Documento de Mejoras UX - INMOVA

Resumen Ejecutivo

Este documento identifica las principales oportunidades de mejora en la experiencia de usuario (UX) de la plataforma INMOVA, basándose en un análisis exhaustivo del código actual, las mejores prácticas de diseño y los estándares de accesibilidad WCAG 2.1.

Fecha: Diciembre 2024
Versión: 1.0
Estado: Análisis Completado

Índice

---

1. Accesibilidad

1.1 Estado Actual

✅ **Implementado correctamente**:
- Focus visible mejorado con `ring-4` y `indigo-600`
- Skip links con `.skip-link` class
- Uso de componentes Shadcn UI con ARIA labels

⚠️ **Necesita Mejora**:

1.1.1 Alt Text en Imágenes

Problema: Algunas imágenes tienen alt text genérico o poco descriptivo.

Ejemplo encontrado:

```
// ❌ Antes
<img src={img} alt="Imagen de publicación" />

// ✅ Después
<img src={img} alt={`Imagen ${idx + 1} de $
{post.imagenes.length} en la publicación de ${post.nombreResidente}`} />
```

Recomendación:
- Implementar alt text descriptivo que incluya contexto
- Para imágenes decorativas, usar `alt=""`
- Para gráficos y charts, incluir descripción de los datos

Impacto: Alto - Accesibilidad para usuarios con lectores de pantalla
Esfuerzo: Bajo - 2-4 horas
Prioridad: Alta

1.1.2 Navegación por Teclado

Recomendaciones:

```
// Agregar navegación por teclado en listas interactivas
const handleKeyDown = (e: KeyboardEvent, action: () => void) => {
  if (e.key === 'Enter' || e.key === ' ') {
    e.preventDefault();
    action();
  }
};

<div
  role="button"
  tabIndex={0}
  onKeyDown={(e) => handleKeyDown(e, handleClick)}
  onClick={handleClick}
>
  {content}
</div>
```

Impacto: Alto
Esfuerzo: Medio - 8-12 horas
Prioridad: Alta

1.1.3 Contraste de Colores

Auditoría necesaria en:
- Badges con colores personalizados
- Textos sobre gradientes
- Estados de deshabilitado

Herramienta recomendada: Chrome DevTools Lighthouse

Impacto: Alto
Esfuerzo: Medio - 6-8 horas
Prioridad: Alta

---

2. Navegación y Arquitectura de Información

2.1 Breadcrumbs

✅ **Implementado**: Ya existe el componente `Breadcrumb`

⚠️ **Mejora**: Hacerlo consistente en todas las páginas de detalle

Páginas que necesitan breadcrumbs:
- `/edificios/[id]`
- `/unidades/[id]`
- `/inquilinos/[id]`
- `/contratos/[id]`
- `/room-rental/[unitId]/rooms/[roomId]`

Impacto: Medio
Esfuerzo: Bajo - 2-3 horas
Prioridad: Media

2.2 Menú de Navegación Contextual

Propuesta: Agregar menú secundario contextual según la sección

```
// Ejemplo para sección de edificios
const contextualMenu = [
  { label: 'Ver Edificio', icon: Eye, href: `/edificios/${id}` },
  { label: 'Unidades', icon: Home, href: `/edificios/${id}#unidades` },
  { label: 'Contratos', icon: FileText, href: `/edificios/${id}#contratos` },
  { label: 'Mantenimiento', icon: Wrench, href: `/edificios/${id}#mantenimiento` },
];
```

Beneficio: Reduce clics necesarios para tareas comunes

Impacto: Medio
Esfuerzo: Alto - 16-20 horas
Prioridad: Media

2.3 Atajos de Teclado

Propuesta: Implementar atajos globales

```
const shortcuts = {
  'Ctrl+K': 'Búsqueda global',
  'Ctrl+N': 'Nuevo (contexto actual)',
  'Ctrl+S': 'Guardar',
  'Esc': 'Cerrar modal/drawer',
  '?': 'Mostrar ayuda de atajos',
};
```

Componente recomendado: `cmdk` (ya instalado)

Impacto: Alto - Mejora significativa en productividad
Esfuerzo: Alto - 20-24 horas
Prioridad: Media-Alta

---

3. Formularios y Entrada de Datos

3.1 Validación en Tiempo Real

Estado actual: Validación mayormente en submit

Propuesta: Validación progresiva

```
import { useForm } from 'react-hook-form';
import { zodResolver } from '@hookform/resolvers/zod';
import * as z from 'zod';

const schema = z.object({
  email: z.string().email('Email inválido'),
  phone: z.string().regex(/^[0-9]{9}$/, 'Teléfono debe tener 9 dígitos'),
  rentaMensual: z.number().positive('Debe ser mayor a 0'),
});

// Validación on blur + submit
const { register, formState: { errors } } = useForm({
  resolver: zodResolver(schema),
  mode: 'onBlur', // Valida al salir del campo
});
```

Beneficios:
- Feedback inmediato
- Reduce errores en submit
- Mejora confianza del usuario

Impacto: Alto
Esfuerzo: Alto - 24-32 horas (aplicar a todos los formularios)
Prioridad: Alta

3.2 Autoguardado y Borradores

Problema: Si el usuario pierde conexión o cierra accidentalmente, pierde todo el trabajo

Propuesta: Implementar autoguardado en `localStorage`

```tsx
import { useEffect } from 'react';
import { useDebounce } from 'use-debounce';

function FormWithAutosave({ formId }: { formId: string }) {
  const [formData, setFormData] = useState({});
  const [debouncedData] = useDebounce(formData, 2000);

  useEffect(() => {
    // Recuperar borrador al montar
    const draft = localStorage.getItem(`draft-${formId}`);
    if (draft) {
      setFormData(JSON.parse(draft));
    }
  }, [formId]);

  useEffect(() => {
    // Guardar borrador automáticamente
    if (Object.keys(debouncedData).length > 0) {
      localStorage.setItem(`draft-${formId}`, JSON.stringify(debouncedData));
    }
  }, [debouncedData, formId]);

  const handleSubmit = () => {
    // Limpiar borrador al enviar
    localStorage.removeItem(`draft-${formId}`);
  };
}
```

Impacto: Medio-Alto
Esfuerzo: Medio - 12-16 horas
Prioridad: Media

3.3 Wizard Multi-paso Consistente

✅ **Parcialmente implementado**: `MobileFormWizard` en `/unidades/nuevo`

Propuesta: Extender a otros formularios largos:
- Creación de contratos
- Registro de inquilinos
- Configuración de edificios

Impacto: Alto
Esfuerzo: Medio - 8-12 horas por formulario
Prioridad: Alta

---

4. Feedback Visual y Estados

4.1 Estados de Carga

✅ **Implementado**: `LoadingState`, `SkeletonCard`, `LoadingSpinner`

⚠️ **Inconsistencia**: Algunos componentes aún usan spinners genéricos

Acción: Auditoría y reemplazo

```
# Encontrar usos de spinners genéricos
grep -r "Loader2" --include="*.tsx" | grep -v "LoadingState"
```

Impacto: Medio
Esfuerzo: Bajo - 4-6 horas
Prioridad: Baja

4.2 Optimistic Updates

Propuesta: Actualizar UI antes de confirmación del servidor

```
import { useMutation, useQueryClient } from '@tanstack/react-query';

function useToggleStatus(entityId: string) {
  const queryClient = useQueryClient();

  return useMutation({
    mutationFn: async (newStatus: string) => {
      return fetch(`/api/entities/${entityId}`, {
        method: 'PATCH',
        body: JSON.stringify({ status: newStatus }),
      });
    },
    onMutate: async (newStatus) => {
      // Cancelar queries en curso
      await queryClient.cancelQueries({ queryKey: ['entity', entityId] });

      // Snapshot del valor anterior
      const previousEntity = queryClient.getQueryData(['entity', entityId]);

      // Actualizar optimísticamente
      queryClient.setQueryData(['entity', entityId], (old: any) => ({
        ...old,
        status: newStatus,
      }));

      return { previousEntity };
    },
    onError: (err, newStatus, context) => {
      // Rollback en caso de error
      queryClient.setQueryData(['entity', entityId], context?.previousEntity);
    },
  });
}
```

Beneficio: La UI se siente instantánea

Impacto: Alto
Esfuerzo: Alto - 20-28 horas
Prioridad: Media

4.3 Progress Indicators

Casos de uso:
- Subida de archivos
- Generación de reportes
- Importación de datos
- Procesamiento OCR

Propuesta: `ProgressBar` component

```
interface ProgressBarProps {
  current: number;
  total: number;
  label?: string;
  showPercentage?: boolean;
}

export function ProgressBar({ current, total, label, showPercentage = true }: Pro-
gressBarProps) {
  const percentage = Math.round((current / total) * 100);

  return (
    <div className="w-full space-y-2">
      {label && (
        <div className="flex justify-between text-sm">
          <span className="text-gray-700">{label}</span>
          {showPercentage && (
            <span className="text-gray-500">{percentage}%</span>
          )}
        </div>
      )}
      <div className="w-full bg-gray-200 rounded-full h-2">
        <div
          className="bg-gradient-to-r from-indigo-500 to-violet-500 h-2 rounded-full
transition-all duration-300"
          style={{ width: `${percentage}%` }}
          role="progressbar"
          aria-valuenow={current}
          aria-valuemin={0}
          aria-valuemax={total}
        />
      </div>
    </div>
  );
}
```

Impacto: Medio
Esfuerzo: Bajo - 3-4 horas
Prioridad: Media

---

5. Mobile First y Responsive Design

5.1 Tablas en Mobile

Problema: Las tablas con muchas columnas son difíciles de usar en móvil

Estado actual: Algunas tablas usan scroll horizontal

Propuesta: Vista de tarjetas en mobile

```
function ResponsiveTable({ data, columns }: TableProps) {
  const isMobile = useMediaQuery('(max-width: 768px)');

  if (isMobile) {
    return (
      <div className="space-y-3">
        {data.map((item) => (
          <Card key={item.id} className="p-4">
            {columns.map((col) => (
              <div key={col.key} className="flex justify-between py-1">
                <span className="text-sm font-medium text-gray-500">
                  {col.label}
                </span>
                <span className="text-sm text-gray-900">
                  {col.render ? col.render(item) : item[col.key]}
                </span>
              </div>
            ))}
          </Card>
        ))}
      </div>
    );
  }

  return <DataTable data={data} columns={columns} />;
}
```

Impacto: Alto - Mejora dramática en mobile
Esfuerzo: Alto - 16-24 horas
Prioridad: Alta

5.2 Navigation Drawer en Mobile

✅ **Implementado**: El sidebar ya tiene comportamiento responsive

⚠️ **Mejora**: Agregar gesture de swipe para abrir/cerrar

Librería recomendada: `react-use-gesture`

Impacto: Medio
Esfuerzo: Medio - 6-8 horas
Prioridad: Baja

5.3 Touch Targets

Auditoría necesaria: Verificar que todos los botones tengan al menos 44x44px

```
/* Asegurar touch targets mínimos */
.touch-target {
  min-height: 44px;
  min-width: 44px;
  display: inline-flex;
  align-items: center;
  justify-content: center;
}
```

Impacto: Alto - Usabilidad mobile
Esfuerzo: Bajo - 2-4 horas
Prioridad: Alta

---

6. Performance y Optimización

6.1 Lazy Loading de Componentes Pesados

✅ **Implementado**: Charts ya usan lazy loading vía `lazy-charts-extended`

Propuesta: Extender a otros componentes:

```
import dynamic from 'next/dynamic';

// Lazy load de mapa
const MapView = dynamic(() => import('@/components/MapView'), {
  ssr: false,
  loading: () => <LoadingState message="Cargando mapa..." />,
});

// Lazy load de editor rich text
const RichTextEditor = dynamic(() => import('@/components/RichTextEditor'), {
  ssr: false,
});
```

Candidatos para lazy loading:
- Editor de contratos
- Visualizador de PDFs
- Componentes de chat
- Mapas de ubicación

Impacto: Alto - Mejora tiempo de carga inicial
Esfuerzo: Medio - 8-12 horas
Prioridad: Alta

6.2 Paginación y Virtualización

Problema: Listas muy largas pueden causar lag

Propuesta: Implementar virtualización con `react-window`

```tsx
import { FixedSizeList } from 'react-window';

function VirtualizedList({ items }: { items: any[] }) {
  const Row = ({ index, style }: { index: number; style: React.CSSProperties }) => (
    <div style={style}>
      <ItemCard item={items[index]} />
    </div>
  );

  return (
    <FixedSizeList
      height={600}
      itemCount={items.length}
      itemSize={120}
      width="100%"
    >
      {Row}
    </FixedSizeList>
  );
}
```

Casos de uso:
- Lista de edificios (>100 items)
- Lista de unidades
- Lista de contratos
- Lista de pagos

Impacto: Medio - Mejora en casos específicos
Esfuerzo: Medio - 8-12 horas
Prioridad: Baja

6.3 Image Optimization

Auditoría: Verificar uso del componente `Image` de Next.js

```tsx
// ✅ Correcto
import Image from 'next/image';
<Image src="/photo.jpg" alt="Descripción" width={500} height={300} />

// ❌ Evitar
<img src="/photo.jpg" alt="Descripción" />
```

Acción: Migrar todas las `<img>` a `<Image>` donde sea posible

Impacto: Medio - Mejora tiempo de carga
Esfuerzo: Bajo - 4-6 horas
Prioridad: Media

---

7. Consistencia y Sistema de Diseño

7.1 Tokens de Diseño

✅ **Implementado**: Variables CSS y Tailwind config

Propuesta: Documentar en Storybook

```tsx
// design-tokens.stories.tsx
export default {
  title: 'Design System/Tokens',
};

export const Colors = () => (
  <div className="grid grid-cols-4 gap-4">
    <ColorToken name="Primary" value="hsl(var(--primary))" />
    <ColorToken name="Secondary" value="hsl(var(--secondary))" />
    {/* ... */}
  </div>
);

export const Typography = () => (
  <div className="space-y-4">
    <h1 className="text-4xl font-bold">Heading 1</h1>
    <h2 className="text-3xl font-bold">Heading 2</h2>
    {/* ... */}
  </div>
);
```

Beneficio: Documentación viva del sistema de diseño

Impacto: Bajo - Documentación
Esfuerzo: Alto - 16-20 horas
Prioridad: Baja

7.2 Componentes Reutilizables

✅ **Bien implementado**: Muchos componentes UI reutilizables

Propuesta: Crear más composites

```tsx
// Ejemplo: FormField composite
interface FormFieldProps {
  label: string;
  name: string;
  type?: 'text' | 'email' | 'number';
  required?: boolean;
  helpText?: string;
  error?: string;
}

export function FormField({
  label,
  name,
  type = 'text',
  required,
  helpText,
  error,
}: FormFieldProps) {
  return (
    <div className="space-y-2">
      <Label htmlFor={name}>
        {label}
        {required && <span className="text-red-500 ml-1">*</span>}
      </Label>
      <Input
        id={name}
        name={name}
        type={type}
        required={required}
        aria-describedby={helpText ? `${name}-help` : undefined}
        aria-invalid={!!error}
      />
      {helpText && (
        <p id={`${name}-help`} className="text-sm text-gray-500">
          {helpText}
        </p>
      )}
      {error && (
        <p className="text-sm text-red-500" role="alert">
          {error}
        </p>
      )}
    </div>
  );
}
```

Impacto: Medio - Reduce código duplicado
Esfuerzo: Medio - 12-16 horas
Prioridad: Media

---

8. Onboarding y Primeros Pasos

8.1 Tour Interactivo

✅ **Implementado**: `OnboardingTourEnhanced` component

Propuesta: Extender con tours específicos por módulo

```
const tours = {
  dashboard: [
    { target: '#kpi-cards', content: 'Aquí ves tus KPIs principales' },
    { target: '#recent-activity', content: 'Actividad reciente de tu equipo' },
  ],
  edificios: [
    { target: '#new-building-btn', content: 'Crea tu primer edificio aquí' },
    { target: '#filters', content: 'Filtra por estado, tipo, etc.' },
  ],
  // ...
};
```

Impacto: Alto - Reduce curva de aprendizaje
Esfuerzo: Alto - 20-24 horas
Prioridad: Alta

8.2 Tooltips Contextuales

✅ **Implementado**: `InfoTooltip` component

Propuesta: Agregar en campos complejos

Campos que necesitan tooltips:
- `rendimientoEsperado` (fórmula de cálculo)
- `scoring` (criterios de puntuación)
- `rentaMensual` vs `deposito` (diferencias)
- `estadoConservacion` (criterios de cada nivel)

Impacto: Medio - Reduce confusión
Esfuerzo: Bajo - 4-6 horas
Prioridad: Media

8.3 Data Seeding para Demos

✅ **Implementado**: `DemoDataGenerator` component

Propuesta: Mejoras:
- Agregar opción de “Reset Demo Data”
- Templates predefinidos por vertical
- Datos más realistas (nombres, direcciones)

Impacto: Medio - Mejora experiencia de prueba
Esfuerzo: Medio - 8-12 horas
Prioridad: Baja

---

9. Gestión de Errores

9.1 Error Boundaries

✅ **Implementado**: `ErrorBoundary` component

⚠️ **Inconsistencia**: No todas las páginas lo usan

Acción: Envolver todas las páginas principales

```tsx
// app/layout.tsx
export default function RootLayout({ children }: { children: React.Node }) {
  return (
    <html>
      <body>
        <Providers>
          <ErrorBoundary>
            {children}
          </ErrorBoundary>
        </Providers>
      </body>
    </html>
  );
}
```

Impacto: Alto - Previene crashes completos
Esfuerzo: Bajo - 2-3 horas
Prioridad: Alta

9.2 Mensajes de Error Amigables

Problema: Algunos errores muestran mensajes técnicos

Propuesta: Error mapping

```tsx
const errorMessages: Record<string, string> = {
  NETWORK_ERROR: 'No se pudo conectar al servidor. Verifica tu conexión.',
  AUTH_EXPIRED: 'Tu sesión ha expirado. Por favor, inicia sesión nuevamente.',
  VALIDATION_ERROR: 'Algunos campos tienen errores. Por favor, revísalos.',
  NOT_FOUND: 'El recurso solicitado no existe.',
  PERMISSION_DENIED: 'No tienes permisos para realizar esta acción.',
  // ...
};

function getFriendlyError(error: Error): string {
  const errorCode = error.message.split(':')[0];
  return errorMessages[errorCode] || 'Ocurrió un error inesperado. Intenta nueva-
mente.';
}
```

Impacto: Alto - Mejora confianza del usuario
Esfuerzo: Medio - 8-12 horas
Prioridad: Alta

9.3 Retry Logic

Propuesta: Retry automático con backoff exponencial

```tsx
import { useQuery } from '@tanstack/react-query';

function useFetchWithRetry(endpoint: string) {
  return useQuery({
    queryKey: [endpoint],
    queryFn: () => fetch(endpoint).then(r => r.json()),
    retry: 3,
    retryDelay: (attemptIndex) => Math.min(1000 * 2 ** attemptIndex, 30000),
  });
}
```

Impacto: Medio - Mejora resiliencia
Esfuerzo: Bajo - 4-6 horas
Prioridad: Media

---

10. Internacionalización

10.1 Preparación para i18n

Estado actual: Textos hardcoded en español

Propuesta: Implementar `next-intl` o `react-i18next`

```tsx
// messages/es.json
{
  "common": {
    "save": "Guardar",
    "cancel": "Cancelar",
    "delete": "Eliminar"
  },
  "buildings": {
    "title": "Edificios",
    "create": "Crear Edificio",
    "totalProperties": "Total de propiedades"
  }
}

// Uso
import { useTranslations } from 'next-intl';

function BuildingsPage() {
  const t = useTranslations('buildings');

  return <h1>{t('title')}</h1>;
}
```

Impacto: Alto - Habilita expansión internacional
Esfuerzo: Muy Alto - 40-60 horas
Prioridad: Baja (a menos que sea requisito inmediato)

10.2 Formateo de Fechas y Números

✅ **Implementado**: Uso de `date-fns` con locale `es`

Propuesta: Centralizar en un utility

```tsx
// lib/i18n-utils.ts
import { format as dateFnsFormat } from 'date-fns';
import { es, en, fr } from 'date-fns/locale';

const locales = { es, en, fr };

export function formatDate(date: Date, formatStr: string, locale: string = 'es') {
  return dateFnsFormat(date, formatStr, { locale: locales[locale] });
}

export function formatCurrency(amount: number, currency: string = 'EUR', locale: strin
g = 'es-ES') {
  return new Intl.NumberFormat(locale, {
    style: 'currency',
    currency,
  }).format(amount);
}
```

Impacto: Medio - Preparación para i18n
Esfuerzo: Bajo - 4-6 horas
Prioridad: Baja

Resumen de Prioridades

🔴 Prioridad Alta (Implementar en Sprint 1-2)

1. **Validación en tiempo real de formularios** (24-32h)
2. **Accesibilidad: Alt text en imágenes** (2-4h) ✅ Iniciado
3. **Accesibilidad: Navegación por teclado** (8-12h)
4. **Wizard multi-paso en formularios largos** (8-12h/formulario)
5. **Tablas responsive en mobile** (16-24h)
6. **Lazy loading de componentes pesados** (8-12h)
7. **Error boundaries globales** (2-3h)
8. **Mensajes de error amigables** (8-12h)
9. **Touch targets mínimos (44x44px)** (2-4h)
10. **Onboarding: Tours por módulo** (20-24h)

Total estimado: 98-143 horas (~2.5-3.5 sprints de 40h)

🟡 Prioridad Media (Sprint 3-4)

1. **Autoguardado y borradores** (12-16h)
2. **Optimistic updates** (20-28h)
3. **Progress indicators** (3-4h)
4. **Breadcrumbs consistentes** (2-3h)
5. **Menú navegación contextual** (16-20h)
6. **Retry logic automático** (4-6h)
7. **Image optimization** (4-6h)
8. **Tooltips contextuales** (4-6h)
9. **Componentes composite** (12-16h)

Total estimado: 77-109 horas (~2-2.5 sprints)

🟢 Prioridad Baja (Backlog)

1. **Atajos de teclado globales** (20-24h)
2. **Navigation drawer con gestures** (6-8h)
3. **Virtualización de listas** (8-12h)
4. **Storybook para design system** (16-20h)
5. **Demo data improvements** (8-12h)
6. **Internacionalización completa** (40-60h)
7. **Contraste de colores audit** (6-8h)
8. **Estados de carga consistentes** (4-6h)

Total estimado: 108-150 horas (~2.5-4 sprints)

Métricas de Éxito

KPIs para Medir Mejoras UX

Cuantitativos

1. **Time to First Interaction (TTFI)**
   - Objetivo: < 2 segundos
   - Medición: Lighthouse / Web Vitals

2. **Error Rate**
   - Objetivo: < 1% de transacciones
   - Medición: Sentry / Error tracking

3. **Task Completion Rate**
   - Objetivo: > 95% para tareas principales
   - Medición: Google Analytics / Hotjar

4. **Accessibility Score**
   - Objetivo: 95+ en Lighthouse
   - Medición: Lighthouse CI

5. **Mobile Usability**
   - Objetivo: < 3% abandono en mobile
   - Medición: Google Analytics

Cualitativos

1. **System Usability Scale (SUS)**
   - Objetivo: Score > 75
   - Medición: Encuesta post-onboarding

2. **Net Promoter Score (NPS)**
   - Objetivo: > 50
   - Medición: Encuesta trimestral

3. **User Interviews**
   - Frecuencia: Mensual
   - Objetivo: Identificar pain points

Herramientas Recomendadas

Análisis y Testing

- **Lighthouse**: Auditoría de performance y accesibilidad
- **axe DevTools**: Testing de accesibilidad
- **React DevTools Profiler**: Optimización de renders
- **Chrome DevTools**: Network, Performance

User Research

- **Hotjar**: Heatmaps y session recordings
- **Maze**: User testing remoto
- **UserTesting.com**: Feedback cualitativo

Monitoring

- **Sentry**: Error tracking
- **LogRocket**: Session replay con errores
- **Google Analytics 4**: Comportamiento de usuarios

Siguientes Pasos

Fase 1: Auditoría Completa (1 semana)

- [ ] Lighthouse audit en todas las páginas principales
- [ ] axe accessibility scan
- [ ] Inventario de todos los formularios
- [ ] Identificar puntos de dolor comunes (user interviews)

Fase 2: Quick Wins (2 semanas)

- [ ] Alt text en imágenes ✅
- [ ] Error boundaries globales
- [ ] Touch targets mínimos
- [ ] Breadcrumbs consistentes

Fase 3: Implementación Core (6-8 semanas)

- [ ] Validación en tiempo real
- [ ] Tablas responsive
- [ ] Wizards multi-paso
- [ ] Lazy loading
- [ ] Tours interactivos

Fase 4: Optimización (4 semanas)

- [ ] Optimistic updates
- [ ] Autoguardado
- [ ] Performance tuning

Fase 5: Polish (2-3 semanas)

- [ ] Atajos de teclado
- [ ] Componentes composite
- [ ] Storybook

Conclusiones

La plataforma INMOVA ya tiene una base sólida de UX con componentes bien estructurados y patrones consistentes. Las mejoras propuestas se enfocan en:

1. **Accesibilidad**: Garantizar que todos los usuarios puedan usar la plataforma
2. **Performance**: Tiempos de carga más rápidos y sensación de fluidez
3. **Mobile**: Experiencia óptima en dispositivos móviles
4. **Productividad**: Reducir clics y tiempo para completar tareas
5. **Confianza**: Feedback claro y manejo robusto de errores

La implementación gradual por prioridades permitirá entregar valor de forma incremental mientras se mantiene la estabilidad del sistema.

Documento preparado por: DeepAgent
Fecha: 6 de Diciembre, 2024
Versión: 1.0