Análisis de Mejoras de Usabilidad y Desarrollo para INMOVA

Elevando la App a un Nivel de 10/10

📊 Resumen Ejecutivo

INMOVA es una aplicación robusta y completa con **88 módulos profesionales** y una arquitectura sólida. Sin embargo, hay **mejoras horizontales clave** que la elevarían de un **8/10 a un 10/10** en términos de calidad enterprise.

Puntuación Actual por Área

Área	Puntuación	Potencial
Funcionalidad Vertical	9/10	10/10
UX/UI	7/10	10/10
Rendimiento	6/10	10/10
Accesibilidad	4/10	10/10
Testing	1/10	10/10
Seguridad	7/10	10/10
DevOps/CI-CD	5/10	10/10
Documentación Técnica	6/10	10/10

Puntuación General Actual: 7.5/10
Puntuación Objetivo: 10/10

🎨 1. MEJORAS DE UX/UI (Prioridad: ALTA)

1.1 Sistema de Design System Completo

Problema Actual:
- Colores y estilos dispersos en múltiples archivos
- Falta de guía de estilo centralizada
- Inconsistencias visuales entre módulos

Solución Recomendada:

```
// lib/design-system/tokens.ts
export const designTokens = {
  colors: {
    brand: {
      primary: {
        50: 'hsl(235, 89%, 97%)',
        100: 'hsl(235, 89%, 93%)',
        // ... hasta 900
      },
      // Definir TODOS los colores semánticos
    },
    semantic: {
      success: { light: '...', DEFAULT: '...', dark: '...' },
      error: { light: '...', DEFAULT: '...', dark: '...' },
      warning: { light: '...', DEFAULT: '...', dark: '...' },
      info: { light: '...', DEFAULT: '...', dark: '...' },
    },
  },
  spacing: {
    xs: '0.25rem',
    sm: '0.5rem',
    md: '1rem',
    lg: '1.5rem',
    xl: '2rem',
    // ...
  },
  typography: {
    fontFamily: {
      sans: ['Inter', 'system-ui', '-apple-system', 'sans-serif'],
      heading: ['Poppins', 'sans-serif'],
      mono: ['JetBrains Mono', 'monospace'],
    },
    fontSize: {
      xs: ['0.75rem', { lineHeight: '1rem' }],
      sm: ['0.875rem', { lineHeight: '1.25rem' }],
      // ...
    },
  },
  shadows: {
    sm: '0 1px 2px 0 rgba(0, 0, 0, 0.05)',
    // ...
    primary: '0 10px 40px -10px rgba(79, 70, 229, 0.4)',
  },
  transitions: {
    fast: '150ms cubic-bezier(0.4, 0, 0.2, 1)',
    base: '200ms cubic-bezier(0.4, 0, 0.2, 1)',
    slow: '300ms cubic-bezier(0.4, 0, 0.2, 1)',
  },
};

// components/ui/design-system-provider.tsx
export function DesignSystemProvider({ children }: { children: ReactNode }) {
  // Aplicar tokens vía CSS variables
  useEffect(() => {
    const root = document.documentElement;
    Object.entries(designTokens.colors.brand.primary).forEach(([key, value]) => {
      root.style.setProperty(`--color-primary-${key}`, value);
    });
  }, []);

  return <>{children}</>;
}
```

Impacto: +1.5 puntos en UX/UI

1.2 Micro-interacciones y Feedback Visual

Implementar:

```tsx
// components/ui/interactive-button.tsx
export function InteractiveButton({ children, ...props }: ButtonProps) {
  const [isPressed, setIsPressed] = useState(false);

  return (
    <motion.button
      whileHover={{ scale: 1.02 }}
      whileTap={{ scale: 0.98 }}
      transition={{ type: 'spring', stiffness: 400, damping: 17 }}
      onMouseDown={() => setIsPressed(true)}
      onMouseUp={() => setIsPressed(false)}
      className={cn(
        'relative overflow-hidden',
        isPressed && 'shadow-inner'
      )}
      {...props}
    >
      {/* Ripple effect */}
      <span className="absolute inset-0 animate-ripple" />
      {children}
    </motion.button>
  );
}
```

Áreas de aplicación:
- Botones de acción primaria
- Cards interactivos
- Inputs con feedback visual
- Notificaciones con animaciones suaves

Impacto: +0.5 puntos en UX/UI

1.3 Sistema de Notificaciones Mejorado

Problema: Toast notifications simples sin persistencia ni acciones.

Solución:

```typescript
// lib/notification-system.ts
import { toast } from 'sonner';

interface NotificationOptions {
  type: 'success' | 'error' | 'warning' | 'info';
  title: string;
  description?: string;
  action?: {
    label: string;
    onClick: () => void;
  };
  persistent?: boolean;
  undoable?: boolean;
  onUndo?: () => void;
}

export function showNotification(options: NotificationOptions) {
  const duration = options.persistent ? Infinity : 5000;

  return toast[options.type](options.title, {
    description: options.description,
    duration,
    action: options.action && {
      label: options.action.label,
      onClick: options.action.onClick,
    },
    cancel: options.undoable && {
      label: 'Deshacer',
      onClick: options.onUndo,
    },
    closeButton: options.persistent,
  });
}

// Uso:
showNotification({
  type: 'success',
  title: 'Edificio eliminado',
  description: 'Se ha eliminado "Torre Vista"',
  undoable: true,
  onUndo: async () => {
    await restoreBuilding(buildingId);
  },
});
```

Impacto: +0.5 puntos en UX/UI

1.4 Skeleton Screens Inteligentes

Problema: Loading states genéricos en todas las páginas.

Solución: Ya tienes `SkeletonCard`, pero necesitas variantes específicas:

```
// components/ui/skeleton-variants.tsx
export function TableSkeleton({ rows = 5 }) {
  return (
    <div className="space-y-2">
      {Array.from({ length: rows }).map((_, i) => (
        <div key={i} className="flex gap-4">
          <Skeleton className="h-12 w-12 rounded-full" />
          <div className="flex-1 space-y-2">
            <Skeleton className="h-4 w-3/4" />
            <Skeleton className="h-3 w-1/2" />
          </div>
          <Skeleton className="h-8 w-24" />
        </div>
      ))}
    </div>
  );
}

export function DashboardSkeleton() {
  return (
    <div className="grid gap-6 md:grid-cols-2 lg:grid-cols-4">
      {Array.from({ length: 4 }).map((_, i) => (
        <Card key={i}>
          <CardHeader>
            <Skeleton className="h-4 w-24" />
          </CardHeader>
          <CardContent>
            <Skeleton className="h-8 w-16 mb-2" />
            <Skeleton className="h-3 w-32" />
          </CardContent>
        </Card>
      ))}
    </div>
  );
}
```

Impacto: +0.3 puntos en UX/UI

1.5 Modo Offline con Service Worker Robusto

Problema: PWA básico sin estrategias de caching sofisticadas.

Solución:

```
// public/sw-advanced.js
const CACHE_VERSION = 'v1.2.0';
const STATIC_CACHE = `inmova-static-${CACHE_VERSION}`;
const DYNAMIC_CACHE = `inmova-dynamic-${CACHE_VERSION}`;
const IMAGE_CACHE = `inmova-images-${CACHE_VERSION}`;

// Estrategia: Network First con fallback a cache
const networkFirst = async (request) => {
  try {
    const response = await fetch(request);
    const cache = await caches.open(DYNAMIC_CACHE);
    cache.put(request, response.clone());
    return response;
  } catch (error) {
    const cachedResponse = await caches.match(request);
    return cachedResponse || new Response('Offline', { status: 503 });
  }
};

// Estrategia: Cache First con revalidación en background
const cacheFirst = async (request) => {
  const cachedResponse = await caches.match(request);

  if (cachedResponse) {
    // Revalidate in background
    fetch(request).then((response) => {
      const cache = await caches.open(STATIC_CACHE);
      cache.put(request, response);
    });
    return cachedResponse;
  }

  return fetch(request);
};

self.addEventListener('fetch', (event) => {
  const { request } = event;
  const url = new URL(request.url);

  // API calls: Network First
  if (url.pathname.startsWith('/api/')) {
    event.respondWith(networkFirst(request));
  }
  // Static assets: Cache First
  else if (request.destination === 'image') {
    event.respondWith(cacheFirst(request));
  }
  // Default: Network First
  else {
    event.respondWith(networkFirst(request));
  }
});

// Background sync para operaciones offline
self.addEventListener('sync', (event) => {
  if (event.tag === 'sync-offline-actions') {
    event.waitUntil(syncOfflineActions());
  }
});

async function syncOfflineActions() {
  const db = await openDB('inmova-offline', 1);
  const actions = await db.getAll('pending-actions');

  for (const action of actions) {
    try {
      await fetch(action.url, {
        method: action.method,
        body: JSON.stringify(action.data),
        headers: { 'Content-Type': 'application/json' },
      });
      await db.delete('pending-actions', action.id);
    } catch (error) {
      console.error('Failed to sync action:', action, error);
    }
  }
}
```

Impacto: +0.7 puntos en UX/UI

1.6 Búsqueda Global Mejorada

Problema: Búsqueda básica sin highlighting ni sugerencias.

Solución:

```
// components/ui/advanced-search.tsx
import { Command } from 'cmdk';
import Fuse from 'fuse.js';

export function AdvancedGlobalSearch() {
  const [open, setOpen] = useState(false);
  const [query, setQuery] = useState('');
  const [results, setResults] = useState<SearchResult[]>([]);
  const router = useRouter();

  // Keyboard shortcut: Cmd+K / Ctrl+K
  useEffect(() => {
    const down = (e: KeyboardEvent) => {
      if (e.key === 'k' && (e.metaKey || e.ctrlKey)) {
        e.preventDefault();
        setOpen((open) => !open);
      }
    };
    document.addEventListener('keydown', down);
    return () => document.removeEventListener('keydown', down);
  }, []);

  // Fuzzy search con Fuse.js
  const fuse = useMemo(
    () =>
      new Fuse(searchableItems, {
        keys: ['title', 'description', 'keywords'],
        threshold: 0.3,
        includeScore: true,
        includeMatches: true,
      }),
    [searchableItems]
  );

  useEffect(() => {
    if (query.length > 0) {
      const fuseResults = fuse.search(query);
      setResults(fuseResults.map((r) => r.item));
    } else {
      setResults([]);
    }
  }, [query, fuse]);

  return (
    <Command.Dialog open={open} onOpenChange={setOpen}>
      <Command.Input
        value={query}
        onValueChange={setQuery}
        placeholder="Buscar edificios, inquilinos, contratos..."
      />
      <Command.List>
        {results.map((result) => (
          <Command.Item
            key={result.id}
            onSelect={() => {
              router.push(result.url);
              setOpen(false);
            }}
          >
            <result.icon className="mr-2 h-4 w-4" />
            <div className="flex-1">
              <div className="font-medium">
                <HighlightedText text={result.title} query={query} />
              </div>
              <div className="text-sm text-muted-foreground">
                {result.description}
              </div>
            </div>
            <Badge variant="outline">{result.type}</Badge>
          </Command.Item>
        ))}
      </Command.List>
    </Command.Dialog>
  );
}

function HighlightedText({ text, query }: { text: string; query: string }) {
  // Highlight matching portions
  const parts = text.split(new RegExp(`(${query})`, 'gi'));
  return (
    <span>
      {parts.map((part, i) =>
        part.toLowerCase() === query.toLowerCase() ? (
          <mark key={i} className="bg-yellow-200 text-yellow-900">
            {part}
          </mark>
        ) : (
          part
        )
      )}
    </span>
  );
}
```

Impacto: +0.5 puntos en UX/UI

⚡ 2. MEJORAS DE RENDIMIENTO (Prioridad: ALTA)

2.1 Lazy Loading de Componentes Pesados

Problema: Componentes como charts y calendarios se cargan siempre.

Solución:

```
// app/dashboard/page.tsx
import dynamic from 'next/dynamic';

// Lazy load componentes pesados
const AdvancedAnalytics = dynamic(
  () => import('@/components/dashboard/AdvancedAnalytics'),
  {
    loading: () => <SkeletonChart />,
    ssr: false, // Si no necesita SSR
  }
);

const BigCalendar = dynamic(
  () => import('react-big-calendar').then((mod) => mod.Calendar),
  {
    loading: () => <SkeletonCalendar />,
    ssr: false,
  }
);

const Plotly = dynamic(() => import('react-plotly.js'), {
  loading: () => <div>Cargando gráfico...</div>,
  ssr: false,
});
```

Impacto: -30% en tamaño del bundle inicial, +1.5 puntos en Rendimiento

2.2 Implementar React Query para Cache Inteligente

Problema: Múltiples fetches de los mismos datos sin cache.

Solución:

```typescript
// lib/api/use-buildings.ts
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query';

export function useBuildings(companyId: string) {
  return useQuery({
    queryKey: ['buildings', companyId],
    queryFn: () => fetchBuildings(companyId),
    staleTime: 5 * 60 * 1000, // 5 minutos
    cacheTime: 30 * 60 * 1000, // 30 minutos
    refetchOnWindowFocus: true,
  });
}

export function useCreateBuilding() {
  const queryClient = useQueryClient();

  return useMutation({
    mutationFn: createBuilding,
    onSuccess: (newBuilding) => {
      // Optimistic update
      queryClient.setQueryData(
        ['buildings', newBuilding.companyId],
        (old: Building[] = []) => [...old, newBuilding]
      );

      // Invalidar queries relacionadas
      queryClient.invalidateQueries({ queryKey: ['dashboard-stats'] });
    },
  });
}

// Prefetching para navegación rápida
export function usePrefetchBuilding(buildingId: string) {
  const queryClient = useQueryClient();

  return () => {
    queryClient.prefetchQuery({
      queryKey: ['building', buildingId],
      queryFn: () => fetchBuilding(buildingId),
    });
  };
}
```

Uso:

```
// app/edificios/page.tsx
export default function EdificiosPage() {
  const { data: buildings, isLoading, error } = useBuildings(companyId);
  const createBuilding = useCreateBuilding();
  const prefetchBuilding = usePrefetchBuilding();

  return (
    <div>
      {buildings?.map((building) => (
        <Card
          key={building.id}
          onMouseEnter={() => prefetchBuilding(building.id)()}
        >
          {/* ... */}
        </Card>
      ))}
    </div>
  );
}
```

Impacto: -60% en llamadas API redundantes, +1.0 puntos en Rendimiento

2.3 Optimización de Imágenes con Next/Image

Problema: `images: { unoptimized: true }` en next.config.js

Solución:

```
// next.config.js
module.exports = {
  images: {
    unoptimized: false, // ¡Activar optimización!
    formats: ['image/avif', 'image/webp'],
    deviceSizes: [640, 750, 828, 1080, 1200, 1920, 2048, 3840],
    imageSizes: [16, 32, 48, 64, 96, 128, 256, 384],
    domains: [
      'inmova-assets.s3.amazonaws.com',
      'inmova-cdn.cloudfront.net',
    ],
    loader: 'default',
    minimumCacheTTL: 60 * 60 * 24 * 30, // 30 días
  },
};
```

```
// components/optimized-image.tsx
import Image from 'next/image';
import { useState } from 'react';

export function OptimizedImage({
  src,
  alt,
  priority = false,
  ...props
}: ImageProps) {
  const [isLoading, setIsLoading] = useState(true);

  return (
    <div className="relative overflow-hidden bg-gray-100">
      {isLoading && (
        <div className="absolute inset-0 animate-pulse bg-gray-200" />
      )}
      <Image
        src={src}
        alt={alt}
        priority={priority}
        quality={85}
        placeholder="blur"
        blurDataURL="data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAAQABAAD/..."
        onLoadingComplete={() => setIsLoading(false)}
        {...props}
      />
    </div>
  );
}
```

Impacto: -70% en peso de imágenes, +1.0 puntos en Rendimiento

2.4 Code Splitting por Rutas

Problema: Todo el código se carga inicialmente.

Solución:

```
// app/layout.tsx
import dynamic from 'next/dynamic';

// Módulos admin solo para admins
const AdminRoutes = dynamic(() => import('./admin/layout'), {
  loading: () => <LoadingAdminLayout />,
});

// Módulos multi-vertical solo cuando se necesiten
const STRRoutes = dynamic(() => import('./str/layout'));
const FlippingRoutes = dynamic(() => import('./flipping/layout'));
const ConstructionRoutes = dynamic(() => import('./construction/layout'));
```

Impacto: -40% en JavaScript inicial, +0.8 puntos en Rendimiento

2.5 Virtualización de Listas Largas

Problema: Renderizar 1000+ edificios/inquilinos causa lag.

Solución:

```
// components/ui/virtualized-list.tsx
import { useVirtualizer } from '@tanstack/react-virtual';
import { useRef } from 'react';

export function VirtualizedBuildingList({ buildings }: { buildings: Building[] }) {
  const parentRef = useRef<HTMLDivElement>(null);

  const virtualizer = useVirtualizer({
    count: buildings.length,
    getScrollElement: () => parentRef.current,
    estimateSize: () => 100, // altura estimada de cada item
    overscan: 5, // items extra fuera del viewport
  });

  return (
    <div ref={parentRef} className="h-screen overflow-auto">
      <div
        style={{
          height: `${virtualizer.getTotalSize()}px`,
          width: '100%',
          position: 'relative',
        }}
      >
        {virtualizer.getVirtualItems().map((virtualItem) => {
          const building = buildings[virtualItem.index];
          return (
            <div
              key={virtualItem.key}
              style={{
                position: 'absolute',
                top: 0,
                left: 0,
                width: '100%',
                height: `${virtualItem.size}px`,
                transform: `translateY(${virtualItem.start}px)`,
              }}
            >
              <BuildingCard building={building} />
            </div>
          );
        })}
      </div>
    </div>
  );
}
```

Instalar:

```
yarn add @tanstack/react-virtual
```

Impacto: Renderizar 10,000 items sin lag, +0.5 puntos en Rendimiento

2.6 Optimización de Bundle con Webpack Analyzer

Solución:

```
yarn add -D @next/bundle-analyzer
```

```
// next.config.js
const withBundleAnalyzer = require('@next/bundle-analyzer')({
  enabled: process.env.ANALYZE === 'true',
});

module.exports = withBundleAnalyzer({
  // ... rest of config
  webpack: (config, { isServer }) => {
    // Eliminar librerías duplicadas
    if (!isServer) {
      config.resolve.alias = {
        ...config.resolve.alias,
        'date-fns': require.resolve('date-fns'),
        'lodash': require.resolve('lodash-es'), // usar ES modules
      };
    }

    // Tree shaking agresivo
    config.optimization = {
      ...config.optimization,
      usedExports: true,
      sideEffects: false,
    };

    return config;
  },
});
```

Analizar:

```
ANALYZE=true yarn build
```

Impacto: Identificar y eliminar código muerto, +0.4 puntos en Rendimiento

---

♿ 3. MEJORAS DE ACCESIBILIDAD (Prioridad: MEDIA-ALTA)

3.1 Navegación por Teclado Completa

Problema: Muchos componentes no son navegables con teclado.

Solución:

```
// components/ui/accessible-card.tsx
export function AccessibleCard({
  onClick,
  children,
  ...props
}: CardProps & { onClick?: () => void }) {
  return (
    <Card
      role="button"
      tabIndex={0}
      onKeyDown={(e) => {
        if (e.key === 'Enter' || e.key === ' ') {
          e.preventDefault();
          onClick?.();
        }
      }}
      onClick={onClick}
      className="focus-visible:ring-2 focus-visible:ring-primary"
      {...props}
    >
      {children}
    </Card>
  );
}
```

Aplicar a:
- Cards de edificios/unidades/inquilinos
- Botones de acción
- Dropdowns y menus
- Modales y dialogs

Impacto: +2.0 puntos en Accesibilidad

3.2 Etiquetas ARIA y Roles Semánticos

Solución:

```
// components/layout/sidebar.tsx (mejorado)
export function Sidebar() {
  return (
    <nav
      role="navigation"
      aria-label="Navegación principal"
      className="sidebar"
    >
      <ul role="list">
        {menuItems.map((item) => (
          <li key={item.id}>
            <Link
              href={item.href}
              aria-current={isActive ? 'page' : undefined}
              aria-label={`Ir a ${item.label}`}
            >
              <item.icon aria-hidden="true" />
              <span>{item.label}</span>
            </Link>
          </li>
        ))}
      </ul>
    </nav>
  );
}

// components/ui/kpi-card.tsx (mejorado)
export function KPICard({ title, value, trend, icon: Icon }) {
  return (
    <Card
      role="article"
      aria-label={`KPI: ${title}`}
    >
      <CardHeader>
        <CardTitle>
          <Icon aria-hidden="true" />
          {title}
        </CardTitle>
      </CardHeader>
      <CardContent>
        <div
          className="text-3xl font-bold"
          aria-live="polite"
          aria-atomic="true"
        >
          {value}
        </div>
        {trend && (
          <div
            className={cn(
              'text-sm',
              trend > 0 ? 'text-green-600' : 'text-red-600'
            )}
            role="status"
            aria-label={`Tendencia: ${trend > 0 ? 'positiva' : 'negativa'} de ${Math.a
bs(trend)}%`}
          >
            {trend > 0 ? '+' : ''}{trend}%
          </div>
        )}
      </CardContent>
    </Card>
  );
}
```

Impacto: +1.5 puntos en Accesibilidad

3.3 Modo Alto Contraste

Solución:

```
// lib/accessibility/high-contrast.ts
export function useHighContrast() {
  const [isHighContrast, setIsHighContrast] = useState(false);

  useEffect(() => {
    const mediaQuery = window.matchMedia('(prefers-contrast: high)');
    setIsHighContrast(mediaQuery.matches);

    const handler = (e: MediaQueryListEvent) => {
      setIsHighContrast(e.matches);
    };

    mediaQuery.addEventListener('change', handler);
    return () => mediaQuery.removeEventListener('change', handler);
  }, []);

  useEffect(() => {
    if (isHighContrast) {
      document.documentElement.classList.add('high-contrast');
    } else {
      document.documentElement.classList.remove('high-contrast');
    }
  }, [isHighContrast]);

  return { isHighContrast, toggleHighContrast: () => setIsHighContrast(!isHighCon-
trast) };
}
```

```
/* app/globals.css */
.high-contrast {
  --primary: hsl(220, 100%, 30%);
  --secondary: hsl(220, 100%, 20%);
  --background: hsl(0, 0%, 100%);
  --foreground: hsl(0, 0%, 0%);
  --border: hsl(0, 0%, 0%);
  --muted: hsl(220, 10%, 90%);

  /* Aumentar contraste de todos los elementos */
  * {
    text-shadow: none !important;
    box-shadow: 0 0 0 1px black !important;
  }
}
```

Impacto: +0.8 puntos en Accesibilidad

3.4 Screen Reader Support Completo

Solución:

```
// components/ui/live-region.tsx
export function LiveRegion({ children, priority = 'polite' }: {
  children: ReactNode;
  priority?: 'polite' | 'assertive';
}) {
  return (
    <div
      role="status"
      aria-live={priority}
      aria-atomic="true"
      className="sr-only"
    >
      {children}
    </div>
  );
}

// Uso:
export function BuildingsList() {
  const { data: buildings, isLoading } = useBuildings();

  return (
    <>
      <LiveRegion>
        {isLoading
          ? 'Cargando edificios...'
          : `${buildings.length} edificios cargados`}
      </LiveRegion>

      <div role="list" aria-label="Lista de edificios">
        {buildings.map((building) => (
          <div key={building.id} role="listitem">
            <AccessibleCard>{/* ... */}</AccessibleCard>
          </div>
        ))}
      </div>
    </>
  );
}
```

Impacto: +1.2 puntos en Accesibilidad

3.5 Focus Management

Solución:

```
// hooks/use-focus-trap.ts
import { useEffect, useRef } from 'react';

export function useFocusTrap(isActive: boolean) {
  const containerRef = useRef<HTMLDivElement>(null);

  useEffect(() => {
    if (!isActive || !containerRef.current) return;

    const container = containerRef.current;
    const focusableElements = container.querySelectorAll(
      'button, [href], input, select, textarea, [tabindex]:not([tabindex="-1"])'
    );

    const firstElement = focusableElements[0] as HTMLElement;
    const lastElement = focusableElements[focusableElements.length - 1] as HTMLEle-
ment;

    const handleTabKey = (e: KeyboardEvent) => {
      if (e.key !== 'Tab') return;

      if (e.shiftKey) {
        if (document.activeElement === firstElement) {
          e.preventDefault();
          lastElement.focus();
        }
      } else {
        if (document.activeElement === lastElement) {
          e.preventDefault();
          firstElement.focus();
        }
      }
    };

    container.addEventListener('keydown', handleTabKey);
    firstElement?.focus();

    return () => {
      container.removeEventListener('keydown', handleTabKey);
    };
  }, [isActive]);

  return containerRef;
}

// Uso en Dialog:
export function AccessibleDialog({ open, onOpenChange, children }) {
  const containerRef = useFocusTrap(open);

  return (
    <Dialog open={open} onOpenChange={onOpenChange}>
      <DialogContent ref={containerRef}>
        {children}
      </DialogContent>
    </Dialog>
  );
}
```

Impacto: +0.5 puntos en Accesibilidad

🧪 4. TESTING (Prioridad: ALTA)

4.1 Setup de Testing con Vitest

Problema: 0 tests en todo el proyecto.

Solución:

```
yarn add -D vitest @testing-library/react @testing-library/jest-dom @testing-library/
user-event jsdom @vitejs/plugin-react
```

```
// vitest.config.ts
import { defineConfig } from 'vitest/config';
import react from '@vitejs/plugin-react';
import path from 'path';

export default defineConfig({
  plugins: [react()],
  test: {
    environment: 'jsdom',
    setupFiles: ['./vitest.setup.ts'],
    globals: true,
    coverage: {
      provider: 'v8',
      reporter: ['text', 'json', 'html', 'lcov'],
      exclude: [
        'node_modules/',
        '.next/',
        'coverage/',
        '**/*.config.{js,ts}',
        '**/*.d.ts',
      ],
    },
  },
  resolve: {
    alias: {
      '@': path.resolve(__dirname, './'),
    },
  },
});

// vitest.setup.ts
import '@testing-library/jest-dom';
import { cleanup } from '@testing-library/react';
import { afterEach } from 'vitest';

afterEach(() => {
  cleanup();
});
```

```
// package.json (agregar scripts)
{
  "scripts": {
    "test": "vitest",
    "test:ui": "vitest --ui",
    "test:coverage": "vitest --coverage",
    "test:ci": "vitest run --coverage"
  }
}
```

4.2 Tests Unitarios para Componentes

```
// components/ui/kpi-card.test.tsx
import { describe, it, expect } from 'vitest';
import { render, screen } from '@testing-library/react';
import { KPICard } from './kpi-card';
import { DollarSign } from 'lucide-react';

describe('KPICard', () => {
  it('debería renderizar el título y valor', () => {
    render(
      <KPICard
        title="Ingresos Totales"
        value="€15,234"
        icon={DollarSign}
      />
    );

    expect(screen.getByText('Ingresos Totales')).toBeInTheDocument();
    expect(screen.getByText('€15,234')).toBeInTheDocument();
  });

  it('debería mostrar tendencia positiva en verde', () => {
    render(
      <KPICard
        title="Test"
        value="100"
        trend={12.5}
        icon={DollarSign}
      />
    );

    const trendElement = screen.getByText('+12.5%');
    expect(trendElement).toHaveClass('text-green-600');
  });

  it('debería mostrar tendencia negativa en rojo', () => {
    render(
      <KPICard
        title="Test"
        value="100"
        trend={-5.2}
        icon={DollarSign}
      />
    );

    const trendElement = screen.getByText('-5.2%');
    expect(trendElement).toHaveClass('text-red-600');
  });
});
```

4.3 Tests de Integración para Servicios

```
// lib/coupon-service.test.ts
import { describe, it, expect, beforeEach, vi } from 'vitest';
import { validateCoupon, applyCoupon } from './coupon-service';
import { prisma } from './db';

// Mock Prisma
vi.mock('./db', () => ({
  prisma: {
    discountCoupon: {
      findUnique: vi.fn(),
      update: vi.fn(),
    },
    couponUsage: {
      create: vi.fn(),
      count: vi.fn(),
    },
  },
}));

describe('Coupon Service', () => {
  beforeEach(() => {
    vi.clearAllMocks();
  });

  describe('validateCoupon', () => {
    it('debería validar cupón con descuento porcentual', async () => {
      const mockCoupon = {
        id: '1',
        codigo: 'VERANO2025',
        tipo: 'PERCENTAGE',
        valor: 20,
        estado: 'activo',
        activo: true,
        usosMaximos: 100,
        usosActuales: 50,
        fechaInicio: new Date('2025-01-01'),
        fechaExpiracion: new Date('2025-12-31'),
      };

      prisma.discountCoupon.findUnique.mockResolvedValue(mockCoupon);
      prisma.couponUsage.count.mockResolvedValue(0);

      const result = await validateCoupon({
        codigo: 'VERANO2025',
        userId: 'user1',
        montoOriginal: 1000,
        companyId: 'company1',
      });

      expect(result.valido).toBe(true);
      expect(result.montoDescuento).toBe(200); // 20% de 1000
      expect(result.montoFinal).toBe(800);
    });

    it('debería rechazar cupón expirado', async () => {
      const mockCoupon = {
        id: '1',
        codigo: 'EXPIRED',
        estado: 'activo',
        fechaExpiracion: new Date('2024-01-01'),
      };
      prisma.discountCoupon.findUnique.mockResolvedValue(mockCoupon);

      const result = await validateCoupon({
        codigo: 'EXPIRED',
        userId: 'user1',
        montoOriginal: 1000,
        companyId: 'company1',
      });

      expect(result.valido).toBe(false);
      expect(result.error).toContain('expirado');
    });

    it('debería verificar monto mínimo', async () => {
      const mockCoupon = {
        id: '1',
        codigo: 'MIN100',
        estado: 'activo',
        montoMinimo: 100,
        activo: true,
        fechaInicio: new Date('2025-01-01'),
        fechaExpiracion: new Date('2025-12-31'),
      };

      prisma.discountCoupon.findUnique.mockResolvedValue(mockCoupon);

      const result = await validateCoupon({
        codigo: 'MIN100',
        userId: 'user1',
        montoOriginal: 50,
        companyId: 'company1',
      });

      expect(result.valido).toBe(false);
      expect(result.error).toContain('mínimo');
    });
  });
});
```

4.4 Tests E2E con Playwright

```
yarn add -D @playwright/test
npx playwright install
```

```
// e2e/auth.spec.ts
import { test, expect } from '@playwright/test';

test.describe('Authentication', () => {
  test('debería permitir login con credenciales válidas', async ({ page }) => {
    await page.goto('http://localhost:3000/login');

    await page.fill('input[name="email"]', 'admin@inmova.com');
    await page.fill('input[name="password"]', 'admin123');
    await page.click('button[type="submit"]');

    await expect(page).toHaveURL('http://localhost:3000/dashboard');
    await expect(page.locator('h1')).toContainText('Dashboard');
  });

  test('debería mostrar error con credenciales inválidas', async ({ page }) => {
    await page.goto('http://localhost:3000/login');

    await page.fill('input[name="email"]', 'invalid@test.com');
    await page.fill('input[name="password"]', 'wrong');
    await page.click('button[type="submit"]');

    await expect(page.locator('[role="alert"]')).toContainText('Credenciales inváli-
das');
  });
});

// e2e/buildings.spec.ts
test.describe('Gestión de Edificios', () => {
  test.beforeEach(async ({ page }) => {
    // Login automático
    await page.goto('http://localhost:3000/login');
    await page.fill('input[name="email"]', 'admin@inmova.com');
    await page.fill('input[name="password"]', 'admin123');
    await page.click('button[type="submit"]');
    await page.waitForURL('**/dashboard');
  });

  test('debería crear un nuevo edificio', async ({ page }) => {
    await page.goto('http://localhost:3000/edificios/nuevo');

    await page.fill('input[name="nombre"]', 'Torre Prueba E2E');
    await page.fill('input[name="direccion"]', 'Calle Test 123');
    await page.fill('input[name="numeroUnidades"]', '20');
    await page.click('button[type="submit"]');

    await expect(page).toHaveURL(/\/edificios\/[a-z0-9-]+/);
    await expect(page.locator('h1')).toContainText('Torre Prueba E2E');
  });

  test('debería validar campos requeridos', async ({ page }) => {
    await page.goto('http://localhost:3000/edificios/nuevo');

    await page.click('button[type="submit"]');

    // Verificar que se muestran errores de validación
    const errors = page.locator('.text-red-600');
    await expect(errors).toHaveCount(3); // nombre, dirección, unidades
  });
});
```

```
// playwright.config.ts
import { defineConfig, devices } from '@playwright/test';

export default defineConfig({
  testDir: './e2e',
  fullyParallel: true,
  forbidOnly: !!process.env.CI,
  retries: process.env.CI ? 2 : 0,
  workers: process.env.CI ? 1 : undefined,
  reporter: 'html',
  use: {
    baseURL: 'http://localhost:3000',
    trace: 'on-first-retry',
    screenshot: 'only-on-failure',
  },
  projects: [
    {
      name: 'chromium',
      use: { ...devices['Desktop Chrome'] },
    },
    {
      name: 'firefox',
      use: { ...devices['Desktop Firefox'] },
    },
    {
      name: 'webkit',
      use: { ...devices['Desktop Safari'] },
    },
    {
      name: 'Mobile Chrome',
      use: { ...devices['Pixel 5'] },
    },
    {
      name: 'Mobile Safari',
      use: { ...devices['iPhone 12'] },
    },
  ],
  webServer: {
    command: 'yarn dev',
    url: 'http://localhost:3000',
    reuseExistingServer: !process.env.CI,
  },
});
```

Impacto: +9.0 puntos en Testing (de 1/10 a 10/10)

🔒 5. MEJORAS DE SEGURIDAD (Prioridad: ALTA)

5.1 Rate Limiting Global

Problema: No hay protección contra abuso de APIs.

Solución:

```
yarn add @upstash/ratelimit @upstash/redis
```

```
// lib/rate-limit.ts
import { Ratelimit } from '@upstash/ratelimit';
import { Redis } from '@upstash/redis';

const redis = new Redis({
  url: process.env.UPSTASH_REDIS_REST_URL!,
  token: process.env.UPSTASH_REDIS_REST_TOKEN!,
});

// Different limits for different operations
export const rateLimiters = {
  // API general: 100 requests per minute
  api: new Ratelimit({
    redis,
    limiter: Ratelimit.slidingWindow(100, '1 m'),
    analytics: true,
    prefix: 'inmova:ratelimit:api',
  }),

  // Auth operations: 5 attempts per minute
  auth: new Ratelimit({
    redis,
    limiter: Ratelimit.slidingWindow(5, '1 m'),
    analytics: true,
    prefix: 'inmova:ratelimit:auth',
  }),

  // Exports: 3 per hour
  export: new Ratelimit({
    redis,
    limiter: Ratelimit.slidingWindow(3, '1 h'),
    analytics: true,
    prefix: 'inmova:ratelimit:export',
  }),
};

// Middleware
export async function rateLimit(
  identifier: string,
  limiter: Ratelimit = rateLimiters.api
) {
  const { success, limit, remaining, reset } = await limiter.limit(identifier);

  return {
    success,
    headers: {
      'X-RateLimit-Limit': limit.toString(),
      'X-RateLimit-Remaining': remaining.toString(),
      'X-RateLimit-Reset': new Date(reset).toISOString(),
    },
  };
}

// Uso en API routes:
// app/api/buildings/route.ts
export async function GET(request: NextRequest) {
  const session = await getServerSession(authOptions);
  if (!session) return NextResponse.json({ error: 'Unauthorized' }, { status: 401 });

  const identifier = session.user.id;
  const { success, headers } = await rateLimit(identifier);
  if (!success) {
    return NextResponse.json(
      { error: 'Too many requests' },
      { status: 429, headers }
    );
  }

  // ... rest of handler
}
```

Impacto: +1.0 puntos en Seguridad

5.2 Content Security Policy (CSP)

Solución:

```
// middleware.ts (agregar headers)
export function middleware(request: NextRequest) {
  const nonce = Buffer.from(crypto.randomUUID()).toString('base64');

  const cspHeader = `
    default-src 'self';
    script-src 'self' 'nonce-${nonce}' 'strict-dynamic' https://js.stripe.com;
    style-src 'self' 'unsafe-inline';
    img-src 'self' blob: data: https://inmova-assets.s3.amazonaws.com;
    font-src 'self';
    connect-src 'self' https://api.stripe.com https://inmova-api.abacusai.app;
    frame-src 'self' https://js.stripe.com;
    object-src 'none';
    base-uri 'self';
    form-action 'self';
    frame-ancestors 'none';
    upgrade-insecure-requests;
  `.replace(/\s{2,}/g, ' ').trim();

  const requestHeaders = new Headers(request.headers);
  requestHeaders.set('x-nonce', nonce);
  requestHeaders.set('Content-Security-Policy', cspHeader);
  requestHeaders.set('X-Frame-Options', 'DENY');
  requestHeaders.set('X-Content-Type-Options', 'nosniff');
  requestHeaders.set('Referrer-Policy', 'strict-origin-when-cross-origin');
  requestHeaders.set('Permissions-Policy', 'camera=(), microphone=(), geolocation=()')
;

  const response = NextResponse.next({
    request: {
      headers: requestHeaders,
    },
  });

  response.headers.set('Content-Security-Policy', cspHeader);

  return response;
}
```

Impacto: +0.8 puntos en Seguridad

5.3 Sanitización de Inputs

Problema: Inputs no sanitizados pueden causar XSS.

Solución:

```
yarn add dompurify isomorphic-dompurify
```

```
// lib/security/sanitize.ts
import DOMPurify from 'isomorphic-dompurify';
import { z } from 'zod';

export function sanitizeHtml(dirty: string): string {
  return DOMPurify.sanitize(dirty, {
    ALLOWED_TAGS: ['b', 'i', 'em', 'strong', 'a', 'p', 'br'],
    ALLOWED_ATTR: ['href', 'target', 'rel'],
  });
}

export function sanitizeInput(input: string): string {
  return input
    .trim()
    .replace(/[<>"']/g, '') // Remove potentially dangerous characters
    .slice(0, 1000); // Limit length
}

// Schemas de validación con Zod
export const buildingSchema = z.object({
  nombre: z.string()
    .min(1, 'El nombre es requerido')
    .max(100, 'El nombre es demasiado largo')
    .transform(sanitizeInput),
  direccion: z.string()
    .min(1, 'La dirección es requerida')
    .max(200)
    .transform(sanitizeInput),
  numeroUnidades: z.number()
    .int()
    .positive()
    .max(1000),
  // ...
});

// Uso en API:
export async function POST(request: NextRequest) {
  try {
    const body = await request.json();
    const validatedData = buildingSchema.parse(body);

    // validatedData ahora está sanitizado y validado
    const building = await prisma.building.create({
      data: validatedData,
    });

    return NextResponse.json(building);
  } catch (error) {
    if (error instanceof z.ZodError) {
      return NextResponse.json(
        { error: 'Validation failed', details: error.errors },
        { status: 400 }
      );
    }
    throw error;
  }
}
```

Impacto: +0.7 puntos en Seguridad

5.4 Auditoría de Dependencias

Solución:

```
# Añadir script al package.json
{
  "scripts": {
    "audit": "yarn audit --level moderate",
    "audit:fix": "yarn upgrade-interactive --latest"
  }
}

# GitHub Actions para auditoría automática
```

```
# .github/workflows/security-audit.yml
name: Security Audit

on:
  schedule:
    - cron: '0 0 * * 0' # Weekly on Sundays
  push:
    branches: [main]

jobs:
  audit:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '18'
      - run: yarn install --frozen-lockfile
      - run: yarn audit --level moderate
      - name: Dependabot alerts
        uses: github/dependabot-action@v1
```

Impacto: +0.5 puntos en Seguridad

5.5 Encrypt Sensitive Data at Rest

Solución:

```
// lib/security/encryption.ts
import crypto from 'crypto';

const ALGORITHM = 'aes-256-gcm';
const KEY = Buffer.from(process.env.ENCRYPTION_KEY!, 'hex'); // 32 bytes

export function encrypt(text: string): string {
  const iv = crypto.randomBytes(16);
  const cipher = crypto.createCipheriv(ALGORITHM, KEY, iv);

  let encrypted = cipher.update(text, 'utf8', 'hex');
  encrypted += cipher.final('hex');

  const authTag = cipher.getAuthTag();

  // Format: iv:authTag:encrypted
  return `${iv.toString('hex')}:${authTag.toString('hex')}:${encrypted}`;
}

export function decrypt(encryptedData: string): string {
  const [ivHex, authTagHex, encrypted] = encryptedData.split(':');

  const iv = Buffer.from(ivHex, 'hex');
  const authTag = Buffer.from(authTagHex, 'hex');
  const decipher = crypto.createDecipheriv(ALGORITHM, KEY, iv);

  decipher.setAuthTag(authTag);

  let decrypted = decipher.update(encrypted, 'hex', 'utf8');
  decrypted += decipher.final('utf8');

  return decrypted;
}

// Uso: Encrypt DNI, SSN, credit card numbers
export async function createTenant(data: TenantData) {
  const encryptedDNI = encrypt(data.dni);

  return prisma.tenant.create({
    data: {
      ...data,
      dni: encryptedDNI,
    },
  });
}
```

Impacto: +1.0 puntos en Seguridad

🚀 6. MEJORAS DE DEVOPS/CI-CD (Prioridad: MEDIA)

6.1 GitHub Actions CI/CD Pipeline

```yaml
# .github/workflows/ci.yml
name: CI/CD Pipeline

on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main, develop]

jobs:
  # Job 1: Lint
  lint:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '18'
          cache: 'yarn'
      - run: yarn install --frozen-lockfile
      - run: yarn lint
      - run: yarn tsc --noEmit

  # Job 2: Test
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '18'
          cache: 'yarn'
      - run: yarn install --frozen-lockfile
      - run: yarn test:ci
      - name: Upload coverage
        uses: codecov/codecov-action@v3
        with:
          files: ./coverage/lcov.info

  # Job 3: E2E Tests
  e2e:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '18'
          cache: 'yarn'
      - run: yarn install --frozen-lockfile
      - run: npx playwright install --with-deps
      - run: yarn build
      - run: yarn playwright test
      - uses: actions/upload-artifact@v3
        if: always()
        with:
          name: playwright-report
          path: playwright-report/

  # Job 4: Build
  build:
    runs-on: ubuntu-latest
    needs: [lint, test]
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '18'
          cache: 'yarn'
      - run: yarn install --frozen-lockfile
      - run: yarn prisma generate
      - run: yarn build
      - name: Analyze bundle size
        run: |
          ANALYZE=true yarn build
          ls -lh .next/static

  # Job 5: Deploy (solo en main)
  deploy:
    runs-on: ubuntu-latest
    needs: [build, e2e]
    if: github.ref == 'refs/heads/main'
    steps:
      - uses: actions/checkout@v3
      - name: Deploy to production
        run: |
          # Deployment logic here
          echo "Deploying to production..."
```

Impacto: +2.0 puntos en DevOps

6.2 Docker & Docker Compose

```dockerfile
# Dockerfile
FROM node:18-alpine AS base

# Dependencies
FROM base AS deps
RUN apk add --no-cache libc6-compat
WORKDIR /app

COPY package.json yarn.lock ./
RUN yarn install --frozen-lockfile

# Builder
FROM base AS builder
WORKDIR /app
COPY --from=deps /app/node_modules ./node_modules
COPY . .

ENV NEXT_TELEMETRY_DISABLED 1

RUN yarn prisma generate
RUN yarn build

# Runner
FROM base AS runner
WORKDIR /app

ENV NODE_ENV production
ENV NEXT_TELEMETRY_DISABLED 1

RUN addgroup --system --gid 1001 nodejs
RUN adduser --system --uid 1001 nextjs

COPY --from=builder /app/public ./public
COPY --from=builder /app/.next/standalone ./
COPY --from=builder /app/.next/static ./.next/static

USER nextjs

EXPOSE 3000

ENV PORT 3000
ENV HOSTNAME "0.0.0.0"

CMD ["node", "server.js"]
```

```yaml
# docker-compose.yml
version: '3.8'

services:
  app:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    environment:
      - DATABASE_URL=postgresql://inmova:inmova@postgres:5432/inmova
      - NEXTAUTH_URL=http://localhost:3000
      - NEXTAUTH_SECRET=${NEXTAUTH_SECRET}
    depends_on:
      - postgres
      - redis
    networks:
      - inmova-network

  postgres:
    image: postgres:15-alpine
    environment:
      POSTGRES_USER: inmova
      POSTGRES_PASSWORD: inmova
      POSTGRES_DB: inmova
    volumes:
      - postgres-data:/var/lib/postgresql/data
    ports:
      - "5432:5432"
    networks:
      - inmova-network

  redis:
    image: redis:7-alpine
    ports:
      - "6379:6379"
    networks:
      - inmova-network

volumes:
  postgres-data:

networks:
  inmova-network:
    driver: bridge
```

Impacto: +1.5 puntos en DevOps

6.3 Monitoreo y Logging con Sentry

```
yarn add @sentry/nextjs
npx @sentry/wizard -i nextjs
```

```
// sentry.client.config.ts
import * as Sentry from '@sentry/nextjs';

Sentry.init({
  dsn: process.env.NEXT_PUBLIC_SENTRY_DSN,
  tracesSampleRate: 0.1,
  environment: process.env.NODE_ENV,
  beforeSend(event, hint) {
    // Filter out sensitive data
    if (event.request) {
      delete event.request.cookies;
      delete event.request.headers;
    }
    return event;
  },
  integrations: [
    new Sentry.BrowserTracing({
      tracePropagationTargets: ['localhost', /^https:\/\/inmova\.abacusai\.app/],
    }),
    new Sentry.Replay({
      maskAllText: true,
      blockAllMedia: true,
    }),
  ],
  replaysSessionSampleRate: 0.1,
  replaysOnErrorSampleRate: 1.0,
});

// sentry.server.config.ts
import * as Sentry from '@sentry/nextjs';

Sentry.init({
  dsn: process.env.SENTRY_DSN,
  tracesSampleRate: 0.1,
  environment: process.env.NODE_ENV,
});
```

Impacto: +1.0 puntos en DevOps

6.4 Health Checks y Metrics

```
// app/api/health/route.ts
import { NextResponse } from 'next/server';
import { prisma } from '@/lib/db';
import { redis } from '@/lib/redis';

export async function GET() {
  const checks = {
    status: 'ok',
    timestamp: new Date().toISOString(),
    uptime: process.uptime(),
    checks: {} as Record<string, any>,
  };

  // Database check
  try {
    await prisma.$queryRaw`SELECT 1`;
    checks.checks.database = { status: 'healthy' };
  } catch (error) {
    checks.checks.databas
e = { status: 'unhealthy', error: error.message };
    checks.status = 'degraded';
  }

  // Redis check
  try {
    await redis.ping();
    checks.checks.redis = { status: 'healthy' };
  } catch (error) {
    checks.checks.redis = { status: 'unhealthy', error: error.message };
    checks.status = 'degraded';
  }

  // Memory check
  const memoryUsage = process.memoryUsage();
  checks.checks.memory = {
    status: memoryUsage.heapUsed < 500 * 1024 * 1024 ? 'healthy' : 'warning',
    heapUsed: `${Math.round(memoryUsage.heapUsed / 1024 / 1024)}MB`,
    heapTotal: `${Math.round(memoryUsage.heapTotal / 1024 / 1024)}MB`,
  };

  const statusCode = checks.status === 'ok' ? 200 : 503;

  return NextResponse.json(checks, { status: statusCode });
}

// app/api/metrics/route.ts (Prometheus-style)
export async function GET() {
  const metrics = [
    `# HELP nodejs_heap_size_total_bytes Total heap size`,
    `# TYPE nodejs_heap_size_total_bytes gauge`,
    `nodejs_heap_size_total_bytes ${process.memoryUsage().heapTotal}`,
    `# HELP nodejs_heap_size_used_bytes Used heap size`,
    `# TYPE nodejs_heap_size_used_bytes gauge`,
    `nodejs_heap_size_used_bytes ${process.memoryUsage().heapUsed}`,
    `# HELP process_uptime_seconds Process uptime`,
    `# TYPE process_uptime_seconds counter`,
    `process_uptime_seconds ${process.uptime()}`,
  ].join('\n');

  return new NextResponse(metrics, {
    headers: { 'Content-Type': 'text/plain' },
  });
}
```

Impacto: +0.5 puntos en DevOps

📚 7. MEJORAS DE DOCUMENTACIÓN (Prioridad: MEDIA)

7.1 Storybook para Componentes

```
npx storybook@latest init
```

```
// .storybook/main.ts
import type { StorybookConfig } from '@storybook/nextjs';

const config: StorybookConfig = {
  stories: ['../components/**/*.stories.@(js|jsx|ts|tsx)'],
  addons: [
    '@storybook/addon-links',
    '@storybook/addon-essentials',
    '@storybook/addon-interactions',
    '@storybook/addon-a11y', // Accessibility testing
  ],
  framework: {
    name: '@storybook/nextjs',
    options: {},
  },
  docs: {
    autodocs: 'tag',
  },
};

export default config;

// components/ui/button.stories.tsx
import type { Meta, StoryObj } from '@storybook/react';
import { Button } from './button';

const meta: Meta<typeof Button> = {
  title: 'UI/Button',
  component: Button,
  parameters: {
    layout: 'centered',
  },
  tags: ['autodocs'],
  argTypes: {
    variant: {
      control: 'select',
      options: ['default', 'destructive', 'outline', 'secondary', 'ghost', 'link'],
    },
    size: {
      control: 'select',
      options: ['default', 'sm', 'lg', 'icon'],
    },
  },
};

export default meta;
type Story = StoryObj<typeof Button>;

export const Primary: Story = {
  args: {
    children: 'Button',
    variant: 'default',
  },
};

export const Destructive: Story = {
  args: {
    children: 'Delete',
    variant: 'destructive',
  },
};
```

```
export const WithIcon: Story = {
  args: {
    children: (
      <>
        <Plus className="mr-2 h-4 w-4" />
        Crear Edificio
      </>
    ),
  },
};

export const Loading: Story = {
  args: {
    children: 'Guardando...',
    disabled: true,
  },
  render: (args) => (
    <Button {...args}>
      <Loader2 className="mr-2 h-4 w-4 animate-spin" />
      {args.children}
    </Button>
  ),
};
```

Impacto: +1.5 puntos en Documentación

7.2 API Documentation con Swagger

```
yarn add next-swagger-doc swagger-ui-react
yarn add -D @types/swagger-ui-react
```

```
// lib/swagger.ts
import { createSwaggerSpec } from 'next-swagger-doc';

export const getApiDocs = async () => {
  const spec = createSwaggerSpec({
    apiFolder: 'app/api',
    definition: {
      openapi: '3.0.0',
      info: {
        title: 'INMOVA API Documentation',
        version: '1.0.0',
        description: 'Complete API documentation for INMOVA platform',
      },
      servers: [
        {
          url: 'http://localhost:3000',
          description: 'Development server',
        },
        {
          url: 'https://homming-vidaro-6q1wdi.abacusai.app',
          description: 'Production server',
        },
      ],
      components: {
        securitySchemes: {
          BearerAuth: {
            type: 'http',
            scheme: 'bearer',
            bearerFormat: 'JWT',
          },
        },
        schemas: {
          Building: {
            type: 'object',
            properties: {
              id: { type: 'string' },
              nombre: { type: 'string' },
              direccion: { type: 'string' },
              numeroUnidades: { type: 'integer' },
              // ...
            },
          },
          // Define all schemas
        },
      },
      security: [{ BearerAuth: [] }],
    },
  });
  return spec;
};

// app/api-docs/page.tsx
'use client';

import SwaggerUI from 'swagger-ui-react';
import 'swagger-ui-react/swagger-ui.css';
import { useEffect, useState } from 'react';

export default function ApiDocsPage() {
  const [spec, setSpec] = useState(null);

  useEffect(() => {
    fetch('/api/docs')
      .then((res) => res.json())
      .then(setSpec);
  }, []);

  if (!spec) return <div>Loading...</div>;

  return (
    <div className="container mx-auto py-10">
      <h1 className="text-3xl font-bold mb-6">API Documentation</h1>
      <SwaggerUI spec={spec} />
    </div>
  );
}

// app/api/docs/route.ts
import { NextResponse } from 'next/server';
import { getApiDocs } from '@/lib/swagger';

export async function GET() {
  const spec = await getApiDocs();
  return NextResponse.json(spec);
}

// Documentar endpoints con JSDoc:
// app/api/buildings/route.ts
/**
 * @swagger
 * /api/buildings:
 *   get:
 *     summary: Get all buildings
 *     description: Returns a list of all buildings for the authenticated user's com-
pany
 *     tags: [Buildings]
 *     security:
 *       - BearerAuth: []
 *     responses:
 *       200:
 *         description: Successful response
 *         content:
 *           application/json:
 *             schema:
 *               type: array
 *               items:
 *                 $ref: '#/components/schemas/Building'
 *       401:
 *         description: Unauthorized
 */
export async function GET(request: NextRequest) {
  // Implementation
}
```

Impacto: +1.0 puntos en Documentación

7.3 README Completo y Guías de Contribución

~~~
# README.md

# 🏢 INMOVA - Plataforma Integral de Gestión Inmobiliaria

[![CI/CD](https://i.ytimg.com/vi/GlqQGLz6hfs/mqdefault.jpg)
[![Coverage](https://i.ytimg.com/vi/bNVRxb-MKGo/maxresdefault.jpg)
[![License](https://img.shields.io/badge/license-MIT-blue.svg)](/home/ubuntu/hom-
ming_vidaro/LICENSE)

> Plataforma SaaS profesional para gestión inmobiliaria con 88 módulos especializados

## 🚀 Características Principales

- 🏗️ **Multi-vertical**: Alquiler tradicional, STR, Flipping, Construction, Profes-
sional
- 📊 **Analytics avanzados**: Dashboard en tiempo real, BI, predicciones con IA
- 💳 **Pagos integrados**: Stripe, Open Banking, gestión de morosidad
- 🤖 **Asistente IA**: Chatbot inteligente, OCR, predicciones
- 📱 **PWA**: App offline-first con sincronización automática
- 🔐 **Enterprise-grade**: Multi-tenancy, RBAC, auditoría completa

## 📋 Requisitos

- Node.js 18+
- PostgreSQL 15+
- Redis 7+ (opcional, para rate limiting)
- AWS S3 (para storage)

## 🛠️ Instalación

```bash
# Clonar repositorio
git clone https://github.com/inmova/inmova.git
cd inmova/nextjs_space

# Instalar dependencias
yarn install

# Configurar variables de entorno
cp .env.example .env
# Editar .env con tus credenciales

# Generar Prisma Client
yarn prisma generate

# Ejecutar migraciones
yarn prisma migrate dev

# Seed data (opcional)
yarn prisma db seed

# Iniciar desarrollo
yarn dev
~~~

🧪 Testing

```
# Tests unitarios
yarn test

# Tests con coverage
yarn test:coverage

# Tests E2E
yarn playwright test

# Storybook (componentes)
yarn storybook
```

📦 Deployment

```
# Build producción
yarn build

# Docker
docker-compose up -d

# Deploy
yarn deploy
```

📚 Documentación

- 📖 Docs completa (https://docs.inmova.com)
- 🔌 API Reference (https://homming-vidaro-6q1wdi.abacusai.app/api-docs)
- 🎨 Storybook (https://storybook.inmova.com)
- 🏗️ Architecture (./docs/ARCHITECTURE.md)
- 👥 Contributing (./CONTRIBUTING.md)

🔒 Seguridad

- Security Policy (./SECURITY.md)
- Para reportar vulnerabilidades: security@inmova.com

📄 Licencia

MIT License - ver LICENSE (./LICENSE)

CONTRIBUTING.md

Guía de Contribución

¡Gracias por tu interés en contribuir a INMOVA!

🌟 Código de Conducta

Todos los contribuyentes deben seguir nuestro Code of Conduct (./CODE_OF_CONDUCT.md).

🔧 Setup de Desarrollo

1. Fork el repositorio
2. Clona tu fork: `git clone https://github.com/tu-usuario/inmova.git`
3. Crea una rama: `git checkout -b feature/mi-feature`
4. Instala dependencias: `yarn install`
5. Configura pre-commit hooks: `yarn husky install`

📝 Convenciones

Commits

Usamos Conventional Commits (https://www.conventionalcommits.org/):

```
feat: add support for room rental module
fix: resolve hydration error in dashboard
docs: update API documentation
style: format code with prettier
refactor: extract building service logic
test: add unit tests for coupon validation
chore: update dependencies
```

Code Style

```
# Lint
yarn lint

# Format
yarn format

# Type check
yarn tsc
```

Branches

- `main`: Producción estable
- `develop`: Desarrollo activo
- `feature/*`: Nuevas funcionalidades
- `fix/*`: Bug fixes

- `docs/*`: Documentación

✅ Pull Request Process

1. Actualiza tu rama con `main`:
   `bash`
   `git checkout main`
   `git pull upstream main`
   `git checkout tu-rama`
   `git rebase main`

2. Asegúrate de que todos los tests pasen:
   `bash`
   `yarn test`
   `yarn playwright test`

3. Actualiza documentación si es necesario

4. Crea el PR con template completo:
   - Descripción clara del cambio
   - Screenshots (si aplica)
   - Tests agregados
   - Checklist completado

5. Espera revisión de al menos 2 maintainers

🧪 Testing Guidelines

- **Cobertura mínima**: 80%
- Todos los componentes UI deben tener tests
- APIs críticas deben tener tests de integración
- Features nuevas deben incluir tests E2E

Ejemplo de test:

```
describe('BuildingCard', () => {
  it('should render building name', () => {
    render(<BuildingCard building={mockBuilding} />);
    expect(screen.getByText('Torre Vista')).toBeInTheDocument();
  });
});
```

📊 Performance Guidelines

- Components pesados deben ser lazy-loaded
- Images deben usar Next/Image
- APIs deben implementar pagination
- Queries DB deben tener índices apropiados

♿ Accessibility Guidelines

- Todos los componentes interactivos deben ser navegables por teclado
- Usar etiquetas ARIA apropiadas
- Contraste mínimo WCAG AA (4.5:1)
- Tests de accesibilidad obligatorios

🎨 UI/UX Guidelines

- Seguir Design System definido
- Usar componentes de `components/ui`
- Mantener consistencia visual
- Mobile-first approach

📞 Contacto

- Discord: https://discord.gg/inmova (https://discord.gg/inmova)
- Email: dev@inmova.com

~~~
**Impacto:** +1.5 puntos en Documentación

### 7.4 Arquitectura y Diagramas

```markdown
# docs/ARCHITECTURE.md

# Arquitectura INMOVA

## 🏗️ Visión General

INMOVA sigue una arquitectura de capas con separación clara de responsabilidades:
~~~

```
┌─────────────────────────────────────────────────┐
│ Presentation Layer │
│ (Next.js Pages, Components, UI, Client State) │
└─────────────────────────────────────────────────┘
↓
┌─────────────────────────────────────────────────┐
│ API Layer │
│ (API Routes, Middlewares, Validation, Auth) │
└─────────────────────────────────────────────────┘
↓
┌─────────────────────────────────────────────────┐
│ Service Layer │
│ (Business Logic, Calculations, Integrations) │
└─────────────────────────────────────────────────┘
↓
┌─────────────────────────────────────────────────┐
│ Data Layer │
│ (Prisma ORM, Database, Cache, Storage) │
└─────────────────────────────────────────────────┘
```

```
## 📂 Estructura de Carpetas
```

nextjs_space/
├── app/ # Next.js App Router
│ ├── (auth)/ # Auth layout group
│ ├── (dashboard)/ # Dashboard layout group
│ ├── api/ # API Routes
│ └── ... # Feature modules
├── components/ # React Components
│ ├── ui/ # Base UI components
│ └── layout/ # Layout components
├── lib/ # Core utilities
│ ├── services/ # Business logic
│ ├── hooks/ # Custom React hooks
│ └── utils/ # Helper functions
├── prisma/ # Database schema
├── public/ # Static assets
└── types/ # TypeScript types

```
## 🔄 Data Flow

### Request Flow
```

User Action
↓
Component Event Handler
↓
API Call (fetch/React Query)
↓
API Route Handler
↓
Middleware (Auth, Validation, Rate Limit)
↓
Service Layer (Business Logic)
↓
Database (Prisma)
↓
Response
↓
Component Update

~~~
### Authentication Flow

```mermaid
sequenceDiagram
    participant U as User
    participant C as Client
    participant A as API
    participant D as Database

    U->>C: Enter credentials
    C->>A: POST /api/auth/signin
    A->>D: Verify credentials
    D-->>A: User data
    A-->>C: JWT Token + Session
    C->>C: Store session
    C-->>U: Redirect to dashboard
~~~

🗄️ Database Schema

Core Entities

- **User**: Usuarios del sistema
- **Company**: Empresas (multi-tenancy)
- **Building**: Edificios
- **Unit**: Unidades/apartamentos
- **Tenant**: Inquilinos
- **Contract**: Contratos
- **Payment**: Pagos
- **Maintenance**: Mantenimiento

Relationships

```
Company 1──N User
Company 1──N Building
Building 1──N Unit
Unit 1──N Contract
Contract N──1 Tenant
Contract 1──N Payment
```

🔐 Security Architecture

Authentication

- NextAuth.js con JWT
- Session management
- Refresh tokens (7 días)

Authorization

- RBAC (Role-Based Access Control)
- Permissions: `super_admin`, `administrador`, `gestor`, `operador`
- Route protection via middleware
- API endpoint guards

Data Protection

- Encryption at rest (sensitive fields)
- TLS/SSL in transit
- CSP headers
- Rate limiting
- Input sanitization

🚀 Performance Optimizations

Frontend

- Code splitting por ruta
- Lazy loading de componentes pesados
- Image optimization (Next/Image)
- React Query cache
- Service Worker caching

Backend

- Database query optimization
- Indexes en campos frecuentes
- Connection pooling
- Redis caching
- CDN para assets

📊 Monitoring & Observability

- **Logging**: Sentry
- **Metrics**: Prometheus format
- **Health checks**: /api/health
- **Error tracking**: Sentry
- **Analytics**: Google Analytics

🔄 CI/CD Pipeline

```
Push to GitHub
    ↓
GitHub Actions
    ↓
├── Lint & Type Check
├── Unit Tests
├── E2E Tests
└── Build
    ↓
Deploy to Production
    ↓
Health Check
    ↓
Rollback if fails
```

🎯 Módulos Principales

Dashboard

Central hub con KPIs y analytics

Gestión de Edificios

CRUD completo, galería de fotos, documentos

Gestión de Inquilinos

Perfil, contratos, historial de pagos, comunicación

Mantenimiento

Órdenes de trabajo, tracking, proveedores

Contabilidad

Ingresos/gastos, reportes, conciliación

Multi-vertical Extensions

- **STR**: Sincronización Airbnb/Booking
- **Flipping**: Gestión de proyectos
- **Construction**: Control de obras
- **Professional**: Servicios profesionales

🔌 Integraciones Externas

- **Stripe**: Procesamiento de pagos
- **AWS S3**: Storage de archivos
- **SendGrid**: Emails transaccionales
- **Twilio**: SMS
- **Open Banking**: Conciliación bancaria
- **Zucchetti/ContaSimple**: Contabilidad

📱 PWA Features

- Service Worker con caching strategies
- Offline support
- Background sync
- Push notifications
- Install prompt

~~~
**Impacto:** +1.0 puntos en Documentación

---

## 🎯 8. MEJORAS ADICIONALES DE UX (Prioridad: MEDIA)

### 8.1 Onboarding Interactivo Mejorado

```typescript
// components/onboarding/InteractiveWalkthrough.tsx
import Joyride, { Step } from 'react-joyride';

const onboardingSteps: Step[] = [
  {
    target: '.dashboard-kpis',
    content: '¡Bienvenido a INMOVA! Aquí verás los KPIs principales de tu negocio.',
    disableBeacon: true,
  },
  {
    target: '.sidebar-buildings',
    content: 'Gestiona todos tus edificios desde aquí.',
  },
  {
    target: '.sidebar-tenants',
    content: 'Administra inquilinos, contratos y comunicaciones.',
  },
  {
    target: '.quick-actions',
    content: 'Accede rápidamente a las acciones más comunes.',
  },
  {
    target: '.search-global',
    content: 'Busca cualquier cosa con Cmd/Ctrl + K',
  },
];

export function InteractiveWalkthrough() {
  const [run, setRun] = useState(false);
  const [stepIndex, setStepIndex] = useState(0);

  useEffect(() => {
    const hasSeenOnboarding = localStorage.getItem('onboarding-completed');
    if (!hasSeenOnboarding) {
      setTimeout(() => setRun(true), 1000);
    }
  }, []);

  return (
    <Joyride
      steps={onboardingSteps}
      run={run}
      stepIndex={stepIndex}
      continuous
      showProgress
      showSkipButton
      styles={{
        options: {
          primaryColor: 'hsl(var(--primary))',
          zIndex: 10000,
        },
      }}
      callback={(data) => {
~~~

```
        const { status, action } = data;

        if (status === 'finished' || status === 'skipped') {
          localStorage.setItem('onboarding-completed', 'true');
          setRun(false);
        }

        if (action === 'next') {
          setStepIndex(stepIndex + 1);
        }
      }}
    />
  );
}
```

Instalar:

```
yarn add react-joyride
```

8.2 Keyboard Shortcuts Global

```
// hooks/use-keyboard-shortcuts.ts
import { useEffect } from 'react';
import { useRouter } from 'next/navigation';

export function useKeyboardShortcuts() {
  const router = useRouter();

  useEffect(() => {
    const handleKeyPress = (e: KeyboardEvent) => {
      const isMac = navigator.platform.toUpperCase().indexOf('MAC') >= 0;
      const modifier = isMac ? e.metaKey : e.ctrlKey;

      if (!modifier) return;

      switch (e.key.toLowerCase()) {
        case 'k':
          e.preventDefault();
          // Open search
          document.dispatchEvent(new CustomEvent('open-search'));
          break;
        case 'b':
          e.preventDefault();
          router.push('/edificios');
          break;
        case 't':
          e.preventDefault();
          router.push('/inquilinos');
          break;
        case 'd':
          e.preventDefault();
          router.push('/dashboard');
          break;
        case '/':
          e.preventDefault();
          // Show shortcuts modal
          document.dispatchEvent(new CustomEvent('show-shortcuts'));
          break;
      }
    };

    document.addEventListener('keydown', handleKeyPress);
    return () => document.removeEventListener('keydown', handleKeyPress);
  }, [router]);
}

// components/ShortcutsModal.tsx
export function ShortcutsModal() {
  const [open, setOpen] = useState(false);

  useEffect(() => {
    const handler = () => setOpen(true);
    document.addEventListener('show-shortcuts', handler);
    return () => document.removeEventListener('show-shortcuts', handler);
  }, []);

  const shortcuts = [
    { keys: ['Cmd', 'K'], description: 'Búsqueda global' },
    { keys: ['Cmd', 'B'], description: 'Ir a Edificios' },
    { keys: ['Cmd', 'T'], description: 'Ir a Inquilinos' },
    { keys: ['Cmd', 'D'], description: 'Ir a Dashboard' },
    { keys: ['Cmd', '/'], description: 'Ver atajos' },
    { keys: ['Esc'], description: 'Cerrar modal/diálogo' },
  ];

  return (
    <Dialog open={open} onOpenChange={setOpen}>
      <DialogContent>
        <DialogHeader>
          <DialogTitle>Atajos de Teclado</DialogTitle>
        </DialogHeader>
        <div className="space-y-4">
          {shortcuts.map((shortcut, i) => (
            <div key={i} className="flex items-center justify-between">
              <span>{shortcut.description}</span>
              <div className="flex gap-1">
                {shortcut.keys.map((key) => (
                  <kbd
                    key={key}
                    className="px-2 py-1 text-xs font-semibold bg-muted rounded"
                  >
                    {key}
                  </kbd>
                ))}
              </div>
            </div>
          ))}
        </div>
      </DialogContent>
    </Dialog>
  );
}
```

8.3 Drag & Drop para Múltiples Contextos

```
// lib/drag-and-drop/useDragAndDrop.ts
import { useCallback } from 'react';

export function useDragAndDrop<T>({
  onDrop,
  onDragOver,
}: {
  onDrop: (items: T[], targetId: string) => void;
  onDragOver?: (targetId: string) => void;
}) {
  const handleDragStart = useCallback((e: React.DragEvent, item: T) => {
    e.dataTransfer.effectAllowed = 'move';
    e.dataTransfer.setData('application/json', JSON.stringify(item));
  }, []);

  const handleDragOver = useCallback(
    (e: React.DragEvent, targetId: string) => {
      e.preventDefault();
      e.dataTransfer.dropEffect = 'move';
      onDragOver?.(targetId);
    },
    [onDragOver]
  );

  const handleDrop = useCallback(
    (e: React.DragEvent, targetId: string) => {
      e.preventDefault();
      const data = e.dataTransfer.getData('application/json');
      const items = JSON.parse(data);
      onDrop([items], targetId);
    },
    [onDrop]
  );

  return {
    handleDragStart,
    handleDragOver,
    handleDrop,
  };
}

// Uso: Arrastrar inquilinos a unidades
export function TenantAssignment() {
  const { handleDragStart, handleDragOver, handleDrop } = useDragAndDrop({
    onDrop: async (tenants, unitId) => {
      await assignTenantToUnit(tenants[0].id, unitId);
      toast.success('Inquilino asignado a la unidad');
    },
  });

  return (
    <div className="grid grid-cols-2 gap-6">
      <div>
        <h2>Inquilinos Sin Asignar</h2>
        {unassignedTenants.map((tenant) => (
          <Card
            key={tenant.id}
            draggable
            onDragStart={(e) => handleDragStart(e, tenant)}
            className="cursor-move"
          >
            {tenant.nombre}
          </Card>
        ))}
      </div>

      <div>
        <h2>Unidades Disponibles</h2>
        {units.map((unit) => (
          <Card
            key={unit.id}
            onDragOver={(e) => handleDragOver(e, unit.id)}
            onDrop={(e) => handleDrop(e, unit.id)}
            className="min-h-[100px] border-dashed"
          >
            {unit.numero}
          </Card>
        ))}
      </div>
    </div>
  );
}
```

8.4 Undo/Redo System

```
// lib/undo-redo/useUndoRedo.ts
import { useReducer, useCallback } from 'react';

interface State<T> {
  past: T[];
  present: T;
  future: T[];
}

type Action<T> =
  | { type: 'SET'; newPresent: T }
  | { type: 'UNDO' }
  | { type: 'REDO' }
  | { type: 'CLEAR' };

function undoRedoReducer<T>(state: State<T>, action: Action<T>): State<T> {
  switch (action.type) {
    case 'SET':
      return {
        past: [...state.past, state.present],
        present: action.newPresent,
        future: [],
      };
    case 'UNDO':
      if (state.past.length === 0) return state;
      const previous = state.past[state.past.length - 1];
      const newPast = state.past.slice(0, state.past.length - 1);
      return {
        past: newPast,
        present: previous,
        future: [state.present, ...state.future],
      };
    case 'REDO':
      if (state.future.length === 0) return state;
      const next = state.future[0];
      const newFuture = state.future.slice(1);
      return {
        past: [...state.past, state.present],
        present: next,
        future: newFuture,
      };
    case 'CLEAR':
      return {
        past: [],
        present: state.present,
        future: [],
      };
    default:
      return state;
  }
}

export function useUndoRedo<T>(initialPresent: T) {
  const [state, dispatch] = useReducer(undoRedoReducer, {
    past: [],
    present: initialPresent,
    future: [],
  } as State<T>);

  const set = useCallback((newPresent: T) => {
    dispatch({ type: 'SET', newPresent });
  }, []);
  const undo = useCallback(() => {
    dispatch({ type: 'UNDO' });
  }, []);

  const redo = useCallback(() => {
    dispatch({ type: 'REDO' });
  }, []);

  const clear = useCallback(() => {
    dispatch({ type: 'CLEAR' });
  }, []);

  const canUndo = state.past.length > 0;
  const canRedo = state.future.length > 0;

  return {
    state: state.present,
    setState: set,
    undo,
    redo,
    canUndo,
    canRedo,
    clear,
  };
}

// Uso en editor de edificio:
export function BuildingEditor({ initialBuilding }: { initialBuilding: Building }) {
  const { state, setState, undo, redo, canUndo, canRedo } = useUn-
doRedo(initialBuilding);

  return (
    <div>
      <div className="flex gap-2 mb-4">
        <Button
          onClick={undo}
          disabled={!canUndo}
          variant="outline"
          size="sm"
        >
          <Undo className="h-4 w-4" />
        </Button>
        <Button
          onClick={redo}
          disabled={!canRedo}
          variant="outline"
          size="sm"
        >
          <Redo className="h-4 w-4" />
        </Button>
      </div>

      <BuildingForm
        building={state}
        onChange={setState}
      />
    </div>
  );
}
```

Impacto conjunto sección 8: +1.0 puntos en UX/UI

📊 RESUMEN Y ROADMAP DE IMPLEMENTACIÓN

Impacto Total por Área

Área	Actual	Mejoras	Final	Ganancia
Funcionalidad Vertical	9/10	-	9/10	-
UX/UI	7/10	+4.0	10/10	+3.0
Rendimiento	6/10	+5.6	10/10	+4.0
Accesibilidad	4/10	+6.0	10/10	+6.0
Testing	1/10	+9.0	10/10	+9.0
Seguridad	7/10	+4.0	10/10	+3.0
DevOps	5/10	+5.0	10/10	+5.0
Documentación	6/10	+5.0	10/10	+4.0

🎯 Puntuación Final Proyectada: 9.75/10

🚀 ROADMAP DE IMPLEMENTACIÓN (4 Fases)

📅 Fase 1: Fundación (2-3 semanas)

Prioridad: CRÍTICA

Semana 1-2: Testing & CI/CD

- [ ] Setup Vitest + Testing Library
- [ ] Escribir tests para componentes UI críticos (20 tests)
- [ ] Tests de servicios core (coupon, payments, tenants)
- [ ] Setup GitHub Actions CI/CD
- [ ] Configurar Playwright E2E (5 flows críticos)

Semana 2-3: Seguridad

- [ ] Implementar Rate Limiting (Upstash Redis)
- [ ] Añadir CSP headers
- [ ] Input sanitization con Zod schemas
- [ ] Encryption para datos sensibles
- [ ] Security audit de dependencias

Entregable Fase 1:
- ✅ 80%+ test coverage en módulos críticos

- ✅ CI/CD pipeline funcional
- ✅ Security vulnerabilities resueltas

📅 Fase 2: Performance & Accesibilidad (2-3 semanas)

Prioridad: ALTA

Semana 3-4: Performance

- [ ] React Query para cache inteligente
- [ ] Lazy loading de componentes pesados
- [ ] Virtualización de listas (Tanstack Virtual)
- [ ] Activar optimización de imágenes
- [ ] Code splitting por rutas
- [ ] Bundle analyzer + optimizaciones

Semana 4-5: Accesibilidad

- [ ] Navegación por teclado completa
- [ ] Etiquetas ARIA en todos los componentes
- [ ] Focus management en modales
- [ ] Modo alto contraste
- [ ] Screen reader testing

Entregable Fase 2:
- ✅ Lighthouse Score > 90 (Performance, Accessibility)
- ✅ WCAG AA compliance
- ✅ Bundle inicial < 300KB

📅 Fase 3: UX Excellence (2-3 semanas)

Prioridad: MEDIA-ALTA

Semana 5-6: Design System & UI

- [ ] Design System completo (tokens.ts)
- [ ] Micro-interacciones con Framer Motion
- [ ] Sistema de notificaciones mejorado
- [ ] Skeleton screens específicos
- [ ] Service Worker avanzado

Semana 6-7: Búsqueda & Interacción

- [ ] Búsqueda global con Fuse.js
- [ ] Keyboard shortcuts (Cmd+K, etc.)
- [ ] Drag & Drop contexts
- [ ] Undo/Redo system
- [ ] Onboarding interactivo

Entregable Fase 3:
- ✅ UI pulida y consistente

- ✅ Feedback visual en todas las acciones
- ✅ Experiencia offline robusta

📅 Fase 4: DevOps & Documentación (2 semanas)

Prioridad: MEDIA

Semana 7-8: DevOps

- [ ] Docker + Docker Compose
- [ ] Sentry para monitoring
- [ ] Health checks y metrics
- [ ] Staging environment
- [ ] Automated deployments

Semana 8: Documentación

- [ ] Storybook para todos los componentes UI
- [ ] API docs con Swagger
- [ ] README completo + CONTRIBUTING.md
- [ ] Architecture docs con diagramas
- [ ] Video tutorials (opcional)

Entregable Fase 4:
- ✅ Infraestructura profesional
- ✅ Documentación completa y actualizada
- ✅ Onboarding fácil para nuevos devs

⚡ QUICK WINS (Implementar en 1 semana)

Si el tiempo es limitado, priorizar:

1. **Testing básico** (3 días)
   - 20 tests unitarios de componentes críticos
   - 5 tests E2E de flows principales
   - Setup CI/CD básico

2. **Security essentials** (2 días)
   - Rate limiting
   - Input validation con Zod
   - CSP headers

3. **Performance** (1 día)
   - React Query cache
   - Lazy loading de charts/calendars
   - Optimización de imágenes

4. **Accessibility** (1 día)
   - Navegación por teclado
   - Etiquetas ARIA básicas
   - Contraste de colores

Impacto Quick Wins: De 7.5/10 a 8.5/10 en 1 semana

💰 ESTIMACIÓN DE ESFUERZO

Categoría	Esfuerzo (horas)	Prioridad	Impacto
Testing	40-50h	CRÍTICA	⭐⭐⭐⭐⭐
Seguridad	20-25h	CRÍTICA	⭐⭐⭐⭐⭐
Performance	30-35h	ALTA	⭐⭐⭐⭐
Accesibilidad	25-30h	ALTA	⭐⭐⭐⭐
UX/UI	35-40h	MEDIA-ALTA	⭐⭐⭐⭐
DevOps	20-25h	MEDIA	⭐⭐⭐
Documentación	15-20h	MEDIA	⭐⭐⭐

Total: 185-225 horas (~5-6 semanas a tiempo completo)

🎓 RECURSOS DE APRENDIZAJE

Testing

- Testing Library Docs (https://testing-library.com/react)
- Vitest Guide (https://vitest.dev/guide/)
- Playwright Best Practices (https://playwright.dev/docs/best-practices)

Performance

- Next.js Performance (https://nextjs.org/docs/advanced-features/measuring-performance)
- React Query (https://tanstack.com/query/latest)
- Web Vitals (https://web.dev/vitals/)

Accessibility

- WCAG 2.1 Guidelines (https://www.w3.org/WAI/WCAG21/quickref/)
- A11y Project (https://www.a11yproject.com/)
- Radix UI A11y (https://www.radix-ui.com/docs/primitives/overview/accessibility)

Security

- OWASP Top 10 (https://owasp.org/www-project-top-ten/)
- Next.js Security (https://nextjs.org/docs/advanced-features/security-headers)

✅ CHECKLIST DE IMPLEMENTACIÓN

Usa este checklist para trackear progreso:

Testing

- [ ] Vitest configurado
- [ ] 20+ tests unitarios
- [ ] 10+ tests de integración
- [ ] 5+ tests E2E
- [ ] Coverage > 80%
- [ ] CI/CD ejecutando tests

Seguridad

- [ ] Rate limiting implementado
- [ ] CSP headers configurados
- [ ] Input sanitization (Zod)
- [ ] Encryption para datos sensibles
- [ ] Security audit passed

Performance

- [ ] React Query configurado
- [ ] Lazy loading de componentes
- [ ] Virtualización de listas
- [ ] Imágenes optimizadas
- [ ] Bundle < 300KB
- [ ] Lighthouse > 90

Accesibilidad

- [ ] Navegación por teclado
- [ ] Etiquetas ARIA
- [ ] Focus management
- [ ] Modo alto contraste
- [ ] WCAG AA compliance

UX/UI

- [ ] Design System definido
- [ ] Micro-interacciones
- [ ] Notificaciones mejoradas
- [ ] Skeletons específicos
- [ ] Búsqueda global
- [ ] Keyboard shortcuts

DevOps

- [ ] Docker setup
- [ ] Monitoring (Sentry)
- [ ] Health checks
- [ ] Automated deployment

Documentación

- [ ] Storybook
- [ ] API docs (Swagger)
- [ ] README completo
- [ ] CONTRIBUTING.md
- [ ] Architecture docs

🏁 CONCLUSIÓN

INMOVA tiene una **base sólida (7.5/10)** con funcionalidad vertical excelente. Las mejoras propuestas la elevarán a un **9.75/10** mediante:

1. **Testing robusto** (el gap más grande actual)
2. **Seguridad enterprise-grade**
3. **Performance optimizado** (Lighthouse 90+)
4. **Accesibilidad completa** (WCAG AA)
5. **UX pulida** con micro-interacciones y feedback
6. **DevOps profesional** con CI/CD
7. **Documentación completa**

Siguiente Paso Recomendado

Empezar con Fase 1 (Testing & Seguridad) - son las áreas más críticas y tienen el mayor impacto en la calidad percibida y real de la aplicación.

¿Quieres que implemente alguna de estas mejoras ahora mismo? 🚀