'week' as ViewMode, label: 'Settimana', icon: CalendarRange },
    { id: 'month' as ViewMode, label: 'Mese', icon: CalendarDays },
  ];

  return (
    <div className="flex bg-gray-100 rounded-lg p-1">
      {views.map((view) => {
        const Icon = view.icon;
        const isActive = currentView === view.id;

        return (
          <button
            key={view.id}
            onClick={() => onViewChange(view.id)}
            className={`flex items-center gap-2 px-4 py-2 rounded-md text-
sm font-medium transition-colors duration-200 ${
              isActive
                ? 'bg-white text-indigo-600 shadow-sm'
                : 'text-gray-600 hover:text-gray-900 hover:bg-gray-200'
            }`}
          >
            <Icon className="w-4 h-4" />
            <span className="hidden sm:inline">{view.label}</span>
          </button>
        );
      })}
    </div>
  );
};

export default ViewSelector;
```

2. DateNavigator Component

src/components/DateNavigator.tsx:

```
import React from 'react';
import { ChevronLeft, ChevronRight, Calendar } from 'lucide-react';

interface DateNavigatorProps {
  currentDate: Date;
  onDateChange: (date: Date) => void;
  viewMode: 'day' | 'week' | 'month';
}

const DateNavigator: React.FC<DateNavigatorProps> = ({
  currentDate,
  onDateChange,
  viewMode
```

```
}) => {
  const formatDate = (date: Date, mode: string) => {
    const options: Intl.DateTimeFormatOptions = {
      year: 'numeric',
      month: 'long',
    };

    if (mode === 'day') {
      options.day = 'numeric';
      options.weekday = 'long';
    } else if (mode === 'week') {
      const startOfWeek = new Date(date);
      const endOfWeek = new Date(date);
      startOfWeek.setDate(date.getDate() - date.getDay() + 1);
      endOfWeek.setDate(startOfWeek.getDate() + 6);

      return `${startOfWeek.toLocaleDateString('it-IT', { day: 'numeric',
month: 'short' })} - ${endOfWeek.toLocaleDateString('it-IT', { day:
'numeric', month: 'short', year: 'numeric' })}`;
    }

    return date.toLocaleDateString('it-IT', options);
  };

  const navigateDate = (direction: 'prev' | 'next') => {
    const newDate = new Date(currentDate);

    if (viewMode === 'day') {
      newDate.setDate(currentDate.getDate() + (direction === 'next' ? 1 :
-1));
    } else if (viewMode === 'week') {
      newDate.setDate(currentDate.getDate() + (direction === 'next' ? 7 :
-7));
    } else if (viewMode === 'month') {
      newDate.setMonth(currentDate.getMonth() + (direction === 'next' ? 1
: -1));
    }

    onDateChange(newDate);
  };

  const goToToday = () => {
    onDateChange(new Date());
  };

  return (
    <div className="flex items-center gap-4">
      <button
        onClick={() => navigateDate('prev')}
        className="p-2 text-gray-600 hover:text-indigo-600 hover:bg-gray-
100 rounded-lg transition-colors duration-200"
      >
        <ChevronLeft className="w-5 h-5" />
      </button>
```

```
      <div className="flex items-center gap-3">
        <Calendar className="w-5 h-5 text-indigo-600" />
        <h2 className="text-xl font-semibold text-gray-900">
          {formatDate(currentDate, viewMode)}
        </h2>
      </div>

      <button
        onClick={() => navigateDate('next')}
        className="p-2 text-gray-600 hover:text-indigo-600 hover:bg-gray-
100 rounded-lg transition-colors duration-200"
      >
        <ChevronRight className="w-5 h-5" />
      </button>

      <button
        onClick={goToToday}
        className="btn-outline text-sm px-3 py-1"
      >
        Oggi
      </button>
    </div>
  );
};

export default DateNavigator;
```

📝 Form e Gestione {#form-gestione}

AppointmentForm Component

src/components/AppointmentForm.tsx:

```
import React, { useState, useEffect } from 'react';
import { X, User, Phone, Mail, Calendar, Clock, FileText, DollarSign }
from 'lucide-react';
import type { Appointment, Treatment } from '../types/index';

interface AppointmentFormProps {
  isOpen: boolean;
  onClose: () => void;
  onSubmit: (appointmentData: Omit<Appointment, 'id' | 'createdAt' |
'updatedAt'>) => void;
  appointment?: Appointment;
  treatments: Treatment[];
  selectedDate?: string;
  selectedTime?: string;
}
```

```
const AppointmentForm: React.FC<AppointmentFormProps> = ({
  isOpen,
  onClose,
  onSubmit,
  appointment,
  treatments,
  selectedDate,
  selectedTime
}) => {
  const [formData, setFormData] = useState({
    clientName: '',
    clientPhone: '',
    clientEmail: '',
    treatmentId: '',
    date: '',
    startTime: '',
    notes: ''
  });

  const [errors, setErrors] = useState<Record<string, string>>({});

  useEffect(() => {
    if (appointment) {
      setFormData({
        clientName: appointment.clientName,
        clientPhone: appointment.clientPhone,
        clientEmail: appointment.clientEmail,
        treatmentId: appointment.treatmentId,
        date: appointment.date,
        startTime: appointment.startTime,
        notes: appointment.notes || ''
      });
    } else {
      setFormData({
        clientName: '',
        clientPhone: '',
        clientEmail: '',
        treatmentId: '',
        date: selectedDate || '',
        startTime: selectedTime || '',
        notes: ''
      });
    }
    setErrors({});
  }, [appointment, selectedDate, selectedTime]);

  const validateForm = () => {
    const newErrors: Record<string, string> = {};

    if (!formData.clientName.trim()) {
      newErrors.clientName = 'Il nome è obbligatorio';
    }

    if (!formData.clientPhone.trim()) {
```

```
        newErrors.clientPhone = 'Il telefono è obbligatorio';
      }

      if (!formData.clientEmail.trim()) {
        newErrors.clientEmail = 'L\'email è obbligatoria';
      } else if (!/\S+@\S+\.\S+/.test(formData.clientEmail)) {
        newErrors.clientEmail = 'Email non valida';
      }

      if (!formData.treatmentId) {
        newErrors.treatmentId = 'Seleziona un trattamento';
      }

      if (!formData.date) {
        newErrors.date = 'Seleziona una data';
      }

      if (!formData.startTime) {
        newErrors.startTime = 'Seleziona un orario';
      }

      setErrors(newErrors);
      return Object.keys(newErrors).length === 0;
    };

    const handleSubmit = (e: React.FormEvent) => {
      e.preventDefault();

      if (!validateForm()) return;

      const selectedTreatment = treatments.find(t => t.id ===
  formData.treatmentId);
      if (!selectedTreatment) return;

      const [startHour, startMinute] =
  formData.startTime.split(':').map(Number);
      const startMinutes = startHour * 60 + startMinute;
      const endMinutes = startMinutes + selectedTreatment.duration;
      const endHour = Math.floor(endMinutes / 60);
      const endMinute = endMinutes % 60;
      const endTime = `${endHour.toString().padStart(2,
  '0')}:${endMinute.toString().padStart(2, '0')}`;

      const appointmentData = {
        clientName: formData.clientName.trim(),
        clientPhone: formData.clientPhone.trim(),
        clientEmail: formData.clientEmail.trim(),
        treatmentId: formData.treatmentId,
        treatment: selectedTreatment,
        date: formData.date,
        startTime: formData.startTime,
        endTime,
        notes: formData.notes.trim(),
        status: 'confirmed' as const
```

```
    };

    onSubmit(appointmentData);
    onClose();
  };

  const handleInputChange = (field: string, value: string) => {
    setFormData(prev => ({ ...prev, [field]: value }));
    if (errors[field]) {
      setErrors(prev => ({ ...prev, [field]: '' }));
    }
  };

  if (!isOpen) return null;

  return (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
justify-center p-4 z-50">
      <div className="bg-white rounded-xl shadow-2xl max-w-md w-full max-
h-[90vh] overflow-y-auto">
        <div className="p-6">
          <div className="flex items-center justify-between mb-6">
            <h2 className="text-xl font-semibold text-gray-900">
              {appointment ? 'Modifica Prenotazione' : 'Nuova
Prenotazione'}
            </h2>
            <button
              onClick={onClose}
              className="p-2 text-gray-400 hover:text-gray-600 hover:bg-
gray-100 rounded-lg"
            >
              <X className="w-5 h-5" />
            </button>
          </div>

          <form onSubmit={handleSubmit} className="space-y-4">
            {/* Nome Cliente */}
            <div>
              <label htmlFor="clientName" className="block text-sm font-
medium text-gray-700 mb-2">
                Nome Cliente *
              </label>
              <div className="relative">
                <User className="absolute left-3 top-1/2 transform -
translate-y-1/2 text-gray-400 w-5 h-5" />
                <input
                  id="clientName"
                  type="text"
                  value={formData.clientName}
                  onChange={(e) => handleInputChange('clientName',
e.target.value)}
                  className={`input-field pl-10 ${errors.clientName ?
'border-red-500' : ''}`}
                  placeholder="Nome e cognome"
```

```
                />
              </div>
              {errors.clientName && (
                <p className="text-red-500 text-sm mt-1">
{errors.clientName}</p>
              )}
            </div>

            {/* Altri campi simili... */}

            {/* Pulsanti */}
            <div className="flex gap-3 pt-4">
              <button
                type="button"
                onClick={onClose}
                className="flex-1 btn-outline"
              >
                Annulla
              </button>
              <button
                type="submit"
                className="flex-1 btn-primary"
              >
                {appointment ? 'Aggiorna' : 'Crea'} Prenotazione
              </button>
            </div>
          </form>
        </div>
      </div>
    </div>
  );
};

export default AppointmentForm;
```

🎛️ Dashboard {#dashboard}

Dashboard Component

src/components/Dashboard.tsx:

```
import React, { useState } from 'react';
import { Plus, Calendar, Users, Clock, TrendingUp } from 'lucide-react';
import type { Appointment, Treatment, ViewMode } from '../types/index';
import ViewSelector from './ViewSelector';
import DateNavigator from './DateNavigator';
import DayView from './DayView';
import WeekView from './WeekView';
import MonthView from './MonthView';
import AppointmentForm from './AppointmentForm';
```

```
// Dati di esempio per i trattamenti
const sampleTreatments: Treatment[] = [
  {
    id: '1',
    name: 'Massaggio Rilassante',
    duration: 60,
    price: 80,
    category: 'massage'
  },
  // ... altri trattamenti
];

// Dati di esempio per le prenotazioni
const sampleAppointments: Appointment[] = [
  {
    id: '1',
    clientName: 'Maria Rossi',
    clientPhone: '+39 123 456 789',
    clientEmail: 'maria.rossi@email.com',
    treatmentId: '1',
    treatment: sampleTreatments[0],
    date: new Date().toISOString().split('T')[0],
    startTime: '10:00',
    endTime: '11:00',
    notes: 'Cliente preferisce olio essenziale alla lavanda',
    status: 'confirmed',
    createdAt: new Date(),
    updatedAt: new Date()
  }
  // ... altre prenotazioni
];

const Dashboard: React.FC = () => {
  const [currentView, setCurrentView] = useState<ViewMode>('day');
  const [currentDate, setCurrentDate] = useState(new Date());
  const [appointments, setAppointments] = useState<Appointment[]>
(sampleAppointments);
  const [treatments] = useState<Treatment[]>(sampleTreatments);
  const [isFormOpen, setIsFormOpen] = useState(false);
  const [editingAppointment, setEditingAppointment] = useState<Appointment
| undefined>();
  const [selectedDate, setSelectedDate] = useState<string>('');
  const [selectedTime, setSelectedTime] = useState<string>('');

  // Statistiche
  const todayAppointments = appointments.filter(apt => apt.date === new
Date().toISOString().split('T')[0]);
  const totalAppointments = appointments.length;
  const confirmedAppointments = appointments.filter(apt => apt.status ===
'confirmed').length;

  const handleAddAppointment = (date: string, time: string) => {
    setSelectedDate(date);
```

```
      setSelectedTime(time);
      setEditingAppointment(undefined);
      setIsFormOpen(true);
    };

    const handleEditAppointment = (appointment: Appointment) => {
      setEditingAppointment(appointment);
      setSelectedDate('');
      setSelectedTime('');
      setIsFormOpen(true);
    };

    const handleDeleteAppointment = (id: string) => {
      if (window.confirm('Sei sicuro di voler cancellare questa
  prenotazione?')) {
        setAppointments(prev => prev.filter(apt => apt.id !== id));
      }
    };

    const handleSubmitAppointment = (appointmentData: Omit<Appointment, 'id'
  | 'createdAt' | 'updatedAt'>) => {
      if (editingAppointment) {
        // Modifica prenotazione esistente
        setAppointments(prev => prev.map(apt =>
          apt.id === editingAppointment.id
            ? { ...appointmentData, id: apt.id, createdAt: apt.createdAt,
  updatedAt: new Date() }
            : apt
        ));
      } else {
        // Nuova prenotazione
        const newAppointment: Appointment = {
          ...appointmentData,
          id: Date.now().toString(),
          createdAt: new Date(),
          updatedAt: new Date()
        };
        setAppointments(prev => [...prev, newAppointment]);
      }
    };

    const getWeekStartDate = (date: Date) => {
      const day = date.getDay();
      const diff = date.getDate() - day + (day === 0 ? -6 : 1);
      return new Date(date.setDate(diff));
    };

    return (
      <div className="min-h-screen bg-gray-50">
        {/* Header della dashboard */}
        <div className="bg-white shadow-sm border-b border-gray-200">
          <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-6">
            <div className="flex flex-col sm:flex-row sm:items-center
  sm:justify-between gap-4">
```

```
              <div>
                <h1 className="text-2xl font-bold text-gray-900">
                  Dashboard Prenotazioni
                </h1>
                <p className="text-gray-600 mt-1">
                  Gestisci le prenotazioni del centro benessere
                </p>
              </div>

              <button
                onClick={() => {
                  setEditingAppointment(undefined);
                  setSelectedDate('');
                  setSelectedTime('');
                  setIsFormOpen(true);
                }}
                className="btn-primary flex items-center gap-2"
              >
                <Plus className="w-4 h-4" />
                Nuova Prenotazione
              </button>
            </div>
          </div>
        </div>

        {/* Statistiche */}
        <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-6">
          <div className="grid grid-cols-1 md:grid-cols-4 gap-6">
            <div className="card">
              <div className="flex items-center">
                <div className="p-2 bg-indigo-100 rounded-lg">
                  <Calendar className="w-6 h-6 text-indigo-600" />
                </div>
                <div className="ml-4">
                  <p className="text-sm font-medium text-gray-600">Oggi</p>
                  <p className="text-2xl font-semibold text-gray-900">
{todayAppointments.length}</p>
                </div>
              </div>
            </div>

            <div className="card">
              <div className="flex items-center">
                <div className="p-2 bg-fuchsia-100 rounded-lg">
                  <Users className="w-6 h-6 text-fuchsia-600" />
                </div>
                <div className="ml-4">
                  <p className="text-sm font-medium text-gray-
600">Totali</p>
                  <p className="text-2xl font-semibold text-gray-900">
{totalAppointments}</p>
                </div>
              </div>
            </div>
```

```
          <div className="card">
            <div className="flex items-center">
              <div className="p-2 bg-green-100 rounded-lg">
                <Clock className="w-6 h-6 text-green-600" />
              </div>
              <div className="ml-4">
                <p className="text-sm font-medium text-gray-
600">Confermate</p>
                <p className="text-2xl font-semibold text-gray-900">
{confirmedAppointments}</p>
              </div>
            </div>
          </div>

          <div className="card">
            <div className="flex items-center">
              <div className="p-2 bg-purple-100 rounded-lg">
                <TrendingUp className="w-6 h-6 text-purple-600" />
              </div>
              <div className="ml-4">
                <p className="text-sm font-medium text-gray-
600">Efficienza</p>
                <p className="text-2xl font-semibold text-gray-900">
                  {totalAppointments > 0 ?
Math.round((confirmedAppointments / totalAppointments) * 100) : 0}%
                </p>
              </div>
            </div>
          </div>
        </div>
      </div>

      {/* Controlli di navigazione */}
      <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-6">
        <div className="flex flex-col lg:flex-row lg:items-center
lg:justify-between gap-4">
          <ViewSelector currentView={currentView} onViewChange=
{setCurrentView} />
          <DateNavigator
            currentDate={currentDate}
            onDateChange={setCurrentDate}
            viewMode={currentView}
          />
        </div>
      </div>

      {/* Contenuto principale */}
      <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 pb-8">
        {currentView === 'day' && (
          <DayView
            date={currentDate.toISOString().split('T')[0]}
            appointments={appointments}
            onAddAppointment={handleAddAppointment}
```

```
            onEditAppointment={handleEditAppointment}
            onDeleteAppointment={handleDeleteAppointment}
          />
        )}

        {currentView === 'week' && (
          <WeekView
            startDate={getWeekStartDate(currentDate)}
            appointments={appointments}
            onAddAppointment={handleAddAppointment}
            onEditAppointment={handleEditAppointment}
            onDeleteAppointment={handleDeleteAppointment}
          />
        )}

        {currentView === 'month' && (
          <MonthView
            currentDate={currentDate}
            appointments={appointments}
            onAddAppointment={handleAddAppointment}
            onEditAppointment={handleEditAppointment}
            onMonthChange={setCurrentDate}
          />
        )}
      </div>

      {/* Form per le prenotazioni */}
      <AppointmentForm
        isOpen={isFormOpen}
        onClose={() => setIsFormOpen(false)}
        onSubmit={handleSubmitAppointment}
        appointment={editingAppointment}
        treatments={treatments}
        selectedDate={selectedDate}
        selectedTime={selectedTime}
      />
    </div>
  );
};

export default Dashboard;
```

🚀 Esecuzione {#esecuzione}

1. Verificare Setup

```
# Il template Vite include già tutto il necessario
# Verifica che le dipendenze siano installate
npm list react react-dom typescript vite
```

2. Avviare Applicazione

```
npm run dev
```

3. Verificare Funzionamento

- ✅ Server avviato su http://localhost:5173
- ✅ Hot reload attivo
- ✅ TypeScript compilation
- ✅ Tailwind CSS funzionante

4. Testare

- **Accesso**: `admin/admin`
- **Navigazione**: Testa viste giorno/settimana/mese
- **CRUD**: Crea/modifica/elimina prenotazioni
- **Responsive**: Testa su mobile/desktop

🧠 Concetti Chiave {#concetti}

1. **Setup Vite**

- Template preconfigurato React + TypeScript
- Configurazione automatica build tools
- Hot reload e sviluppo veloce

2. **Architettura Component-Based**

- Separazione responsabilità
- Props drilling e state lifting
- Componenti riutilizzabili

3. **Gestione Stato**

- `useState` per stato locale
- Controlled components
- Gestione stato condiviso

4. **TypeScript in React**

- Interfacce per props/state
- Type safety
- Generics riutilizzabili

5. **Tailwind CSS 4**

- Utility-first CSS

- Responsive design
- Custom components

6. **Gestione Eventi**

- Form handling con validazione
- Event bubbling
- Gestione asincrona

7. **Algoritmi e Logica**

- Calcolo date/orari
- Verifica sovrapposizioni
- Filtri e statistiche

🎉 Conclusione

Hai ricreato **ArmoniaWellness**, un'app React completa e professionale!

Competenze acquisite:

- ✅ Setup Vite e template preconfigurati
- ✅ Architettura React moderna
- ✅ TypeScript e type safety
- ✅ Tailwind CSS e responsive design
- ✅ Gestione stato complessa
- ✅ UI/UX professionale
- ✅ Best practices React

Prossimi passi:

- Persistenza dati (localStorage/API)
- Autenticazione reale (JWT/OAuth)
- Testing (Jest/React Testing Library)
- Deployment (Vercel/Netlify)

L'app è pronta per uso professionale e può servire come portfolio per dimostrare le tue competenze React!

📚 Note Aggiuntive

Vantaggi del Setup Vite

- 🚀 **Velocità**: Hot reload istantaneo
- ⚙️ **Configurazione**: Zero config per React + TypeScript
- 🏗️ **Build**: Ottimizzazione automatica per produzione
- 🔧 **DevTools**: Integrazione con browser DevTools
- 📦 **Bundle**: Tree shaking e code splitting automatici

Template Inclusi

Il comando `npm create vite@latest` offre template per:

- React + TypeScript (quello utilizzato)
- React + JavaScript
- Vue + TypeScript/JavaScript
- Svelte + TypeScript/JavaScript
- Vanilla + TypeScript/JavaScript