🚀 RapidEats Premium - Prompt de Migración Completa

Proyecto: Transformar RapidEats en una plataforma premium valorada en $100,000 USD

📋 CONTEXTO DEL PROYECTO

Proyecto actual: RapidEats - Plataforma de delivery con MERN stack **Objetivo:** Alcanzar nivel premium tipo Uber Eats con diseño UI/UX minimalista excepcional **Stack tecnológico:** MongoDB, Express, React 18 + TypeScript, Node.js, Socket.io **Mercado objetivo:** Colombia-Venezuela (Cúcuta, San Antonio del Táchira)

Características actuales existentes:

- ✅ Google OAuth authentication
- ✅ Sistema de pedidos con Socket.io (real-time)
- ✅ Dashboard de restaurantes
- ✅ Panel administrativo
- ✅ Stripe payments
- ✅ Sistema de reseñas y favoritos
- ✅ Telegram bot para repartidores
- ✅ Push notifications (Firebase)
- ✅ Sistema de cupones y zonas

🎨 FILOSOFÍA DE DISEÑO

Estilo objetivo: Minimalismo Premium - Inspirado en Uber Eats/DoorDash

- Espacios en blanco generosos
- Tipografía limpia y legible (Inter, SF Pro)
- Colores neutros con acentos vibrantes
- Micro-interacciones fluidas
- Glassmorphism sutil
- Animaciones suaves (Framer Motion)

- Mobile-first responsive design

Paleta de colores premium:

```css
/* Primarios */
--rapid-black: #000000
--rapid-white: #FFFFFF
--rapid-green: #06C167 /* Verde delivery premium */
--rapid-red: #FF3B30 /* Rojo para alertas */

/* Neutros */
--gray-50: #F9FAFB
--gray-100: #F3F4F6
--gray-200: #E5E7EB
--gray-300: #D1D5DB
--gray-400: #9CA3AF
--gray-500: #6B7280
--gray-600: #4B5563
--gray-700: #374151
--gray-800: #1F2937
--gray-900: #111827

/* Semánticos */
--success: #10B981
--warning: #F59E0B
--error: #EF4444
--info: #3B82F6
```

🔥 FASE 1: REDISEÑO COMPLETO DE UI/UX (FRONTEND)

Objetivo: Transformar el frontend en una experiencia premium minimalista

1.1 Sistema de Diseño Base

Instrucciones para Copilot:

Crea un sistema de diseño completo en /frontend/src/styles/ con los siguientes archivos:

1. **design-tokens.css** - Variables CSS premium:
   - Colores (paleta completa con variantes hover/active/disabled)
   - Tipografía (escala modular: 12, 14, 16, 18, 20, 24, 32, 48, 64px)
   - Espaciado (escala 4pt: 4, 8, 12, 16, 24, 32, 48, 64, 96, 128px)
   - Sombras (4 niveles: sm, md, lg, xl con blur y spread precisos)
   - Bordes (radios: 4, 8, 12, 16, 24, 999px)
   - Z-index (sistema consistente: 0, 10, 20, 30, 40, 50)
   - Transiciones (cubic-bezier personalizados para cada tipo)
   - Breakpoints responsive (mobile: 0, tablet: 768, desktop: 1024, wide: 1440px)

2. **animations.css** - Animaciones premium:
   - Fade in/out con scale sutil
   - Slide from directions (top, right, bottom, left)
   - Bounce suave para botones
   - Skeleton loading elegante
   - Shimmer effect para cargas
   - Ripple effect para clicks
   - Spring animations para modales
   - Page transitions fluidas
   - Scroll-triggered animations

3. **utilities.css** - Utilidades consistentes:
   - Grid systems (12 columnas, gaps consistentes)
   - Flexbox helpers (justify, align, direction, wrap)
   - Spacing helpers (margin/padding con escala)
   - Typography helpers (weights, sizes, line-heights)
   - Display utilities (show/hide, responsive visibility)
   - Truncate text utilities (1, 2, 3 líneas)
   - Aspect ratio utilities (1:1, 16:9, 4:3)
   - Elevation system (sombras consistentes)

Asegúrate de usar CSS moderno: CSS Grid, Flexbox, CSS Variables, clamp(), min(), max()

1.2 Componentes Base Premium

Instrucciones para Copilot:

Rediseña todos los componentes en /frontend/src/components/common/ con diseño premium:

1. **Button.tsx** - Sistema de botones completo:
   Props: variant (primary, secondary, outline, ghost, danger), size (sm, md, lg, xl)
   Estados: default, hover, active, disabled, loading
   Incluir: ripple effect, icon support, loading spinner integrado
   Estilos: bordes redondeados, sombras sutiles, transiciones suaves
   Accesibilidad: ARIA labels, keyboard navigation, focus visible

2. **Input.tsx** - Campos de entrada elegantes:
   Tipos: text, email, password, number, tel, search
   Props: label, placeholder, error, helper text, icon (left/right)
   Estados: focus (border color change), error (red border + shake), success, disabled
   Características: auto-validate, show/hide password, clear button
   Animaciones: label float cuando hay contenido, smooth focus transition

3. **Card.tsx** - Tarjetas versátiles:
   Variantes: elevated (sombra), outlined (borde), filled (background)
   Props: padding, radius, hoverable (efecto hover), clickable
   Secciones: header, body, footer con separadores opcionales
   Hover effect: elevación suave, scale 1.02
   Loading state: skeleton placeholder

4. **Modal.tsx** - Modales premium:
   Animación: fade + slide from bottom (mobile) o scale (desktop)
   Backdrop: blur + darkening, dismissible
   Tamaños: sm (400px), md (600px), lg (800px), xl (1200px), full
   Props: onClose, closeButton, dismissible, centered
   Accesibilidad: trap focus, ESC para cerrar, click fuera para cerrar
   Variantes: center, bottom sheet (mobile), side panel

5. **Toast.tsx** - Notificaciones elegantes:
   Posiciones: top-left, top-center, top-right, bottom-left, bottom-center, bottom-right
   Tipos: success, error, warning, info, loading
   Animación: slide + fade from position
   Auto-dismiss con progress bar
   Acciones: dismiss button, action button opcional
   Stack: múltiples toasts con spacing

6. **Skeleton.tsx** - Loading states elegantes:
   Variantes: text, circle, rectangle, custom
   Animación: shimmer effect suave
   Props: width, height, count (múltiples líneas)

Usarlo en: cards, lists, forms mientras carga

7. **Badge.tsx** - Insignias informativas:
   Variantes: solid, outline, soft (background suave)
   Colores: default, success, warning, error, info
   Tamaños: sm, md, lg
   Props: dot (solo punto), icon, removable

8. **Avatar.tsx** - Avatares de usuario:
   Props: src, alt, size (xs, sm, md, lg, xl), shape (circle, rounded, square)
   Fallback: iniciales con background color basado en nombre
   Badge: indicator dot (online, offline, away, busy)
   Group: AvatarGroup para múltiples avatares apilados

9. **Tabs.tsx** - Navegación por pestañas:
   Variantes: line (underline), pill, enclosed
   Animación: sliding indicator smooth
   Props: defaultIndex, onChange, fullWidth
   Lazy loading de contenido
   Responsive: scroll horizontal en mobile

10. **Dropdown.tsx** - Menús desplegables:
    Trigger: click, hover, manual
    Posición: auto-adjust con Popper.js
    Animación: fade + slide from trigger
    Props: items array, onSelect, placement
    Características: keyboard navigation, search filtering

1.3 Rediseño de Páginas Principales

Instrucciones para Copilot:

Rediseña las siguientes páginas en /frontend/src/pages/ con diseño premium minimalista:

=== HOME PAGE (/) ===
Estructura nueva:

1. **Hero Section** - Impacto visual:
   - Headline grande: "Comida deliciosa, entregada rápido" (64px, bold)
   - Subheadline: "Miles de restaurantes, un solo lugar" (20px, regular)
   - Search bar prominente: input grande con icon, placeholder dinámico
   - CTA button: "Explorar restaurantes" (primary, lg)
   - Background: gradient sutil o imagen hero con overlay
   - Altura: 60vh (viewport height)

2. **Categories Section** - Navegación visual:
   - Grid horizontal scrollable (no wrapping)
   - Cards de categoría: imagen circular, nombre abajo
   - Hover effect: scale 1.05 + shadow
   - 8-10 categorías visibles: Pizza, Burgers, Sushi, etc.
   - Scroll smooth con botones prev/next en desktop

3. **Featured Restaurants** - Restaurantes destacados:
   - Grid responsivo: 1 col (mobile), 2 cols (tablet), 3 cols (desktop)
   - Restaurant card premium:
     * Imagen: aspect ratio 16:9, lazy loading, placeholder
     * Logo del restaurante: posición absolute, circular, -20px desde top
     * Info: nombre (18px bold), rating (estrella + número), delivery time
     * Tags: tipo de cocina, "Envío gratis", "Nuevo"
     * Hover: elevación suave, scale 1.02
   - "Ver más" button al final

4. **How It Works** - Proceso simple:
   - 3 pasos con iconos grandes
   - Números grandes (1, 2, 3) con círculos
   - Título y descripción por paso
   - Layout horizontal en desktop, vertical en mobile

5. **CTA Final** - Llamado a la acción:
   - Fondo de color (gradient)
   - Texto grande: "¿Listo para ordenar?"
   - Button: "Comenzar ahora"
   - Padding generoso

Layout general:

- Max-width container: 1280px
- Padding horizontal: 24px (mobile), 48px (desktop)
- Secciones separadas con 96px vertical spacing
- Smooth scroll behavior

=== RESTAURANT PAGE (/restaurant/:slug) ===
Diseño tipo Uber Eats:

1. **Restaurant Header** - Hero con info:
   - Cover image: full width, height 300px, gradient overlay
   - Info overlay (bottom):
     * Logo: circular, 100px, border blanco
     * Nombre: 32px bold, blanco con shadow
     * Rating, reviews count, delivery time, delivery cost
     * Open/Closed badge: verde/rojo con dot
   - Sticky después del scroll: compacto con info principal

2. **Menu Navigation** - Categorías sticky:
   - Barra horizontal sticky: categorías del menú
   - Scroll spy: highlight activo según scroll
   - Smooth scroll al hacer click
   - Sombra cuando está sticky

3. **Menu Items Grid** - Productos organizados:
   - Por categoría con headers grandes
   - Grid 1 col (mobile), 2 cols (desktop grande)
   - Item card:
     * Layout: imagen derecha (30%), info izquierda (70%)
     * Nombre: 18px semibold
     * Descripción: 14px gray, 2 líneas truncate
     * Precio: 16px bold, verde
     * "Agregar" button: circular con +, esquina inferior derecha
     * Imagen: aspect ratio 1:1, lazy loading
   - Hover: sombra + scale sutil
   - Animación de entrada: fade + slide staggered

4. **Floating Cart Button** - Carrito visible:
   - Posición: bottom fixed (mobile), right fixed (desktop)
   - Muestra: "Ver carrito (3)" + precio total
   - Badge con count
   - Click: abre modal de carrito
   - Animación: bounce cuando se agrega item

Características adicionales:
- Búsqueda de productos
- Filtros: vegetariano, sin gluten, picante, etc.

- Ordenar por: popularidad, precio, nombre
- Loading states con skeletons

=== CHECKOUT PAGE (/checkout) ===
Flujo premium en pasos:

1. **Progress Stepper** - Indicador visual:
   - Pasos: Entrega → Pago → Confirmar
   - Visualización: números en círculos con líneas conectoras
   - Step activo: filled, completado: check, pendiente: outline

2. **Step 1: Delivery Info**:
   - Form elegante con:
     * Dirección completa (autocomplete con Google Places)
     * Número de teléfono (formato automático)
     * Instrucciones de entrega (textarea)
     * Mapa interactivo mostrando ubicación
   - Direcciones guardadas: cards seleccionables

3. **Step 2: Payment Method**:
   - Métodos en cards:
     * Tarjeta de crédito/débito (Stripe Elements integrado)
     * Efectivo (con cambio opcional)
     * Billetera (si aplica)
   - Métodos guardados con últimos 4 dígitos
   - Agregar nueva tarjeta: modal

4. **Step 3: Order Review**:
   - Resumen del pedido:
     * Items con imagen pequeña, nombre, cantidad, precio
     * Subtotal
     * Delivery fee (basado en zona)
     * Cupón (si aplica) con descuento verde
     * Total grande y bold
   - Tiempo estimado de entrega
   - Notes especiales
   - "Realizar pedido" button: full width, primary

5. **Sidebar** - Order Summary (desktop only):
   - Sticky: se mantiene visible al scrollear
   - Restaurante info compacta
   - Items list resumido
   - Totales
   - CTA button

Validación y UX:

- Validación en tiempo real
- Error messages claros
- Disable next button hasta completar
- Loading states durante procesamiento
- Animaciones de transición entre pasos

=== ORDER TRACKING (/order/:id) ===
Seguimiento en tiempo real:

1. **Order Status Timeline** - Visual prominente:
   - Estados: Confirmado → Preparando → En camino → Entregado
   - Timeline vertical con iconos
   - Estado actual: animado (pulsing)
   - Timestamp por cada estado
   - Colores: gray (pendiente), green (completado), blue (actual)

2. **Live Map** - Mapa en tiempo real:
   - Mapbox GL o Google Maps
   - Marcadores: restaurante, ubicación delivery, tu ubicación
   - Ruta trazada con polyline
   - Animación: marcador del delivery moviéndose suavemente
   - Zoom automático para ver toda la ruta
   - Full width, height 400px

3. **Delivery Info Card** - Info del repartidor:
   - Avatar, nombre
   - Rating
   - Vehículo (moto, bici, auto)
   - Botones: Llamar, Mensaje
   - Tiempo estimado actualizado en tiempo real

4. **Order Details Card** - Resumen del pedido:
   - Restaurante: logo, nombre
   - Items list colapsable
   - Total pagado
   - Método de pago usado

5. **Support Section** - Ayuda rápida:
   - FAQs colapsables
   - "¿Problema con tu pedido?" button → modal con opciones
   - Chat support button

Características en tiempo real:
- Socket.io para actualizaciones instantáneas
- Notificaciones push cuando cambia estado
- Auto-refresh cada 30s como fallback
- Animaciones suaves para cambios de estado

1.4 Componentes Específicos de Negocio

Instrucciones para Copilot:

Crea componentes específicos premium en /frontend/src/components/business/:

1. **RestaurantCard.tsx** - Card de restaurante elegante:
   - Imagen con lazy loading + placeholder
   - Logo circular overlay
   - Rating con estrellas + número
   - Delivery time y costo
   - Tags: "Envío gratis", "Nuevo", "Popular"
   - Hover effect: elevación + scale
   - Click: navigate a /restaurant/:slug

2. **MenuItemCard.tsx** - Card de producto del menú:
   - Imagen cuadrada (derecha en mobile, arriba en desktop)
   - Nombre, descripción (truncate 2 líneas)
   - Precio destacado
   - Botón agregar: circular con +
   - Modal al agregar: personalizar opciones
   - Loading state al agregar

3. **CartItem.tsx** - Item en el carrito:
   - Thumbnail pequeño
   - Nombre, opciones seleccionadas
   - Quantity picker: - [number] +
   - Precio por unidad y subtotal
   - Botón remover (icono trash)
   - Animación al remover: slide out

4. **OrderCard.tsx** - Card de pedido en historial:
   - Header: fecha, número de orden, estado badge
   - Restaurante info con logo
   - Items count
   - Total
   - Botones: Ver detalle, Reordenar, Calificar
   - Click: expandir con detalles completos

5. **ReviewCard.tsx** - Card de reseña:
   - Avatar del usuario
   - Nombre, fecha
   - Rating con estrellas
   - Texto de la reseña
   - Imágenes subidas (si hay)
   - Respuesta del restaurante (si hay)
   - Botones: Útil (thumbs up con count)

6. **SearchBar.tsx** - Barra de búsqueda premium:
   - Input grande con icono search
   - Placeholder animado
   - Auto-complete dropdown:
     * Restaurantes sugeridos
     * Productos populares
     * Categorías
   - Historial de búsquedas
   - Clear button (X)
   - Loading state

7. **CategoryFilter.tsx** - Filtros de categoría:
   - Pills horizontales scrollables
   - "Todos" siempre primero
   - Active state: filled con color
   - Smooth scroll al hacer click
   - Count de items por categoría

8. **EmptyState.tsx** - Estados vacíos elegantes:
   - Icono ilustrativo grande
   - Título y descripción
   - CTA button (si aplica)
   - Diferentes variantes:
     * Carrito vacío
     * Sin pedidos
     * Sin favoritos
     * Sin resultados de búsqueda

1.5 Animaciones y Transiciones Avanzadas

Instrucciones para Copilot:

Implementa animaciones premium con Framer Motion en /frontend/src/utils/animations.ts:

1. **Page Transitions**:
   - Fade + slide para cambio de páginas
   - Direcciones: left/right según navegación forward/back
   - Spring physics suaves
   - Duración: 300-400ms

2. **List Animations**:
   - Stagger children: 50ms delay entre items
   - Fade + slide up para listas
   - Exit animations al filtrar

3. **Modal Animations**:
   - Backdrop: fade in/out
   - Content: scale + fade (desktop), slide from bottom (mobile)
   - Smooth spring physics

4. **Micro-interactions**:
   - Button hover: scale 1.02
   - Button press: scale 0.98
   - Card hover: elevación suave
   - Icon animations al interactuar

5. **Loading States**:
   - Skeleton shimmer effect
   - Spinner con smooth rotation
   - Progress bars con smooth filling

Configuraciones de Framer Motion:
- Usar spring physics: { type: "spring", stiffness: 300, damping: 30 }
- Easing custom: [0.4, 0, 0.2, 1] (easeInOut)
- layoutId para morphing entre elementos

🔗 FASE 2: MEJORAS BACKEND Y ARQUITECTURA

Objetivo: Optimizar backend para scale y performance

2.1 Optimización de API

Instrucciones para Copilot:

Mejora el backend en /backend/src/:

1. **API Rate Limiting** - Protección contra abuso:
   - Implementar rate limiting con express-rate-limit
   - Límites por endpoint:
     * Auth: 5 requests/15min
     * API general: 100 requests/15min
     * Upload: 10 requests/hour
   - Headers informativos: X-RateLimit-Remaining, X-RateLimit-Reset

2. **Caching Strategy** - Redis para performance:
   - Instalar y configurar Redis
   - Cachear:
     * Lista de restaurantes: 5 minutos
     * Menú de restaurante: 15 minutos
     * Categorías: 1 hora
     * Config del sistema: 1 día
   - Cache invalidation al actualizar datos
   - Implementar cache warming

3. **Pagination Optimizada**:
   - Implementar cursor-based pagination (mejor que offset)
   - Parámetros: limit (default 20, max 100), cursor
   - Response format:
     ```
     {
       data: [...],
       pagination: {
         nextCursor: "...",
         hasMore: boolean
       }
     }
     ```

4. **Search Optimization**:
   - Índices MongoDB para búsqueda:
     * Restaurantes: text index en nombre, descripción
     * Productos: text index en nombre, descripción
   - Full-text search con scoring
   - Fuzzy matching para typos
   - Search suggestions con agregación

5. **Image Optimization**:
   - Configurar Cloudinary transformations:
     * Thumbnails: 200x200, quality 80

```
   * Card images: 800x600, quality 85
   * Hero images: 1920x1080, quality 90
  - Lazy loading URLs
  - WebP format automático
  - Responsive image URLs

6. **Error Handling Mejorado**:
  - Error classes custom:
   * ValidationError (400)
   * AuthenticationError (401)
   * AuthorizationError (403)
   * NotFoundError (404)
   * ConflictError (409)
   * ServerError (500)
  - Error response consistente:
   {
    error: {
     code: "ERR_CODE",
     message: "User-friendly message",
     details: {} // Optional
    }
   }
  - Logging con Winston
  - Sentry para error tracking en producción

7. **Request Validation**:
  - Validar todos los inputs con Joi
  - Sanitizar datos con express-mongo-sanitize
  - Helmet para security headers
  - CORS configurado correctamente

8. **Database Optimization**:
  - Índices en campos frecuentemente consultados
  - Populate selectivo (solo campos necesarios)
  - Lean queries cuando no se necesita documento completo
  - Aggregation pipelines optimizados
  - Connection pooling configurado
```

2.2 Real-time Mejorado

Instrucciones para Copilot:

Mejora Socket.io en /backend/src/sockets/:

1. **Namespace Organization**:
   - /customer - Eventos para clientes
   - /restaurant - Eventos para restaurantes
   - /admin - Eventos para admin
   - /delivery - Eventos para repartidores

2. **Authentication**:
   - Middleware para autenticar socket connections
   - Verificar JWT token
   - Attach user data a socket.data

3. **Room Management**:
   - Rooms por orden: order:{orderId}
   - Rooms por restaurante: restaurant:{restaurantId}
   - Join/leave automático

4. **Event Types Definidos**:
   Customer events:
   - order:created
   - order:status_update
   - order:assigned
   - delivery:location_update
   - delivery:arrived

   Restaurant events:
   - order:new
   - order:cancelled

   Admin events:
   - stats:update
   - alert:new

5. **Optimization**:
   - Emit solo a rooms necesarios
   - Throttle location updates (cada 5 segundos)
   - Compress messages grandes
   - Disconnect inactive sockets

6. **Error Handling**:
   - Try-catch en todos los handlers

```
- Log errors
- Emit error events al client
```

2.3 Pagos y Billing Mejorado

Instrucciones para Copilot:

```
Mejora el sistema de pagos en /backend/src/services/paymentService.js:

1. **Stripe Integration Premium**:
   - Payment Intents con metadata completa
   - Webhook handling robusto
   - Retry logic para failed payments
   - Refund handling automático

2. **Multi-Currency** (preparar para futuro):
   - Estructura para soportar COP, VES, USD
   - Conversion rates (aunque use solo COP ahora)

3. **Invoice Generation**:
   - Generar PDF invoice por orden
   - Enviar por email
   - Almacenar en Cloudinary

4. **Payment Methods**:
   - Stripe (tarjetas)
   - Efectivo (cash on delivery)
   - Billing addresses guardadas
   - Default payment method

5. **Security**:
   - PCI compliance con Stripe
   - No almacenar datos sensibles
   - Audit log de transacciones
```

🎯 FASE 3: FEATURES PREMIUM ADICIONALES

Objetivo: Características que justifican $100k+

3.1 Sistema de Recomendaciones AI

Instrucciones para Copilot:

```
Implementa recomendaciones inteligentes en /backend/src/services/recommendationService.js:

1. **Collaborative Filtering Simple**:
   - Basado en historial de pedidos
   - "Clientes que ordenaron X también ordenaron Y"
   - Algoritmo: Jaccard similarity entre usuarios

2. **Content-Based Filtering**:
   - Basado en categorías favoritas del usuario
   - Tags de productos
   - Preferencias dietéticas

3. **Hybrid Approach**:
   - Combinar ambos métodos
   - Pesos ajustables

4. **Trending Items**:
   - Productos más ordenados en últimas 7 días
   - Por categoría y global
   - Actualizado cada hora con cron job

5. **Personalized Homepage**:
   - Restaurantes recomendados
   - "Volver a pedir de..." (reorder rápido)
   - "Basado en tus gustos"

Frontend: Secciones en homepage con estos datos
```

3.2 Programa de Fidelización

Instrucciones para Copilot:

Crea sistema de puntos y rewards en /backend/src/models/Loyalty.js:

1. **Points System**:
   - Schema: userId, points, tier (Bronze, Silver, Gold, Platinum)
   - Ganar puntos:
     * 1 punto por cada $10 pesos gastados
     * Bonus por primera orden: 100 puntos
     * Bonus por referidos: 50 puntos
   - Redimir: 100 puntos = $10 descuento

2. **Tiers System**:
   - Bronze: 0-999 puntos (beneficios base)
   - Silver: 1000-2999 puntos (5% descuento extra)
   - Gold: 3000-5999 puntos (10% descuento, envío gratis ocasional)
   - Platinum: 6000+ puntos (15% descuento, envío siempre gratis)

3. **Challenges**:
   - Ordena 5 veces este mes: 200 puntos bonus
   - Prueba 3 restaurantes nuevos: 150 puntos
   - Gamification simple

4. **Frontend**:
   - Dashboard de puntos en profile
   - Progress bar a siguiente tier
   - Historial de puntos ganados/gastados
   - Rewards catalog

Endpoints:
- GET /api/loyalty/points
- GET /api/loyalty/history
- POST /api/loyalty/redeem
- GET /api/loyalty/challenges

3.3 Social Features

Instrucciones para Copilot:

Implementa características sociales en /backend/src/models/ y /frontend/src/pages/:

1. **Referral Program**:
   - Cada usuario tiene código único
   - Compartir código: social share buttons
   - Recompensas: $20 para ambos al completar primera orden
   - Tracking de referidos

2. **Social Proof**:
   - "X personas ordenaron de aquí hoy"
   - "Trending ahora en tu zona"
   - Reviews destacadas
   - Fotos de usuarios

3. **Lists/Collections** (futuro MVP):
   - Crear listas: "Mis favoritos", "Para probar"
   - Compartir listas
   - Follow listas de otros

4. **Feed de Actividad** (futuro):
   - Nuevos restaurantes
   - Ofertas especiales
   - Achievements desbloqueados

3.4 Soporte al Cliente Premium

Instrucciones para Copilot:

Sistema de soporte en /backend/src/controllers/supportController.js:

1. **Live Chat** (Socket.io):
   - Chat 1-on-1 con soporte
   - Typing indicators
   - Read receipts
   - File sharing
   - Canned responses para soporte

2. **Ticket System**:
   - Crear ticket: categoría, prioridad, descripción
   - Estados: Open, In Progress, Resolved, Closed
   - SLA tracking
   - Email notifications

3. **Help Center**:
   - FAQs categorizadas
   - Búsqueda
   - Artículos útiles
   - Videos tutoriales

4. **In-App Feedback**:
   - Widget flotante "¿Necesitas ayuda?"
   - Quick actions: problema con orden, pregunta general, reportar bug
   - Screenshot capture automático

Frontend:
- /support página
- Chat widget flotante
- Help modal

3.5 Analytics y Admin Dashboard Avanzado

Instrucciones para Copilot:

Mejora admin dashboard en /frontend/src/pages/admin/:

1. **Dashboard Overview** - KPIs en tiempo real:
   - Cards con métricas:
     * Revenue hoy/semana/mes (con % change)
     * Órdenes totales (con gráfica mini)
     * Usuarios activos
     * Tasa de conversión
     * Tiempo promedio de entrega
     * Rating promedio de plataforma

2. **Charts Premium** con Recharts:
   - Revenue chart: line chart últimos 30 días
   - Orders chart: bar chart por día de la semana
   - Top restaurants: horizontal bar chart
   - Category distribution: pie chart
   - User growth: area chart
   - Delivery times: histogram

3. **Real-time Monitoring**:
   - Mapa con órdenes activas
   - Lista live de órdenes (auto-refresh)
   - Alertas: órdenes retrasadas, cancelaciones

4. **Reports Generation**:
   - Date range picker
   - Filtros: por restaurante, categoría, zona
   - Export a PDF/Excel
   - Scheduled reports por email

5. **User Management**:
   - Tabla con búsqueda y filtros
   - Acciones: ver perfil, suspender, cambiar rol
   - Activity log por usuario

6. **Restaurant Management**:
   - Aprobar/rechazar nuevos restaurantes
   - Editar info de restaurantes
   - Estadísticas por restaurante
   - Comisiones configurables

Importante: Usar Recharts o Victory para gráficas, table con pagination y sorting

🚀 FASE 4: MOBILE APP (REACT NATIVE)

Objetivo: App nativa para iOS y Android

4.1 Setup React Native

Instrucciones para Copilot:

Crea app móvil en /mobile/ con React Native + Expo:

1. **Inicializar Proyecto**:
```bash
  npx create-expo-app RapidEats --template
  cd RapidEats
  npx expo install expo-router react-native-safe-area-context react-native-screens expo-linking expo-constants expo-status-bar
```

2. **Estructura de Carpetas**:
  /mobile
    /src
      /components (reutilizar lógica de web)
      /screens
      /navigation
      /services
      /hooks
      /contexts
      /utils
      /assets
    /app (Expo Router)
    app.json
    package.json

3. **Navigation** con Expo Router:
  - Tabs: Home, Search, Orders, Profile
  - Stacks: dentro de cada tab
  - Modals: Restaurant detail, Checkout

4. **UI Components** con React Native Paper o NativeBase:
  - Reutilizar lógica de componentes web
  - Adaptar estilos a nativos
  - Platform-specific code cuando necesario

5. **State Management**:
  - Redux Toolkit (mismo que web)
  - Redux Persist para offline

4.2 Features Mobile-First

Instrucciones para Copilot:

Implementa features mobile en /mobile/src/:

1. **Geolocation**:
   - expo-location para ubicación actual
   - Detección automática de zona
   - Mapa interactivo para delivery

2. **Push Notifications**:
   - expo-notifications
   - Request permissions
   - Handle notifications en foreground/background
   - Deep linking a orden cuando se toca notif

3. **Camera/Gallery**:
   - expo-image-picker
   - Tomar fotos de reseñas
   - Avatar upload

4. **Offline Support**:
   - Redux Persist
   - Queue acciones cuando offline
   - Sync al reconectar

5. **Biometric Auth**:
   - expo-local-authentication
   - Fingerprint/Face ID para login rápido

6. **App Theming**:
   - Dark/Light mode
   - Seguir theme del sistema
   - Smooth transitions

7. **Performance**:
   - Lazy loading de imágenes
   - FlatList para listas largas
   - Memoization de componentes
   - React.memo, useMemo, useCallback

Nota: Muchos componentes pueden shared entre web y mobile usando responsive

🎨 FASE 5: DETALLES PREMIUM QUE MARCAN DIFERENCIA

Objetivo: Micro-interacciones y polish que elevan la experiencia

5.1 Micro-interacciones Premium

Instrucciones para Copilot:

Implementa estas micro-interacciones en componentes relevantes:

1. **Haptic Feedback** (web y mobile):
   - Vibración sutil al agregar al carrito
   - navigator.vibrate(10) en web
   - Haptics.selectionAsync() en mobile

2. **Pull to Refresh**:
   - En listas: restaurantes, órdenes, productos
   - Animación de loading custom
   - Smooth spring animation

3. **Swipe Actions**:
   - Swipe to delete en cart items
   - Swipe to reorder en order history
   - Reveal buttons con smooth animation

4. **Gesture Animations**:
   - Long press para preview
   - Pinch to zoom en imágenes
   - Drag to reorder (favoritos)

5. **Loading States Creativos**:
   - Skeleton screens que morphean a contenido real
   - Progress indicators creativos
   - Animated placeholders

6. **Success Animations**:
   - Checkmark animado al completar acción
   - Confetti al completar primer pedido
   - Lottie animations para celebraciones

7. **Empty States Delightful**:
   - Ilustraciones custom SVG animadas
   - Mensajes amigables y humor sutil
   - CTAs claros

8. **Error States Elegantes**:
   - Ilustraciones de error amigables
   - Mensajes claros sin jargon técnico
   - Sugerencias de qué hacer
   - Retry buttons obvios

Usa Lottie para animaciones complejas: https://lottiefiles.com/

5.2 Accessibility (A11y) Premium

Instrucciones para Copilot:

Implementa accesibilidad nivel AAA en todos los componentes:

1. **Semantic HTML**:
   - Usar tags correctos: <nav>, <main>, <article>, <button>, <a>
   - Headings jerárquicos (h1 → h2 → h3)
   - Landmarks ARIA cuando necesario

2. **Keyboard Navigation**:
   - Tab order lógico
   - Focus visible (outline custom)
   - Shortcuts de teclado: / para search, ESC para cerrar modals
   - Skip to content link

3. **Screen Reader Support**:
   - ARIA labels en todos los elementos interactivos
   - Alt text descriptivo en imágenes
   - aria-live para notificaciones
   - aria-expanded, aria-selected en componentes

4. **Color Contrast**:
   - WCAG AAA compliance (ratio 7:1 para texto normal)
   - Verificar con herramientas: https://webaim.org/resources/contrastchecker/
   - No usar solo color para transmitir info

5. **Focus Management**:
   - Trap focus en modals
   - Return focus al cerrar modal
   - Focus visible en todos los estados

6. **Responsive Typography**:
   - Usar rem/em en vez de px
   - Permitir zoom hasta 200%
   - Line height adecuado (1.5 mínimo)

7. **Forms Accessibility**:
   - Labels asociados a inputs
   - Error messages linked con aria-describedby
   - Required fields marcados
   - Autocomplete attributes

Testing: usar axe DevTools extension

5.3 Performance Optimization Premium

Instrucciones para Copilot:

Optimizaciones de performance en frontend y backend:

=== FRONTEND ===

1. **Code Splitting**:
   - Lazy load rutas: React.lazy + Suspense
   - Lazy load componentes pesados
   - Dynamic imports para libraries grandes

2. **Image Optimization**:
   - Lazy loading con Intersection Observer
   - Responsive images con srcset
   - WebP con fallback
   - Blur placeholder con BlurHash

3. **Bundle Optimization**:
   - Tree shaking automático con Vite
   - Analizar bundle con rollup-plugin-visualizer
   - Code split por routes
   - Eliminar unused dependencies

4. **React Optimization**:
   - React.memo para componentes puros
   - useMemo para cálculos pesados
   - useCallback para funciones en props
   - React DevTools Profiler para encontrar bottlenecks

5. **State Management**:
   - Redux Toolkit con RTK Query para caching
   - Selector memoization con reselect
   - Normalizar state shape

6. **Network Optimization**:
   - HTTP/2 o HTTP/3
   - Gzip/Brotli compression
   - CDN para assets estáticos
   - Service Worker para caching offline

7. **Rendering Optimization**:
   - Virtualization para listas largas (react-window)
   - Debounce search inputs
   - Throttle scroll events
   - RequestIdleCallback para tareas no urgentes

=== BACKEND ===

1. **Database**:
   - Índices en campos frecuentemente consultados
   - Connection pooling
   - Lean queries
   - Aggregation pipelines optimizados

2. **Caching**:
   - Redis para data frecuentemente leída
   - Cache warming en startup
   - Invalidation strategy clara

3. **API**:
   - Compression middleware (gzip)
   - Response pagination
   - GraphQL para queries flexibles (futuro)
   - Rate limiting

Targets:
- LCP (Largest Contentful Paint): < 2.5s
- FID (First Input Delay): < 100ms
- CLS (Cumulative Layout Shift): < 0.1
- Lighthouse score: 90+

5.4 SEO y Marketing

Instrucciones para Copilot:

Optimizaciones SEO en /frontend:

1. **Meta Tags** con react-helmet-async:
   - Title dinámico por página
   - Meta description
   - Open Graph tags (Facebook, WhatsApp)
   - Twitter Card tags
   - Canonical URLs

2. **Structured Data** (JSON-LD):
   - Schema.org markup:
     * Restaurant
     * LocalBusiness
     * Review
     * AggregateRating
     * MenuItem
   - Validar con Google Rich Results Test

3. **Sitemap**:
   - Generar sitemap.xml dinámico
   - Incluir todas las rutas públicas
   - Submit a Google Search Console

4. **Robots.txt**:
   - Permitir crawling de páginas públicas
   - Bloquear admin, checkout, profile

5. **Performance = SEO**:
   - Core Web Vitals optimizados
   - Mobile-first responsive
   - Fast page load

6. **Content Strategy**:
   - Blog con recetas, tips (futuro)
   - Landing pages por ciudad/zona
   - Rich snippets para restaurantes

7. **Analytics**:
   - Google Analytics 4
   - Event tracking: pedidos, registros, conversiones
   - Funnel analysis

8. **Social Sharing**:

- Share buttons
- Pre-filled messages
- Referral tracking

Importante: Como es SPA, considerar SSR con Next.js en futuro para mejor SEO

📱 FASE 6: PWA Y OFFLINE EXPERIENCE

Objetivo: Progressive Web App con experiencia offline

6.1 PWA Setup

Instrucciones para Copilot:

Convertir en PWA premium en /frontend:

1. **Service Worker** con Workbox:
   - Instalar workbox-webpack-plugin
   - Estrategias de caching:
     * Network First: API calls
     * Cache First: assets estáticos (images, fonts)
     * Stale While Revalidate: data que puede estar desactualizada
   - Precache: shell de la app, assets críticos
   - Runtime caching: imágenes, API responses

2. **Manifest.json** premium:
```json
   {
     "name": "RapidEats - Food Delivery",
     "short_name": "RapidEats",
     "description": "Pide comida de tus restaurantes favoritos",
     "start_url": "/",
     "display": "standalone",
     "background_color": "#000000",
     "theme_color": "#06C167",
     "orientation": "portrait",
     "icons": [
       {
         "src": "/icons/icon-72x72.png",
         "sizes": "72x72",
         "type": "image/png",
         "purpose": "any maskable"
       },
       // ... más tamaños: 96, 128, 144, 152, 192, 384, 512
     ]
   }
```

3. **App Install Prompt**:
   - Detectar beforeinstallprompt event
   - Custom install button con buen UX
   - Show después de 2+ visitas
   - Animación smooth

4. **Offline Experience**:
   - Offline page custom (no el "dinosaurio")
   - Detectar online/offline status

- Queue acciones offline (agregar a carrito)
- Sync cuando vuelve online
- Toast notification de status

5. **Background Sync**:
- Sync pedidos fallidos
- Upload reseñas offline
- Background Sync API

6. **Push Notifications**:
- Request permission con buen timing
- Custom notification style
- Actions en notifications
- Click handler con deep linking

7. **App Update Flow**:
- Detectar nuevo service worker
- Toast: "Nueva versión disponible"
- Reload button
- Smooth transition

Icons: Generar con https://realfavicongenerator.net/
Testing: Lighthouse audit, PWA checklist

🔐 FASE 7: SEGURIDAD Y COMPLIANCE

Objetivo: Seguridad enterprise-grade

7.1 Security Hardening

Instrucciones para Copilot:

Implementa seguridad premium en backend:

1. **Authentication Security**:
   - bcrypt con salt rounds: 12
   - JWT con refresh tokens
   - Rotate refresh tokens
   - Token blacklisting en logout
   - Session management
   - Account lockout después de 5 intentos fallidos

2. **Authorization**:
   - Role-based access control (RBAC)
   - Middleware de permisos
   - Resource-level permissions
   - Audit log de accesos

3. **Input Validation & Sanitization**:
   - Joi para validation
   - express-mongo-sanitize
   - xss-clean para XSS prevention
   - express-validator
   - Trim y lowercase where appropriate

4. **Security Headers** con Helmet:
   - Content-Security-Policy
   - X-Frame-Options: DENY
   - X-Content-Type-Options: nosniff
   - Referrer-Policy
   - Permissions-Policy

5. **HTTPS Everywhere**:
   - Force HTTPS en producción
   - HSTS header
   - Secure cookies (httpOnly, secure, sameSite)

6. **Rate Limiting**:
   - Global: 100 req/15min
   - Auth endpoints: 5 req/15min
   - Expensive operations: 10 req/hour

7. **SQL/NoSQL Injection Prevention**:
   - Parameterized queries
   - ORM/ODM (Mongoose)

- Input sanitization

8. **CORS Configuration**:
   - Whitelist origins
   - Credentials: true
   - Specific methods y headers

9. **File Upload Security**:
   - Validar MIME types
   - Límite de tamaño
   - Scan for malware (ClamAV en futuro)
   - Store en CDN (Cloudinary), no en server

10. **Logging & Monitoring**:
    - Winston para logging
    - Log levels: error, warn, info, debug
    - No loggear datos sensibles
    - Rotate logs
    - Sentry para error tracking

11. **Environment Variables**:
    - Nunca commitear .env
    - Diferentes configs: dev, staging, prod
    - Secrets management (AWS Secrets Manager en futuro)

12. **Dependencies**:
    - npm audit regularmente
    - Dependabot para updates
    - Pinned versions en package.json

13. **API Security**:
    - API versioning: /api/v1/
    - Deprecation warnings
    - API documentation (Swagger)

Testing:
- OWASP ZAP para security scanning
- Manual penetration testing
- Regular security audits

7.2 GDPR/Privacy Compliance

Instrucciones para Copilot:

Implementa compliance de privacidad:

1. **Privacy Policy**:
   - Página /privacy con política completa
   - Qué datos recopilamos
   - Cómo los usamos
   - Cookies policy
   - Derechos del usuario

2. **Terms of Service**:
   - Página /terms
   - Reglas de uso
   - Limitaciones de responsabilidad
   - Política de cancelación/reembolso

3. **Cookie Consent**:
   - Banner de cookies obligatorio
   - Categorías: Necesarias, Funcionales, Analytics, Marketing
   - Opción de rechazar no-esenciales
   - Guardar preferencias

4. **User Data Rights**:
   - Endpoint: GET /api/user/data (exportar datos)
   - Endpoint: DELETE /api/user/account (right to be forgotten)
   - Email confirmation para delete
   - 30 días grace period

5. **Data Minimization**:
   - Solo recopilar datos necesarios
   - Anonimizar analytics data
   - Regular data cleanup (ordenes antiguas)

6. **Consent Management**:
   - Opt-in para marketing emails
   - Opt-in para push notifications
   - Preferencias granulares

7. **Audit Trail**:
   - Log accesos a datos sensibles
   - Log cambios en profile
   - Retention por 2 años

Importante: Consultar con abogado para compliance real

🧪 FASE 8: TESTING Y QA

Objetivo: Código robusto y sin bugs

8.1 Testing Strategy

Instrucciones para Copilot:

Implementa testing comprehensivo:

=== FRONTEND TESTING ===

1. **Unit Tests** con Vitest + React Testing Library:
   - Test todos los componentes:
     * Render sin errores
     * Props correctas
     * Event handlers
     * Conditional rendering
   - Test custom hooks
   - Test utilities y helpers
   - Coverage target: 80%+

Ejemplo estructura:
```typescript
describe('Button', () => {
  it('renders correctly', () => { ... })
  it('calls onClick when clicked', () => { ... })
  it('shows loading state', () => { ... })
  it('is disabled when disabled prop', () => { ... })
})
```

2. **Integration Tests**:
   - Test flujos completos:
     * Login → Browse → Add to cart → Checkout
     * Order tracking
     * Review submission
   - Mock API calls con MSW (Mock Service Worker)

3. **E2E Tests** con Playwright:
   - Critical paths:
     * User registration
     * Place order
     * Track order
     * Write review
   - Run en CI/CD
   - Screenshots en failures

4. **Visual Regression Testing** con Chromatic:
   - Storybook para componentes
   - Automatic visual diffs

- Review changes en PRs

=== BACKEND TESTING ===

1. **Unit Tests** con Jest:
   - Test controllers:
     * Happy paths
     * Error cases
     * Validation
   - Test services:
     * Business logic
     * Mocks de dependencies
   - Test utilities

2. **Integration Tests**:
   - Test API endpoints:
     * Requests y responses
     * Authentication
     * Authorization
     * Database interactions
   - Test real database (MongoDB Memory Server)

3. **Load Testing** con k6:
   - Simular carga:
     * 100 users concurrentes
     * 1000 requests/minute
   - Identificar bottlenecks
   - Response time targets: < 200ms

Ejemplo test de endpoint:
```javascript
describe('POST /api/orders', () => {
  it('creates order successfully', async () => {
    const res = await request(app)
      .post('/api/orders')
      .set('Authorization', `Bearer ${token}`)
      .send(orderData)
      .expect(201);

    expect(res.body).toHaveProperty('orderId');
  });

  it('returns 400 for invalid data', async () => { ... });
  it('returns 401 without auth', async () => { ... });
```

```
});
```

Testing commands en package.json:
```json
{
  "scripts": {
    "test": "vitest",
    "test:coverage": "vitest --coverage",
    "test:e2e": "playwright test",
    "test:api": "jest --config jest.config.api.js"
  }
}
```

8.2 CI/CD Pipeline

Instrucciones para Copilot:

Setup CI/CD con GitHub Actions en .github/workflows/:

1. **ci.yml** - Continuous Integration:
```yaml
name: CI
on: [push, pull_request]
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: 18
      - run: npm ci
      - run: npm run lint
      - run: npm run test:coverage
      - run: npm run build
```

2. **cd.yml** - Continuous Deployment:
```yaml
name: CD
on:
  push:
    branches: [main]
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - Deploy frontend a Vercel
      - Deploy backend a Railway/Render
      - Run database migrations
      - Invalidate CDN cache
```

3. **Quality Gates**:
- Tests must pass
- Coverage > 80%
- No linting errors
- Build successful
- Performance budget (Lighthouse score > 90)

4. **Branch Protection**:
   - Require PR reviews
   - Require status checks
   - No direct push to main

📊 FASE 9: ANALYTICS Y MONITOREO

Objetivo: Data-driven decisions

9.1 Analytics Implementation

Instrucciones para Copilot:

Implementa analytics comprehensivo:

1. **Google Analytics 4**:
   - Setup en /frontend/src/utils/analytics.ts
   - Events tracking:
     * page_view (automático)
     * search (query, category)
     * view_item (product, restaurant)
     * add_to_cart (item, quantity, price)
     * begin_checkout (cart_value, items_count)
     * purchase (order_id, value, items)
     * sign_up (method)
     * login (method)
   - User properties:
     * user_role
     * favorite_category
     * total_orders
     * lifetime_value

2. **Custom Analytics Dashboard**:
   - Endpoint: /api/analytics/*
   - Métricas:
     * DAU/MAU (Daily/Monthly Active Users)
     * Conversion rate
     * Average order value
     * Customer lifetime value
     * Churn rate
     * Retention cohorts
   - Visualizar en admin dashboard

3. **Heatmaps** con Hotjar o Clarity:
   - Ver dónde hacen click usuarios
   - Session recordings
   - Identificar pain points

4. **A/B Testing** con Split.io:
   - Test diferentes CTAs
   - Test layouts
   - Test pricing strategies

5. **Funnel Analysis**:
   - Homepage → Restaurant → Cart → Checkout → Order

- Identificar drop-off points
- Optimizar conversion

9.2 Monitoring y Alerting

Instrucciones para Copilot:

Implementa monitoring robusto:

1. **APM** (Application Performance Monitoring):
   - New Relic o Datadog
   - Métricas:
     * Response times
     * Error rates
     * Throughput
     * Database query times
     * External service calls

2. **Error Tracking** con Sentry:
   - Frontend y backend
   - Source maps para stack traces
   - Release tracking
   - User context en errors
   - Breadcrumbs

3. **Logging** con Winston + Logtail:
   - Structured logging
   - Log levels
   - Searchable logs
   - Retention policy

4. **Uptime Monitoring** con UptimeRobot:
   - Ping cada 5 minutos
   - Email/SMS alerts
   - Status page público

5. **Database Monitoring**:
   - MongoDB Atlas monitoring
   - Slow query alerts
   - Connection pool monitoring
   - Disk space alerts

6. **Alerting**:
   - Slack/Discord webhooks
   - Alert rules:
     * Error rate > 1%
     * Response time > 1s
     * CPU > 80%
     * Memory > 85%
     * Disk > 90%

7. **Health Checks**:
   - Endpoint: GET /health
   - Return: status, uptime, version, db status
   - Use en load balancer

🎓 FASE 10: DOCUMENTACIÓN Y ONBOARDING

Objetivo: Código mantenible y equipo escalable

10.1 Documentación Técnica

Instrucciones para Copilot:

Crea documentación completa en /docs:

1. **README.md** - Overview:
   - Descripción del proyecto
   - Tech stack
   - Quick start guide
   - Links a docs detallados

2. **ARCHITECTURE.md** - Arquitectura:
   - Diagrams (frontend, backend, database)
   - Data flow
   - Design patterns usados
   - Decisiones de arquitectura

3. **API.md** - API Documentation:
   - Swagger/OpenAPI spec
   - Generar con swagger-jsdoc
   - Ejemplos de requests/responses
   - Authentication flow
   - Error codes

4. **SETUP.md** - Setup Guide:
   - Pre-requisitos
   - Instalación paso a paso
   - Configuración de .env
   - Troubleshooting común

5. **CONTRIBUTING.md** - Guía para contribuir:
   - Code style guide
   - Git workflow (feature branches)
   - PR template
   - Commit message conventions

6. **DEPLOYMENT.md** - Deployment:
   - Guía de deploy a producción
   - Environment variables
   - Database migrations
   - Rollback procedure

7. **COMPONENTS.md** - Component Library:
   - Storybook deployado
   - Guía de uso de cada componente
   - Props documentation

- Examples

8. **CODE_STYLE.md** - Style Guide:
   - ESLint rules
   - Prettier config
   - Naming conventions
   - File organization

Herramientas:
- Storybook para componentes
- Swagger para API
- Mermaid para diagramas

10.2 Onboarding

Instrucciones para Copilot:

Facilita onboarding de nuevos desarrolladores:

1. **Developer Setup Script**:
   - setup.sh o setup.ps1
   - Instala todo automáticamente:
     * Node, MongoDB
     * Dependencies
     * Sample data
     * Git hooks

2. **Sample Data**:
   - npm run seed
   - Generar data realista:
     * 50 restaurantes
     * 500 productos
     * 100 usuarios
     * 1000 órdenes

3. **Docker Compose**:
   - docker-compose.yml
   - Services: app, mongodb, redis
   - One command: docker-compose up

4. **Video Tutorials**:
   - Screen recordings de:
     * Project walkthrough
     * Making first change
     * Running tests
     * Deploying

5. **Onboarding Checklist**:
   - [ ] Clonar repo
   - [ ] Setup local
   - [ ] Leer ARCHITECTURE.md
   - [ ] Run tests
   - [ ] Make first PR (fix typo)
   - [ ] Review código existente

🚀 DEPLOYMENT CHECKLIST

Pre-Launch Checklist Final

Antes de lanzar a producción, verificar:

FUNCIONALIDAD:
- [ ] Todos los tests pasan
- [ ] No hay console.logs en producción
- [ ] Error handling robusto
- [ ] Loading states en todo
- [ ] Forms validan correctamente
- [ ] Payments funcionan (Stripe test mode → live mode)
- [ ] Emails se envían
- [ ] Push notifications funcionan
- [ ] Real-time updates funcionan (Socket.io)

PERFORMANCE:
- [ ] Lighthouse score > 90 en todas las páginas
- [ ] Imágenes optimizadas
- [ ] Code splitting implementado
- [ ] CDN configurado
- [ ] Caching strategy definida
- [ ] Database índices creados
- [ ] API response times < 200ms

SEGURIDAD:
- [ ] HTTPS everywhere
- [ ] Security headers (Helmet)
- [ ] Rate limiting activo
- [ ] Input validation/sanitization
- [ ] Authentication seguro
- [ ] Secrets no expuestos
- [ ] Dependencies actualizadas (npm audit)

UX/UI:
- [ ] Responsive en todos los devices
- [ ] Accessibility compliance (WCAG AA mínimo)
- [ ] Error messages user-friendly
- [ ] Loading states elegantes
- [ ] Micro-interactions pulidas
- [ ] Cross-browser testing (Chrome, Safari, Firefox)

SEO:
- [ ] Meta tags en todas las páginas
- [ ] Sitemap.xml
- [ ] Robots.txt
- [ ] Structured data (Schema.org)
- [ ] Google Search Console configurado
- [ ] Google Analytics configurado

LEGAL:
- [ ] Privacy policy
- [ ] Terms of service
- [ ] Cookie consent
- [ ] GDPR compliance (si aplica)

MONITORING:
- [ ] Sentry configurado
- [ ] Analytics configurado
- [ ] Uptime monitoring activo
- [ ] Error alerting configurado
- [ ] Logging funcionando

BACKUP:
- [ ] Database backups automáticos
- [ ] Disaster recovery plan
- [ ] Rollback procedure documentado

DOCUMENTATION:
- [ ] README actualizado
- [ ] API docs actualizadas
- [ ] Deployment guide
- [ ] Runbook para incidentes

📝 NOTAS FINALES PARA COPILOT

Principles to Follow

1. **Mobile-First Always**: Diseñar primero para mobile, luego desktop
2. **Performance Budget**: Cada decisión debe considerar performance
3. **Accessibility First**: No es optional, es requirement
4. **Progressive Enhancement**: Funciona sin JS como fallback

5. **Semantic HTML**: Usar tags correctos, no para todo
6. **DRY Principle**: No repetir código, crear abstracciones
7. **SOLID Principles**: En arquitectura de código
8. **Error Handling**: Siempre, en todo, con buenos mensajes
9. **Testing**: Write tests como parte del feature, no después
10. **Documentation**: Code is read more than written

Code Quality Standards

- **TypeScript**: Usar tipos estrictos, no 'any'
- **ESLint**: No warnings en producción
- **Prettier**: Formateo consistente
- **Naming**: Descriptivo y consistente
- **Comments**: Solo para "por qué", no "qué"
- **Functions**: Pequeñas, una responsabilidad
- **Files**: < 300 líneas idealmente
- **Commits**: Mensajes descriptivos (Conventional Commits)

Common Pitfalls to Avoid

- ❌ No hardcodear URLs, usar env variables
- ❌ No usar var, usar const/let
- ❌ No mutar state directamente (React)
- ❌ No hacer fetch en loops
- ❌ No ignorar errores con try/catch vacíos
- ❌ No usar == , usar ===
- ❌ No inline styles en JSX (usar CSS/Tailwind)
- ❌ No componentes gigantes, split them

🎯 IMPLEMENTATION PRIORITY

Orden recomendado de implementación:

SEMANA 1-2: UI/UX Foundation
- Fase 1.1: Sistema de diseño
- Fase 1.2: Componentes base
- Comenzar Fase 1.3: Home y Restaurant pages

SEMANA 3-4: Core Features Polish
- Completar Fase 1.3: Todas las páginas principales

- Fase 1.4: Componentes de negocio
- Fase 5.1: Micro-interacciones básicas

SEMANA 5-6: Backend Optimization
- Fase 2.1: API optimization
- Fase 2.2: Real-time mejorado
- Fase 7.1: Security hardening

SEMANA 7-8: Premium Features
- Fase 3.1: Recomendaciones AI
- Fase 3.2: Programa de fidelización
- Fase 3.4: Soporte premium

SEMANA 9-10: Quality & Polish
- Fase 8.1: Testing comprehensivo
- Fase 5.2: Accessibility
- Fase 5.3: Performance optimization

SEMANA 11-12: Launch Prep
- Fase 9: Analytics y monitoring
- Fase 10: Documentation
- Fase 6: PWA features
- Pre-launch checklist

POST-LAUNCH:
- Fase 4: Mobile app (React Native)
- Fase 3.3: Social features
- Continuous improvement basado en analytics

💰 JUSTIFICACIÓN DE VALOR $100K+

Por qué este proyecto vale $100,000 USD:

1. **Tech Stack Premium**: MERN completo + TypeScript + Real-time + PWA
   Valor: $15,000

2. **UI/UX Excepcional**: Diseño minimalista premium tipo Uber Eats
   - Sistema de diseño completo
   - Micro-interacciones pulidas
   - Animaciones fluidas

- Responsive perfecto
Valor: $20,000

3. **Features Completas**:
- 3 Dashboards (Customer, Restaurant, Admin)
- Real-time tracking con Socket.io
- Payments integrados (Stripe)
- Push notifications
- Reviews y ratings
- Recomendaciones AI
- Programa de fidelización
Valor: $30,000

4. **Mobile App** (React Native):
- iOS y Android
- Feature parity con web
Valor: $15,000

5. **Infrastructure Premium**:
- Testing comprehensivo
- CI/CD pipeline
- Monitoring y analytics
- Security enterprise-grade
- Performance optimizada
- PWA con offline support
Valor: $10,000

6. **Documentation y Scalability**:
- Código clean y mantenible
- Documentation completa
- Arquitectura escalable
- Onboarding automático
Valor: $5,000

7. **Polish y Detalles**:
- Accessibility WCAG AA
- SEO optimizado
- Legal compliance
- Error handling robusto
- Loading states elegantes
Valor: $5,000

TOTAL: $100,000

🤝 SUPPORT Y RECURSOS

Recursos útiles para Copilot:

```
DESIGN INSPIRATION:
- Uber Eats: https://www.ubereats.com
- DoorDash: https://www.doordash.com
- Deliveroo: https://deliveroo.com
- Dribbble: buscar "food delivery app"

COMPONENT LIBRARIES:
- shadcn/ui: https://ui.shadcn.com
- Headless UI: https://headlessui.com
- Radix UI: https://www.radix-ui.com

ANIMATIONS:
- Framer Motion: https://www.framer.com/motion
- Lottie: https://lottiefiles.com
- GSAP: https://greensock.com/gsap

ICONS:
- Lucide: https://lucide.dev
- Heroicons: https://heroicons.com
- Phosphor: https://phosphoricons.com

TOOLS:
- Figma: diseño UI/UX
- Storybook: component library
- Chromatic: visual testing
- Playwright: E2E testing
```

¡LISTO PARA EMPEZAR!

Copilot, sigue estas fases en orden. Pregunta si necesitas clarificación en algún punto. Prioriza calidad sobre velocidad. Este es un proyecto premium que debe lucir y funcionar perfectamente.

Remember: Cada línea de código debe justificar el valor de $100,000. No shortcuts, no code smells, no tech debt.

Let's build something amazing! 🚀