🚀 Plan de Mejoras - Dónde Queda?

📋 URGENTES (Críticas - Hacer YA)

🔴 Seguridad

- [ ] **Nunca guardar passwords en localStorage** - Solo tokens JWT
- [ ] Implementar refresh tokens para sesiones largas
- [ ] Agregar HTTPS obligatorio (no usar HTTP en producción)
- [ ] Sanitizar inputs del usuario (prevenir XSS)
- [ ] Agregar rate limiting en el backend
- [ ] Validar tokens en cada petición

🔴 Bugs Críticos

- [ ] Arreglar `response.date` → `response.data` en Api.jsx (línea 20)
- [ ] Eliminar código duplicado en loginUser (dos `return response.data`)
- [ ] Validar que el backend retorne estructura consistente `{ user, token }`
- [ ] Manejar errores 401 (sesión expirada) globalmente
- [ ] Agregar loading states en todas las peticiones async

🟡 CORTO PLAZO (1-2 semanas)

Backend Integration

- [ ] Crear endpoint `/usuario/verificar-email` para checkEmail
- [ ] Implementar recuperación de contraseña real (no alert)
- [ ] Conectar categorías reales desde backend
- [ ] Cargar slides del carousel desde API
- [ ] Implementar búsqueda real de negocios

UX/UI Mejoras

- [ ] Agregar skeleton loaders mientras carga contenido
- [ ] Mejorar feedback visual en errores (toast notifications)
- [ ] Agregar animaciones de carga en botones
- [ ] Implementar modo oscuro
- [ ] Hacer navbar responsive en móvil (hamburger menu)

Code Quality

- [ ] Crear archivo `.env` para API_URL (no hardcodear IPs)
- [ ] Separar constantes en archivo `constants.js`

- [ ] Agregar PropTypes en TODOS los componentes
- [ ] Implementar ESLint + Prettier
- [ ] Documentar funciones complejas con JSDoc

🟢 MEDIANO PLAZO (3-4 semanas)

Funcionalidades Nuevas

- [ ] Sistema de favoritos real (guardar en backend)
- [ ] Notificaciones push
- [ ] Filtros avanzados de búsqueda
- [ ] Mapa interactivo con ubicación de negocios
- [ ] Sistema de reviews y calificaciones
- [ ] Chat en tiempo real (WebSockets)

Performance

- [ ] Implementar lazy loading de imágenes
- [ ] Code splitting por rutas
- [ ] Cachear respuestas del backend (React Query o SWR)
- [ ] Optimizar bundle size (análisis con webpack-bundle-analyzer)
- [ ] Implementar service workers (PWA)

Testing

- [ ] Tests unitarios para componentes (Jest + React Testing Library)
- [ ] Tests de integración para flujos críticos (login/register)
- [ ] Tests E2E con Cypress
- [ ] Cobertura mínima 70%

🔵 LARGO PLAZO (1-3 meses)

Arquitectura

- [ ] Migrar a TypeScript
- [ ] Implementar Redux/Zustand para estado global complejo
- [ ] Crear design system con Storybook
- [ ] Migrar CSS Modules a styled-components o Tailwind
- [ ] Implementar micro-frontends si escala

Features Avanzadas

- [ ] Login social real (Google, Facebook, Apple OAuth)

- [ ] Sistema de subscripciones/membresías
- [ ] Panel de administración para negocios
- [ ] Analytics y métricas (Google Analytics/Mixpanel)
- [ ] Geolocalización y recomendaciones por proximidad
- [ ] Sistema de reservas/citas

DevOps

- [ ] CI/CD con GitHub Actions
- [ ] Deploy automático en Vercel/Netlify
- [ ] Monitoreo de errores (Sentry)
- [ ] Logs estructurados
- [ ] Backup automático de BD

🛠️ REFACTORIZACIONES IMPORTANTES

Estructura de Archivos

```
src/
├── api/
│   ├── config.js          (axios config)
│   ├── auth.api.js        (login, register, logout)
│   ├── business.api.js    (negocios)
│   └── categories.api.js  (categorías)
├── components/
│   ├── common/            (Badge, Button, Input)
│   ├── layout/            (Navbar, Footer, Sidebar)
│   └── features/          (Carousel, PromoCard, etc)
├── hooks/
│   ├── useAuth.js
│   ├── useDebounce.js
│   └── useLocalStorage.js
├── contexts/
│   └── UserContext.jsx
├── utils/
│   ├── validators.js      (validación de email, passwords)
│   ├── formatters.js      (fechas, monedas)
│   └── constants.js       (URLs, colores, etc)
└── styles/
    ├── theme.js
    └── globals.css
```

Componentes a Dividir

- [ ] **Navbar**: Separar en SearchBar, UserMenu, NavIcons
- [ ] **LoginModal**: Extraer FormFields, PasswordInput, SocialButtons
- [ ] **Carousel**: Simplificar lógica, usar librería (Swiper.js)
- [ ] **Home**: Dividir en secciones independientes

📊 MÉTRICAS DE ÉXITO

Performance Goals

- [ ] Lighthouse Score > 90
- [ ] First Contentful Paint < 1.5s
- [ ] Time to Interactive < 3s
- [ ] Bundle size < 200KB (gzipped)

Code Quality Goals

- [ ] Test coverage > 70%
- [ ] 0 errores de ESLint
- [ ] Accesibilidad WCAG 2.1 AA

User Experience Goals

- [ ] Tiempo de login < 2s
- [ ] Tasa de error en forms < 5%
- [ ] Bounce rate < 40%

🎯 PRIORIZACIÓN (Matriz de Impacto)

Alto Impacto + Fácil Implementación ⭐ ⭐ ⭐

1. Arreglar bugs de Api.jsx
2. Agregar .env para configuración
3. Mejorar mensajes de error
4. Skeleton loaders
5. Modo oscuro

Alto Impacto + Difícil Implementación ⭐ ⭐

1. TypeScript migration
2. Testing completo

3. Performance optimization
4. Sistema de notificaciones
5. Chat en tiempo real

Bajo Impacto + Fácil ⭐

1. Prettier/ESLint setup
2. JSDoc documentation
3. PropTypes en todos los componentes

🗓️ CRONOGRAMA SUGERIDO

Semana 1-2: Fundaciones

- Arreglar bugs críticos
- Setup .env y constantes
- Implementar validaciones robustas
- Mejorar manejo de errores

Semana 3-4: UX

- Skeleton loaders
- Toast notifications
- Responsive navbar
- Modo oscuro

Semana 5-8: Features

- Favoritos reales
- Búsqueda avanzada
- Sistema de reviews
- Mapa interactivo

Semana 9-12: Quality

- Testing suite completo
- Performance optimization
- Refactoring arquitectura
- Documentation

🔥 QUICK WINS (Mejoras en < 1 hora)

1. ✅ Arreglar typo `response.date` → `response.data`
2. ✅ Agregar `.env.example` con variables necesarias
3. ✅ Implementar debounce en búsqueda (useDebounce hook)
4. ✅ Agregar favicon y meta tags
5. ✅ Implementar toast notifications (react-hot-toast)
6. ✅ Agregar loading spinners en botones
7. ✅ Validar email con regex mejorado
8. ✅ Agregar auto-focus en inputs del modal

💡 MEJORES PRÁCTICAS A IMPLEMENTAR

React

- [ ] Usar React.memo para componentes pesados
- [ ] Implementar error boundaries
- [ ] Usar Suspense + lazy para code splitting
- [ ] Custom hooks para lógica reutilizable

Estado

- [ ] Normalizar estructura de datos
- [ ] Evitar prop drilling (context o estado global)
- [ ] Inmutabilidad estricta

API

- [ ] Interceptors de Axios para tokens
- [ ] Retry logic para peticiones fallidas
- [ ] Request cancellation (AbortController)
- [ ] Optimistic updates

Seguridad

- [ ] Content Security Policy headers
- [ ] CORS configurado correctamente
- [ ] Input sanitization
- [ ] SQL injection prevention (backend)

📚 RECURSOS RECOMENDADOS

- **Testing**: React Testing Library
- **State Management**: Zustand
- **Forms**: React Hook Form
- **Styling**: Tailwind CSS
- **API**: React Query
- **Auth**: Auth0 o Firebase Auth

✅ CHECKLIST ANTES DE PRODUCCIÓN

- [ ] Todas las variables en .env
- [ ] HTTPS habilitado
- [ ] Rate limiting implementado
- [ ] Logs de errores configurados (Sentry)
- [ ] Tests críticos pasando
- [ ] Performance audit aprobado
- [ ] Seguridad auditada
- [ ] Backup strategy definida
- [ ] Monitoreo configurado
- [ ] Documentación actualizada