Spécifications Techniques Mises à Jour - Breeze IT

Intégration Templates CRSINE & TASTYC

🎯 Templates de Référence Analysés

CRSINE - Car Washing Template

- ✅ Système de réservation avancé avec créneaux
- ✅ Interface de sélection de services visuels
- ✅ Tracking en temps réel avec étapes
- ✅ Formulaire véhicule détaillé
- ✅ Estimation de prix dynamique
- ✅ Dashboard agent complet

TASTYC - Restaurant Template

- ✅ Menu interactif avec filtres
- ✅ Système de réservation OpenTable
- ✅ Galerie avec lightbox
- ✅ Chefs & testimonials
- ✅ Blog système intégré
- ✅ E-commerce shop pages
- ✅ Animations parallax légères

📋 Impacts Majeurs sur l'Architecture

1. Structure Frontend Enrichie

```
breeze-it/apps/client-web/
├── components/
│   ├── ui/
│   │   ├── booking/        # NOUVEAU - Système réservation avancé
│   │   ├── menu/           # NOUVEAU - Menu interactif filtrable
│   │   ├── gallery/        # NOUVEAU - Lightbox gallery
│   │   ├── testimonials/   # NOUVEAU - Avis clients
│   │   └── pricing/        # NOUVEAU - Calculateur prix
│   ├── templates/
│   │   ├── crsine/         # NOUVEAU - Composants style CRSINE
│   │   └── tastyc/         # NOUVEAU - Composants style TASTYC
│   └── services/
│       ├── lavage/         # Basé sur CRSINE
│       ├── restaurant/     # Basé sur TASTYC
│       ├── fastfood/       # Style TASTYC adapté
│       ├── coiffure/       # Style CRSINE adapté
│       └── boutique/       # Hybride des deux
```

2. Nouvelles Fonctionnalités Critiques

A. Système de Réservation Avancé (Inspiré CRSINE)

```typescript
// Booking System avec Time Slots visuels
interface AdvancedBookingProps {
  service: ServiceType;
  availableSlots: TimeSlot[];
  pricing: PricingRule[];
  vehicleForm?: boolean;
  tableSelection?: boolean;
}

// Time Slot Picker Visual
interface TimeSlotProps {
  date: Date;
  slots: Array<{
    time: string;
    available: boolean;
    price?: number;
    duration: number;
  }>;
  onSlotSelect: (slot: TimeSlot) => void;
}
```

B. Menu Interactif Filtrable (Inspiré TASTYC)

```typescript
// Interactive Menu avec filtres
interface InteractiveMenuProps {
  categories: MenuCategory[];
  filters: MenuFilter[];
  sortOptions: SortOption[];
  viewMode: 'grid' | 'list' | 'cards';
}

// Menu Item avec modal détail
interface MenuItemModalProps {
  item: MenuItem;
  customizations: Customization[];
  onAddToCart: (item: MenuItem, options: any) => void;
}
```

C. Galerie avec Lightbox

```typescript
// Gallery Component
interface GalleryProps {
  images: GalleryImage[];
  categories: string[];
  lightboxEnabled: boolean;
  layout: 'masonry' | 'grid' | 'carousel';
}
```

🎨 Design System Enrichi

3. Palettes Spécialisées par Service

css

```css
/* Service Lavage - Style CRSINE */
:root {
  --lavage-primary: #825313;
  --lavage-secondary: #e18e02;
  --lavage-accent: #2563eb;      /* Bleu confiance */
  --lavage-success: #10b981;     /* Vert validation */
  --lavage-gradient: linear-gradient(135deg, #825313 0%, #e18e02 100%);
}

/* Service Restaurant - Style TASTYC Elegant */
:root {
  --restaurant-primary: #825313;
  --restaurant-secondary: #d4af37;  /* Or plus raffiné */
  --restaurant-accent: #8b4513;     /* Brun profond */
  --restaurant-warm: #f4f3f0;       /* Fond chaleureux */
  --restaurant-gradient: linear-gradient(135deg, #825313 0%, #d4af37 100%);
}

/* Service Fast-Food - Style TASTYC Fun */
:root {
  --fastfood-primary: #e18e02;      /* Orange dominant */
  --fastfood-secondary: #ff6b35;    /* Rouge-orange vibrant */
  --fastfood-accent: #ffd60a;       /* Jaune énergique */
  --fastfood-fun: #ff006e;          /* Rose fun */
  --fastfood-gradient: linear-gradient(135deg, #e18e02 0%, #ff6b35 100%);
}
```

4. Composants UI Inspirés des Templates

tsx

```
// CRSINE-inspired Car Wash Booking
interface CarWashBookingProps {
  step: 'service' | 'vehicle' | 'datetime' | 'payment';
  services: WashService[];
  onStepChange: (step: string) => void;
}

export function CarWashBooking({ step, services, onStepChange }: CarWashBookingProps) {
  return (
    <div className="booking-container">
      {/* Step Indicator */}
      <div className="steps-indicator">
        <StepIndicator steps={['Service', 'Véhicule', 'Horaire', 'Paiement']} current={step} />
      </div>

      {/* Service Selection - Visual Cards */}
      {step === 'service' && (
        <div className="service-grid">
          {services.map(service => (
            <ServiceCard
              key={service.id}
              service={service}
              pricing={service.pricing}
              duration={service.duration}
              features={service.features}
              className="hover-lift transition-all"
            />
          ))}
        </div>
      )}

      {/* Vehicle Form */}
      {step === 'vehicle' && (
        <VehicleDetailsForm
          brands={CAR_BRANDS}
          onVehicleSelect={(vehicle) => {
            // Auto-suggest compatible services
            updateServiceRecommendations(vehicle);
          }}
        />
      )}

      {/* Time Slots Visual */}
      {step === 'datetime' && (
        <TimeSlotPicker
          availableSlots={getAvailableSlots()}
          pricing={getDynamicPricing()}
          className="visual-time-picker"
        />
      )}
    </div>
  );
}

// TASTYC-inspired Restaurant Menu
interface RestaurantMenuProps {
  categories: MenuCategory[];
  chefs: Chef[];
  testimonials: Testimonial[];
}

export function RestaurantMenu({ categories, chefs, testimonials }: RestaurantMenuProps) {
  const [activeCategory, setActiveCategory] = useState('entrées');
  const [viewMode, setViewMode] = useState<'grid' | 'list'>('grid');

  return (
    <div className="restaurant-menu">
      {/* Menu Header avec parallax */}
      <MenuHeader className="parallax-bg">
        <h1 className="font-playfair text-4xl text-primary">Notre Menu</h1>
        <p className="text-lg text-gray-600">Cuisine gastronomique avec des produits frais</p>
      </MenuHeader>

      {/* Category Filters */}
      <MenuCategoryTabs
        categories={categories}
        active={activeCategory}
        onChange={setActiveCategory}
        className="sticky-tabs"
      />

      {/* Menu Items Grid/List */}
      <MenuItemsGrid
        items={getItemsByCategory(activeCategory)}
        viewMode={viewMode}
        onItemClick={(item) => openMenuItemModal(item)}
        className="menu-items-container"
      />

      {/* Chef Section */}
      <ChefSection chefs={chefs} className="mt-16" />

      {/* Testimonials */}
      <TestimonialsCarousel testimonials={testimonials} />
    </div>
  );
}
```

🔧 Nouvelles Fonctionnalités Backend

5. API Enrichies pour Templates

```typescript
```

```
// Advanced Booking API (inspiré CRSINE)
interface BookingController {
  // Disponibilités avec pricing dynamique
  @Get('/availability')
  async getAvailability(
    @Query('service') serviceId: string,
    @Query('date') date: string,
    @Query('vehicle') vehicleType?: string
  ): Promise<AvailabilityResponse> {
    const slots = await this.bookingService.getAvailableSlots(serviceId, date);
    const pricing = await this.pricingService.calculateDynamicPricing(slots, vehicleType);

    return {
      date,
      slots: slots.map(slot => ({
        ...slot,
        price: pricing[slot.id],
        estimatedDuration: this.getDurationByService(serviceId, vehicleType)
      }))
    };
  }

  // Estimation prix en temps réel
  @Post('/estimate')
  async estimatePrice(
    @Body() estimateRequest: EstimateRequest
  ): Promise<PriceEstimate> {
    const basePrice = await this.pricingService.getBasePrice(estimateRequest.serviceId);
    const vehicleMultiplier = this.getVehicleMultiplier(estimateRequest.vehicleType);
    const timeMultiplier = this.getTimeMultiplier(estimateRequest.timeSlot);

    return {
      basePrice,
      totalPrice: basePrice * vehicleMultiplier * timeMultiplier,
      breakdown: {
        service: basePrice,
        vehicleType: vehicleMultiplier,
        timeSlot: timeMultiplier,
        addons: estimateRequest.addons?.reduce((sum, addon) => sum + addon.price, 0) || 0
      }
    };
  }
}

// Menu API (inspiré TASTYC)
interface MenuController {
  // Menu avec filtres avancés
  @Get('/menu')
  async getMenu(
    @Query('category') category?: string,
    @Query('dietary') dietary?: string[],
    @Query('price_range') priceRange?: string,
    @Query('sort') sort?: 'price' | 'popularity' | 'name'
  ): Promise<MenuResponse> {
    const filters = {
      category,
    dietary,
    priceRange: priceRange ? JSON.parse(priceRange) : undefined,
    sort
  };

  const menu = await this.menuService.getFilteredMenu(filters);
  const categories = await this.menuService.getCategories();

  return {
    items: menu,
    categories,
    totalCount: menu.length,
    filters: await this.menuService.getAvailableFilters()
  };
}

// Recommandations intelligentes
@Get('/recommendations')
async getRecommendations(
  @Query('userId') userId?: string,
  @Query('itemId') itemId?: string
): Promise<MenuItem[]> {
  if (userId) {
    return this.recommendationService.getPersonalizedRecommendations(userId);
  }
  if (itemId) {
    return this.recommendationService.getSimilarItems(itemId);
  }
  return this.recommendationService.getPopularItems();
}
}
```

6. Modèles de Données Enrichis

```prisma
```

```
// Enhanced Booking Model (CRSINE-inspired)
model Reservation {
  id String @id @default(cuid())

  // Basic fields...
  userId String
  serviceId String
  scheduledAt DateTime
  status ReservationStatus

  // NOUVEAU - Vehicle details pour lavage
  vehicleDetails Json? @map("vehicle_details") // { brand, model, type, size, color, licensePlate }

  // NOUVEAU - Table preferences pour restaurant
  tablePreferences Json? @map("table_preferences") // { tableNumber, location, capacity }

  // NOUVEAU - Service customizations
  selectedAddons Json[] @default([]) @map("selected_addons")
  specialRequests String? @map("special_requests")

  // NOUVEAU - Pricing breakdown
  pricingBreakdown Json @map("pricing_breakdown") // { base, addons, multipliers, total }

  // NOUVEAU - Tracking
  estimatedDuration Int? @map("estimated_duration")
  actualStartTime DateTime? @map("actual_start_time")
  actualEndTime DateTime? @map("actual_end_time")

  // Relations
  user User @relation(fields: [userId], references: [id])
  service Service @relation(fields: [serviceId], references: [id])
  reviews Review[]

  @@map("reservations")
}

// Enhanced Menu Model (TASTYC-inspired)
model MenuItem {
  id String @id @default(cuid())
  serviceId String @map("service_id")

  // Basic fields
  name String
  description String
  price Decimal @db.Decimal(10,2)
  category String

  // NOUVEAU - Rich content
  images Json @default("[]") // Multiple images avec descriptions
  ingredients Json @default("[]") // Liste ingrédients détaillée
  nutritionalInfo Json? @map("nutritional_info") // Calories, allergens, etc.

  // NOUVEAU - Dietary info
  dietaryTags String[] @default([]) @map("dietary_tags") // ["vegetarian", "gluten-free", "vegan"]
  spiceLevel Int? @map("spice_level") // 1-5
  // NOUVEAU - Customizations
  customizations Json @default("[]") @map("customizations") // Size, cooking level, sides

  // NOUVEAU - Popularity tracking
  viewCount Int @default(0) @map("view_count")
  orderCount Int @default(0) @map("order_count")
  averageRating Decimal? @map("average_rating") @db.Decimal(3,2)

  // NOUVEAU - Chef info
  chefId String? @map("chef_id")
  chef Chef? @relation(fields: [chefId], references: [id])

  // Relations
  service Service @relation(fields: [serviceId], references: [id])
  orderItems OrderItem[]
  reviews Review[]

  @@map("menu_items")
}

// NOUVEAU - Chef Model (TASTYC-inspired)
model Chef {
  id String @id @default(cuid())
  name String
  title String // "Chef Exécutif", "Sous Chef"
  bio String?
  avatar String?
  specialties String[] @default([])
  experience Int? // Years of experience
  awards Json @default("[]")

  // Social links
  socialLinks Json @default("{}") @map("social_links")

  // Relations
  menuItems MenuItem[]

  @@map("chefs")
}

// NOUVEAU - Review Model
model Review {
  id String @id @default(cuid())
  userId String @map("user_id")

  // Polymorphic relations
  menuItemId String? @map("menu_item_id")
  reservationId String? @map("reservation_id")

  rating Int // 1-5
  title String?
  comment String
  images Json @default("[]")

  // Moderation
  isApproved Boolean @default(false) @map("is_approved")
  moderatedAt DateTime? @map("moderated_at")
  createdAt DateTime @default(now()) @map("created_at")

  // Relations
  user User @relation(fields: [userId], references: [id])
  menuItem MenuItem? @relation(fields: [menuItemId], references: [id])
  reservation Reservation? @relation(fields: [reservationId], references: [id])

  @@map("reviews")
}
```

🎬 Animations et Interactions

7. Animations Inspirées des Templates

```
tsx
```

```
// CRSINE-style animations pour booking flow
import { motion, AnimatePresence } from 'framer-motion';

export function BookingStepTransition({ children, step }: { children: React.ReactNode, step: string }) {
  return (
    <AnimatePresence mode="wait">
      <motion.div
        key={step}
        initial={{ opacity: 0, x: 50 }}
        animate={{ opacity: 1, x: 0 }}
        exit={{ opacity: 0, x: -50 }}
        transition={{ duration: 0.3, ease: "easeInOut" }}
        className="booking-step"
      >
        {children}
      </motion.div>
    </AnimatePresence>
  );
}

// TASTYC-style parallax et hover effects
export function MenuItemCard({ item }: { item: MenuItem }) {
  return (
    <motion.div
      className="menu-item-card"
      whileHover={{
        scale: 1.02,
        boxShadow: "0 20px 40px rgba(130, 83, 19, 0.15)"
      }}
      whileTap={{ scale: 0.98 }}
      transition={{ duration: 0.2 }}
    >
      <div className="relative overflow-hidden">
        <motion.img
          src={item.image}
          alt={item.name}
          className="w-full h-48 object-cover"
          whileHover={{ scale: 1.1 }}
          transition={{ duration: 0.3 }}
        />

        {/* Overlay gradient */}
        <div className="absolute inset-0 bg-gradient-to-t from-black/50 to-transparent opacity-0 hover:opacity-
      </div>

      <div className="p-6">
        <h3 className="font-semibold text-lg text-primary">{item.name}</h3>
        <p className="text-gray-600 mt-2">{item.description}</p>

        <div className="flex justify-between items-center mt-4">
          <span className="text-2xl font-bold text-secondary">{item.price} FCFA</span>
          <motion.button
            className="bg-primary text-white px-4 py-2 rounded-lg"
            whileHover={{ backgroundColor: "var(--secondary)" }}
            whileTap={{ scale: 0.95 }}
          >
        Commander
      </motion.button>
    </div>
  </div>
</motion.div>
);
}
```

📱 Mobile Experience Optimisée

8. Components Mobile-First (Templates-inspired)

```
tsx
```

```
// Mobile Car Wash Booking (CRSINE-inspired)
export function MobileCarWashBooking() {
  const [currentStep, setCurrentStep] = useState(0);

  return (
    <div className="mobile-booking">
      {/* Bottom Sheet Navigation */}
      <BottomSheet
        isOpen={true}
        snapPoints={[0.3, 0.6, 0.9]}
        header={
          <StepProgress
            steps={['Service', 'Véhicule', 'Horaire', 'Confirmation']}
            current={currentStep}
          />
        }
      >
        <SwipeableSteps
          currentStep={currentStep}
          onStepChange={setCurrentStep}
        >
          <ServiceSelection />
          <VehicleForm />
          <TimeSlotSelection />
          <BookingConfirmation />
        </SwipeableSteps>
      </BottomSheet>
    </div>
  );
}

// Mobile Restaurant Menu (TASTYC-inspired)
export function MobileRestaurantMenu() {
  return (
    <div className="mobile-menu">
      {/* Sticky Category Header */}
      <StickyHeader>
        <HorizontalCategoryScroll categories={categories} />
      </StickyHeader>

      {/* Virtual Scroll Menu Items */}
      <VirtualizedMenuList
        items={menuItems}
        renderItem={(item) => (
          <MenuItemCard
            item={item}
            layout="horizontal"
            onClick={() => openMenuModal(item)}
          />
        )}
      />

      {/* Floating Cart Button */}
      <FloatingCartButton
        itemCount={cartItems.length}
        total={cartTotal}
        onClick={() => openCart()}
      />
    </div>
  );
}
```

🔄 Workflows Mis à Jour

9. Car Wash Workflow (CRSINE-style)

```
mermaid

graph TD
    A[Landing Page] --> B[Service Selection Visual]
    B --> C[Vehicle Details Form]
    C --> D[Add-ons Selection]
    D --> E[Time Slot Picker Visual]
    E --> F[Price Calculator Display]
    F --> G[Customer Info]
    G --> H[Payment Selection]
    H --> I[Booking Confirmation]
    I --> J[QR Code Generation]
    J --> K[Agent Dashboard Update]
    K --> L[WhatsApp Notification]

    L --> M[Customer Arrival]
    M --> N[QR Scan by Agent]
    N --> O[Pre-wash Stage]
    O --> P[Main Wash Stage]
    P --> Q[Finishing Stage]
    Q --> R[Completion Notification]
    R --> S[Review Request]
```

10. Restaurant Workflow (TASTYC-style)

```
mermaid

graph TD
    A[Restaurant Landing] --> B[Menu Browsing]
    B --> C[Table Reservation]
    C --> D[Date/Time Selection]
    D --> E[Party Size & Preferences]
    E --> F[Reservation Confirmation]
    F --> G[Email/SMS Confirmation]

    G --> H[Customer Arrival]
    H --> I[Table Assignment]
    I --> J[Digital Menu via QR]
    J --> K[Order Taking by Waiter]
    K --> L[Kitchen Display]
    L --> M[Food Preparation]
    M --> N[Service to Table]
    N --> O[Payment Processing]
    O --> P[Review & Feedback]
```

📊 Nouvelles Métriques et Analytics

11. KPIs Inspirés des Templates

```typescript
// Car Wash Analytics (CRSINE-inspired)
interface CarWashMetrics {
  bookingConversionRate: number; // % visitors who complete booking
  averageServiceTime: number; // Minutes per wash
  vehicleTypeDistribution: Record<string, number>;
  peakHours: Array<{ hour: number, bookings: number }>;
  addonsPopularity: Array<{ addon: string, percentage: number }>;
  customerSatisfactionScore: number; // 1-5
  agentEfficiency: Record<string, number>; // Cars per hour per agent
}

// Restaurant Analytics (TASTYC-inspired)
interface RestaurantMetrics {
  tableUtilizationRate: number; // % tables occupied during peak
  averageDiningTime: number; // Minutes per table
  menuItemPopularity: Array<{ item: string, orders: number }>;
  reservationShowUpRate: number; // % confirmed reservations honored
  averageOrderValue: number; // FCFA per order
  chefRatings: Record<string, number>; // Average rating per chef
  peakDiningTimes: Array<{ hour: number, reservations: number }>;
}
```

🚀 Plan d'Implémentation Mis à Jour

Phase 1: Infrastructure + Template Setup (3 semaines)

```bash
# Semaine 1: Setup base + template analysis
claude-code create "Setup Breeze IT monorepo with CRSINE and TASTYC template components"

# Semaine 2: Design system basé sur templates
claude-code create "Create design system based on CRSINE and TASTYC with Breeze IT colors"

# Semaine 3: Base components template-inspired
claude-code create "Build reusable UI components inspired by CRSINE booking flow and TASTYC menu system"
```

Phase 2: Service Lavage (CRSINE-style) (2 semaines)

```bash
# Semaine 4: Booking system avancé
claude-code create "Implement CRSINE-style car wash booking with visual time slots and dynamic pricing"

# Semaine 5: Agent dashboard + tracking
claude-code create "Create car wash agent dashboard with QR tracking and multi-step workflow"
```

Phase 3: Restaurant (TASTYC-style) (2 semaines)

```bash
```

```bash
# Semaine 6: Menu interactif + réservations
claude-code create "Build TASTYC-style restaurant menu with filters, lightbox gallery, and OpenTable integration

# Semaine 7: Chef profiles + reviews
claude-code create "Implement chef profiles, testimonials system, and review management"
```

Phase 4: Fast-Food (TASTYC adapté) (2 semaines)

```bash
# Semaine 8: Interface fun + gamification
claude-code create "Create youth-oriented fast-food interface with TASTYC design adapted for quick ordering"

# Semaine 9: Real-time order tracking
claude-code create "Implement real-time order tracking with gamified user experience"
```

Phase 5: Services restants + intégrations (3 semaines)

```bash
# Semaine 10: Coiffure (CRSINE adapté)
claude-code create "Build hair salon booking system using CRSINE booking flow for appointments"

# Semaine 11: Boutique (hybride)
claude-code create "Create auto parts shop using hybrid CRSINE/TASTYC design for product catalog"

# Semaine 12: Intégrations finales
claude-code create "Integrate WhatsApp, payments, Odoo, and implement template-inspired animations"
```

🎨 Assets et Ressources Nécessaires

12. Images et Médias (Template-style)

```typescript
```

```
// Asset Requirements based on templates
interface AssetRequirements {
  carWash: {
    heroImages: string[]; // 3-4 hero banners CRSINE-style
    serviceIcons: string[]; // Wash types icons
    beforeAfter: string[]; // Before/after car images
    processSteps: string[]; // 3-step process illustrations
    teamPhotos: string[]; // Agent photos
  };

  restaurant: {
    heroImages: string[]; // Elegant restaurant ambiance
    dishPhotos: string[]; // High-quality food photography
    chefPortraits: string[]; // Professional chef photos
    interiorShots: string[]; // Restaurant interior views
    ingredientPhotos: string[]; // Fresh ingredients
  };

  fastFood: {
    productPhotos: string[]; // Vibrant food photography
    lifestyleImages: string[]; // Young people enjoying food
    brandElements: string[]; // Fun graphics and icons
    processIcons: string[]; // Order process steps
  };
}
```

💡 Conclusion et Recommandations

Avantages de cette Approche Template-Inspired:

1. 🎯 **UX Prouvée** - Templates testés avec vraie audience
2. ⚡ **Développement Accéléré** - Composants pré-conçus à adapter
3. 📱 **Mobile-First** - Templates déjà responsive
4. 🎨 **Design Cohérent** - Esthétique professionnelle garantie
5. 🔧 **Fonctionnalités Avancées** - Booking, menu, gallery prêts

Adaptations Nécessaires:

1. **Couleurs** - Adapter 🟫 #825313 et 🟧 #e18e02 aux templates
2. **Contenu** - Localiser pour marché sénégalais
3. **Paiements** - Intégrer Wave/Orange Money
4. **Langue** - Interface français + Wolof optionnel
5. **Features** - Ajouter fonctionnalités spécifiques Breeze IT

Cette approche nous donne une base solide et éprouvée pour créer une expérience utilisateur exceptionnelle en s'inspirant des meilleurs templates du marché tout en conservant l'identité unique de Breeze IT.