

# Système de Commande Restaurant Adaptatif - Le Gourmet -

Alexis Dubarry - Florent Bélot -  
Shanti Noel - Adam Mir-sadjadi



# Sommaire

- Présentation du projet :
  - Système interactif,
  - Focus utilisateurs (Personas),
  - Stack logicielle
- Présentation des adaptations :
  - Adaptation au dispositif,
  - Adaptations système,
  - Adaptation initiée par l'utilisateur : Mode enfant
- API & codebase
  - Backend Idéal
  - Réutilisations de composants IHM
  - Réutilisations de code

# Caractéristiques du système interactif

## Objectifs & supports

- Système de commande pour restaurant avec adaptation contextuelle :
  - Dispositif : Tablette (serveur) ou Smartphone (client)
  - Profil : Adulte ou Enfant
  - Contexte : Service normal ou Heure de pointe

## Problématique

- Comment créer une seule interface qui s'adapte intelligemment à différents contextes tout en maximisant la réutilisation des composants ?

# Personas Marie – Serveuse

## Profil

- Âge : 30 ans
- Métier : Serveuse polyvalente
- Expérience : 8 ans en restauration
- Dispositif : Tablette

## Objectifs

- Prendre les commandes rapidement sans erreur
- Gérer plusieurs tables simultanément
- Limiter les allers-retours avec les clients

## Contexte d'utilisation

- En salle, debout, parfois en mouvement
- Bruit ambiant élevé
- Pics d'activité le midi et le soir



# Personas Jean, Sophie & Léa – Petite famille

## Profil

- Âge : 35–40 & 10 ans
- Situation : Couple avec enfants
- Dispositif : Smartphone / Tablette (serveur)

## Objectifs

- Trouver des plats équilibrés pour la famille
- Adapter les choix aux goûts des enfants
- Contrôler le budget et les allergènes

## Contexte d'utilisation

- Repas en famille le week-end
- Temps modéré, ambiance détendue



# Personas Marc – Client Pressé

## Profil

- Âge : 35 ans
- Situation : Actif, pause déjeuner courte
- Dispositif : Smartphone

## Objectifs

- Commander et manger en moins de 30 minutes
- Éviter l'attente inutile
- Aller à l'essentiel

## Contexte d'utilisation

- Heures de pointe (12h-14h)
- Seul ou avec collègues



# Stack logicielle

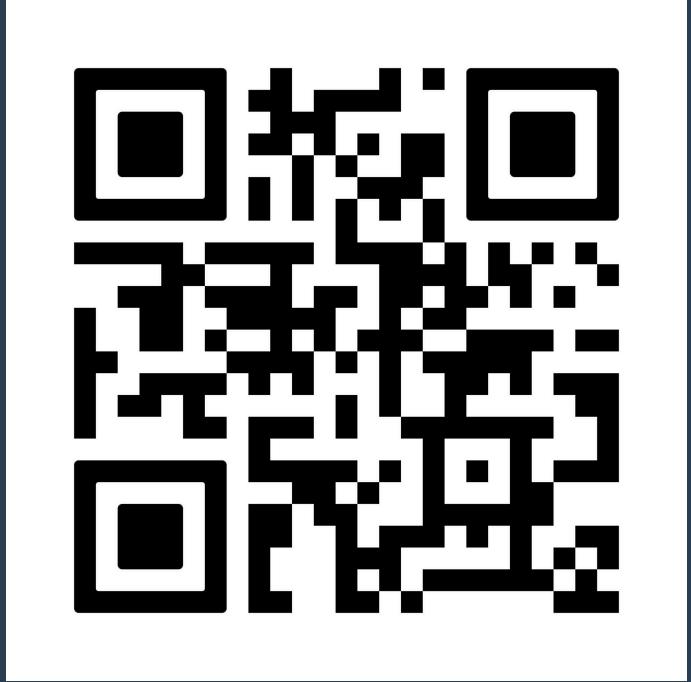
- Application Web : **React + Typescript + Tailwind CSS**
  - Donne un cadre de développement clair et homogène
  - Favorise la réutilisation de composants/code
  - Durée de vie des composants gérée par le framework, assure la scalabilité
  - Facilite la persistance des données entre les composants
  - Offre un vaste éventail de libraires pour étendre les fonctionnalités de l'application
- Framework de déploiement : **Vite**
  - Configuration simple et rapide appropriée au développement en React
- Hébergée sur : **Vercel** (+ possibilité d'installer une **WebApp**)
- Utilisation de **Figma** et des outils IA intégrés pour le prototypage
  - Génération de code pour la stack choisie
  - Prompts basés sur les besoins identifiés puis améliorations itératives
- Backend simulé

# Adaptation au dispositif

- QR Code disponible sur chaque table
- Commande possible sur smartphone
- Alternative proposée par le serveur :
  - Tablette pour commande groupée
- Adaptation selon préférences et contraintes utilisateur

# Dispositif : Smartphone

- Format portrait
- Utilisation personnelle à une main
- Interface :
  - Scrollable
  - Simplifiée
  - Optimisée mobile
- Accès via QR Code

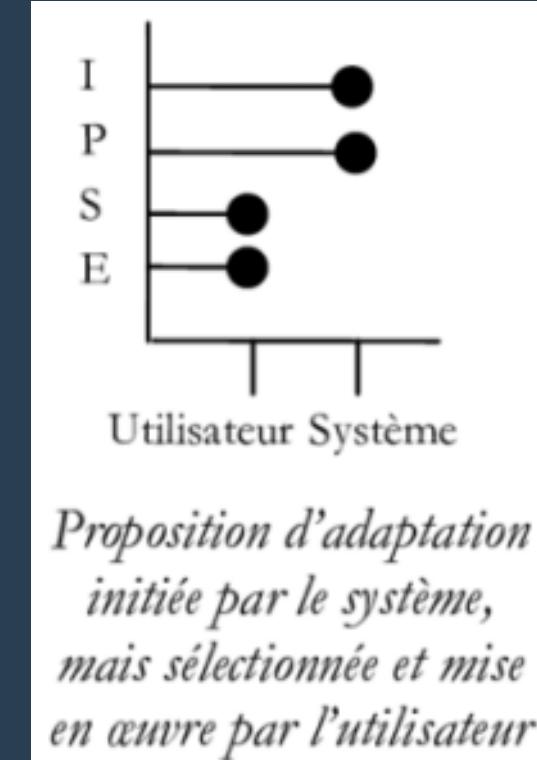


Version  
smartphone

# Dispositif : Tablette

- Format paysage : vue sur carte et panier
- interface de selection de table pour le serveur
- WebApp dédiée
- Interface :
  - Boutons plus grands
  - Lecture rapide
- Idéal pour commande collective

# Adaptation système : Heure de pointe

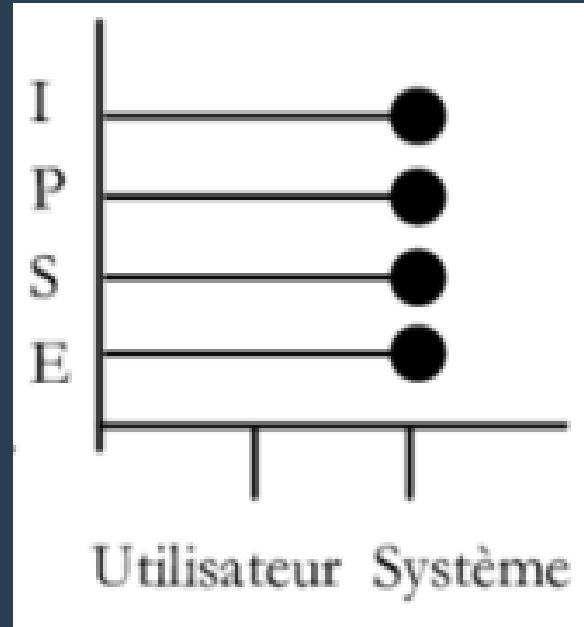


- Adaptation initiée par le système, mais sélectionnée et mise en oeuvre par l'utilisateur
- Détection automatique du nombre de commandes, cumule le temps  
=> à un certain seuil, le mode “rush” s'active
- Apparition d'une alerte :
  - “Temps d'attente élevé”
- Proposition d'une interface dédié, initiée par le système

# Interface Rush : manger rapidement

- Bouton : “Je souhaite manger rapidement”
- Redirection vers une interface dédiée :
  - Catégories spéciales
  - Mise en avant des plats rapides
  - Temps de préparation affiché
- Réduction du nombre d'options visibles

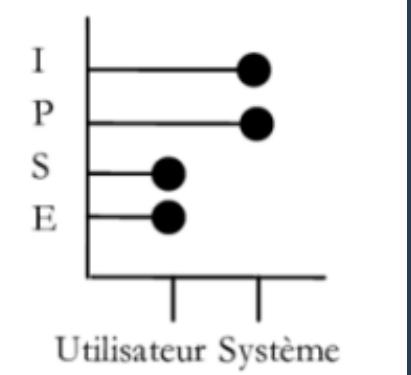
# Adaptation système : Charge cognitive



- Adaptation : aucun contrôle par l'utilisateur
- Si l'utilisateur hésite > 6 secondes :
  - Affichage de suggestions automatiques
- Objectif : réduire la charge cognitive
- Amélioration : système qui propose des plats réellement similaires

# Mode Enfant

- Adaptation initiée par le système, mais sélectionnée et mise en oeuvre par l'utilisateur
- Pensé pour :
  - Réduire la complexité
  - Faciliter la compréhension sous forme d'un jeu
  - Offrir des récompenses à la clé du processus pour motiver les choix
- Expérience différente du mode adulte : la prise de commande devient un petit jeu
- Prix et portions adaptés



*Proposition d'adaptation  
initiée par le système,  
mais sélectionnée et mise  
en œuvre par l'utilisateur*

# Présentation de l'API

## RÉCUPÉRATION

L'API permet la récupération des menus et des informations relatives au restaurant et au mode enfant à l'ouverture de l'application

## ENVOI DE COMMANDES

Une fois les commandes faites l'application envoie un message au backend pour finaliser la commande, puis un retour du backend permet la validation de la commande.

## GESTION

La gestion du mode rush se fait en temps réel (vérification toutes les 10 secondes). Les suggestions se font sur une requête ponctuelle du front.

# Backend Idéal

STRUCTURE:

```
{  
  "id": string,          // Identifiant unique (ex: "e1", "p1", "d1")  
  "name": string,        // Nom du plat  
  "description": string, // Description courte du plat  
  "category": string,   // "entrée" | "plat" | "dessert"  
  "subcategory": string, // Sous-catégorie (ex: "Salades", "Viandes")  
  "price": number,       // Prix en euros (ex: 9.50)  
  "prepTime": number,    // Temps de préparation en minutes  
  "popularity": number, // Score de popularité (1-5)  
  "isSpecialOfDay": boolean, // Est-ce le plat du jour ?  
  "isQuick": boolean,    // Plat rapide (< 15 min) ?  
  "imageUrl": string,    // URL de l'image du plat  
  "kidFriendly": boolean, // Adapté aux enfants ?  
  "hasVegetables": boolean, // Contient des légumes ?  
  "ingredients": string[], // Liste d'ingrédients pour recherche  
  "isVegetarian": boolean, // Option végétarienne ?  
  "isVegan": boolean,     // Option végane ?  
  "isGlutenFree": boolean, // Sans gluten ?  
  "spicyLevel": number,   // Niveau de piment (0-3)  
  "isLight": boolean,     // Option légère/santé ?  
  "isLocal": boolean,    // Produits locaux ?  
  "cuisine": string       // "française" | "italienne" | "asiatique" | ...  
}
```

- Données des plats :

# Backend Idéal

- Données des cadeaux :
- Données des textes du mode enfant :
  - Utile côté backend pour la réutilisation du code et la personnalisation de l'interface
  - Si le backend ne le fournit pas, utiliser valeurs par défaut

Structure des récompenses:

```
{  
  "id": string,      // Identifiant unique (ex: "lollipop", "icecream")  
  "name": string,    // Nom de la récompense  
  "emoji": string,   // Emoji représentant la récompense  
  "stars": number,   // Coût en étoiles (3 ou 6)  
  "description": string, // Description de la récompense  
  "imageUrl": string  // URL de l'image (optionnel, peut utiliser emoji)  
}
```

Configuration du mode enfant:

```
{  
{  
  "welcome": string,    // Message d'accueil initial  
  "entrée": string,     // Message lors de la sélection d'entrée  
  "plat": string,       // Message lors de la sélection du plat  
  "dessert": string,    // Message lors de la sélection du dessert  
  "complete": string,   // Message de félicitation finale  
  "cart": string,       // Message dans l'écran du panier  
  "rewards": string     // Message dans l'écran des récompenses  
},  
"encouragements": ["Message 1", "Message 2", "Message 3", ...]  
}
```

# Backend Idéal

- Configuration du restaurant :

STRUCTURE:

```
{  
    "name": string,          // Nom du restaurant  
    "logo": string,          // URL du logo  
    "welcomeMessage": string, // Message d'accueil  
    "rushHourConfig": {  
        "bannerMessage": string, // Message de la bannière Rush  
        "warningThreshold": number // Seuil d'alerte temps d'attente  
    },  
    "features": {  
        "childMode": boolean,      // Mode enfant disponible ?  
        "adaptiveSuggestions": boolean, // Suggestions adaptatives ?  
        "ingredientSearch": boolean, // Recherche d'ingrédients ?  
        "multipleDevices": boolean // Support tablette + smartphone ?  
    }  
}
```

# Backend Idéal

Le mode suggestion.

Fonctionnement:

1. L'application détecte qu'un utilisateur hésite
2. L'application interroge POST /api/suggestion

Body: { "dishId": "..."}

3. Le backend renvoie une liste de suggestion de commandes :

Réponse: { "suggestion": ["dishId1", "dishId2", "dishId3", ...]}

# Backend Idéal

Le mode Rush se base sur la somme du temps de préparation des commandes en cours de préparation.

## FONCTIONNEMENT:

1. L'application interroge GET /api/rush-status toutes les 10 secondes
2. Le backend renvoie le nombre de commandes en préparation
3. Si ce nombre > seuil (configurable), le mode Rush s'active

ENDPOINT: GET /api/rush-status

## REQUEST:

- Aucun paramètre requis
- Headers: Content-Type: application/json

## RESPONSE FORMAT:

```
{  
  "ordersTimeInProgress": number,  
  // Somme actuel des commandes en préparation  
  "timestamp": string  
  // Timestamp ISO 8601 de la réponse  
}
```

## EXEMPLE DE REponse:

```
{  
  "ordersTimeInProgress": 15,  
  "timestamp": "2025-12-06T14:23:45.123Z"  
}
```

# Backend Idéal

- Format de l'envoie de la commande :

```
{  
    "orderId": string,          // ID unique de commande  
    "timestamp": string,        // ISO 8601 datetime  
    "tableNumber": number,      // Numéro de table  
    "deviceType": string,       // "tablet" | "smartphone"  
    "userMode": string,         // "normal" | "child"  
    "items": [  
        {  
            "dishId": string,      // ID du plat  
            "quantity": number,    // Quantité commandée  
            "price": number,       // Prix unitaire  
        }  
    ],  
    "totalPrice": number,        // Prix total  
    "totalItems": number,        // Nombre total d'articles  
    "estimatedPrepTime": number, // Temps de préparation estimé  
    "childModeData": {          // Seulement si userMode = "child"  
        "starsEarned": number,  // Étoiles gagnées  
        "rewardsSelected": string[] // IDs des récompenses choisies  
    }  
}
```

- Réponse:

```
{  
    "success": boolean,  
    "orderId": string,  
    "estimatedTime": number  
}
```

# Exemple :

Récupérer tous les plats :

GET /api/dishes

Response: { "dishes": [...] }

Récupérer la configuration :

GET /api/restaurant-config

Response: { "restaurantConfig": {...} }

Récupérer les récompenses enfant :

GET /api/child-rewards

Response: { "childRewards": [...] }

Vérifier le statut du mode Rush :

GET /api/rush-status

Response: { "ordersTimeInProgress": [...] }

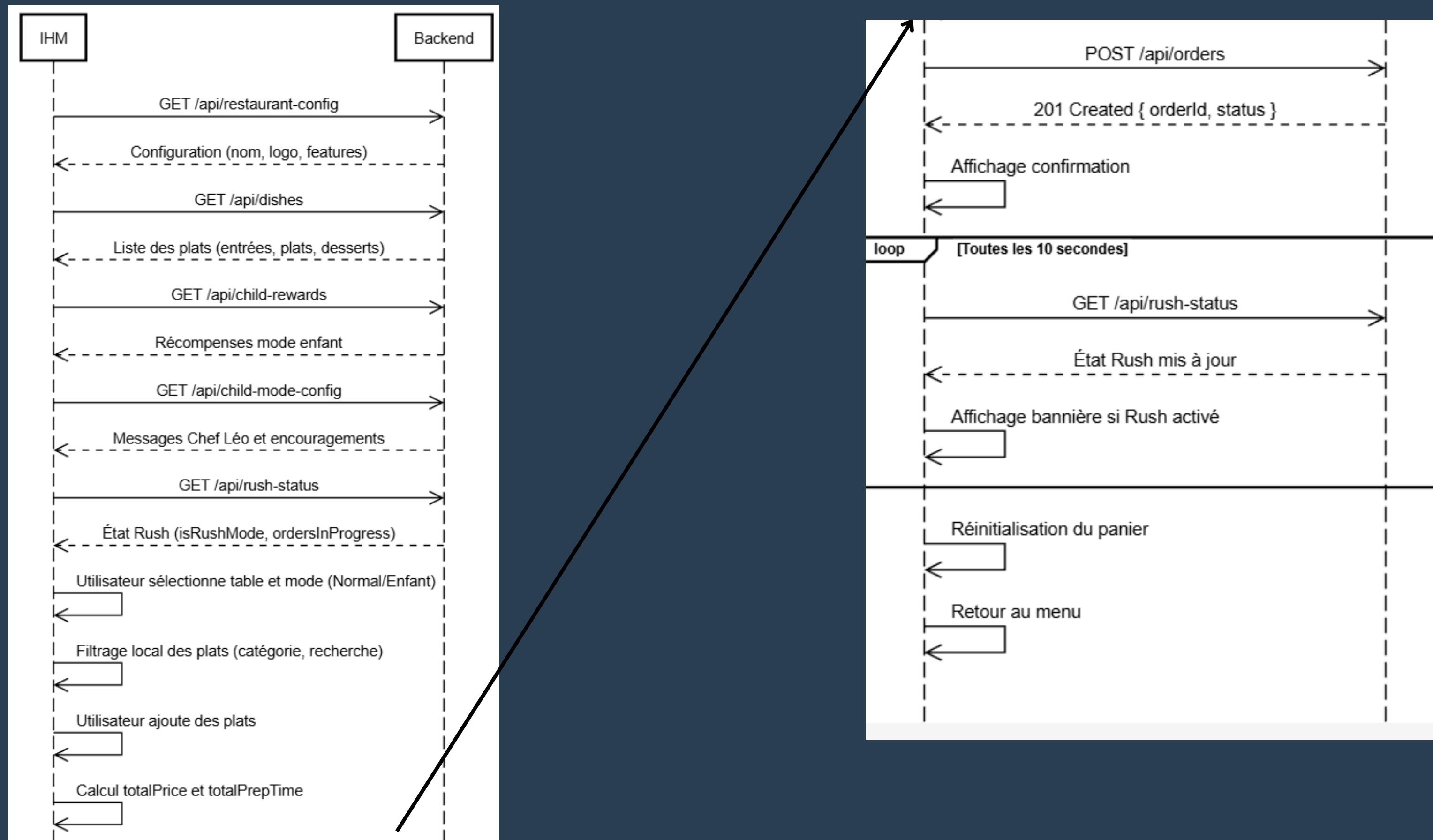
Soumettre une commande :

POST /api/orders

Body: { "orderId": "...", "items": [...], ... }

Response: { "success": true, "orderId": "...",  
"estimatedTime": 20 }

# Diagramme des séquences



# Réutilisation de composants IHM / code

ARCHITECTURE EN 3 COUCHES

COMPOSANTS MÉTIER  
MenuInterface, DishCard, ChildMode...



Réutilise

COMPOSANTS UI  
Button, Card, Badge, Dialog...



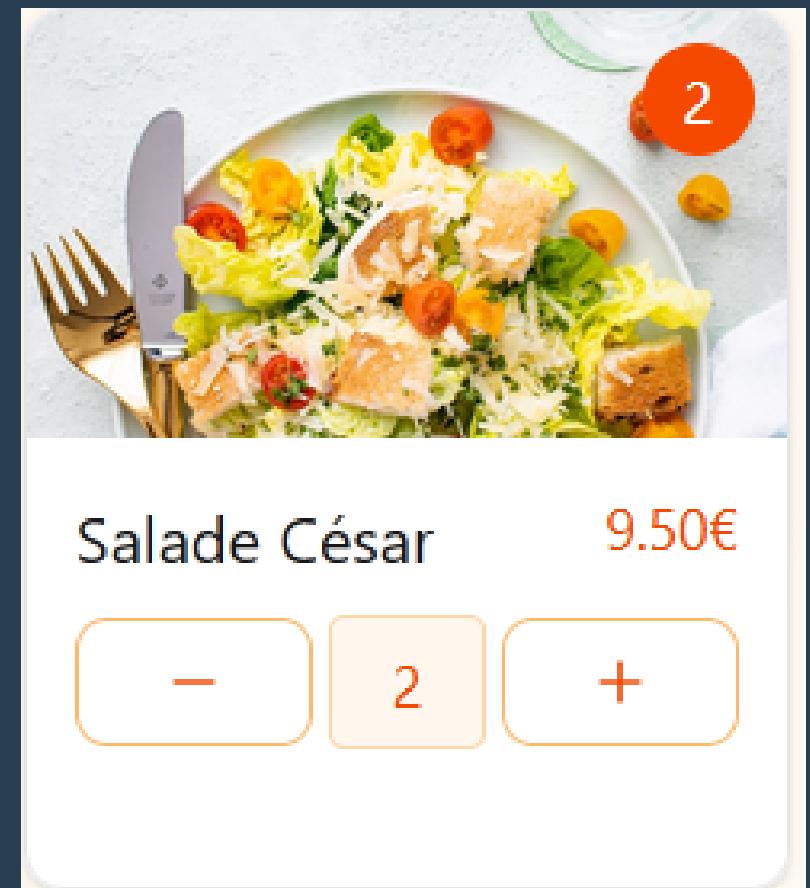
Stylisé par

TAILWIND CSS + Variants CVA  
Classes utilitaires + système de design

# Exemple Métier : DishCard

## DishCard

- └─ Card (structure & bordure)
- └─ ImageWithFallback (image plat)
- └─ Badge (indicateur "Plat du jour")
- └─ Button (ajouter au panier)
  - └─ Icônes (Plus, Minus, Star)
- └─ Classes Tailwind (responsive, hover, spacing)



```
export function DishCard({ dish, onAddToCart, deviceType, quantity }: DishCardProps)
```

# Exemple Métier : DishCard

```
// components/DishCard.tsx - Composition de multiples composants
export function DishCard({ dish, onAddToCart, deviceType, quantity }: DishCardProps) {
  const isTablet = deviceType === 'tablet';
  const hasQuantity = quantity > 0;

  return (
    <div className="bg-white rounded-lg shadow-sm border hover:shadow-md
      transition-all cursor-pointer group relative">
      {/* Badge de quantité - Réutilisation conditionnelle */}
      {isTablet && hasQuantity && (
        <div className="absolute top-2 right-2 bg-orange-600 text-white
          rounded-full w-7 h-7 flex items-center justify-center">
          {quantity}
        </div>
      )}

      {/* Image avec fallback */}
      <div className="relative aspect-video overflow-hidden">
        <ImageWithFallback
          src={dish.imageUrl}
          alt={dish.name}
          className="w-full h-full object-cover group-hover:scale-105 transition-transform"
        />

      {/* Badge "Plat du jour" - Réutilisation avec composition */}
      {dish.isSpecialOfDay && !hasQuantity && (
        <div className="absolute top-2 right-2 bg-orange-600 text-white
          rounded-full flex items-center gap-1 px-2 py-1 text-xs">
          <Star className="w-3 h-3 fill-current" />
          {deviceType === 'tablet' && 'Plat du jour'}
        </div>
      )}
    </div>
  );
}
```

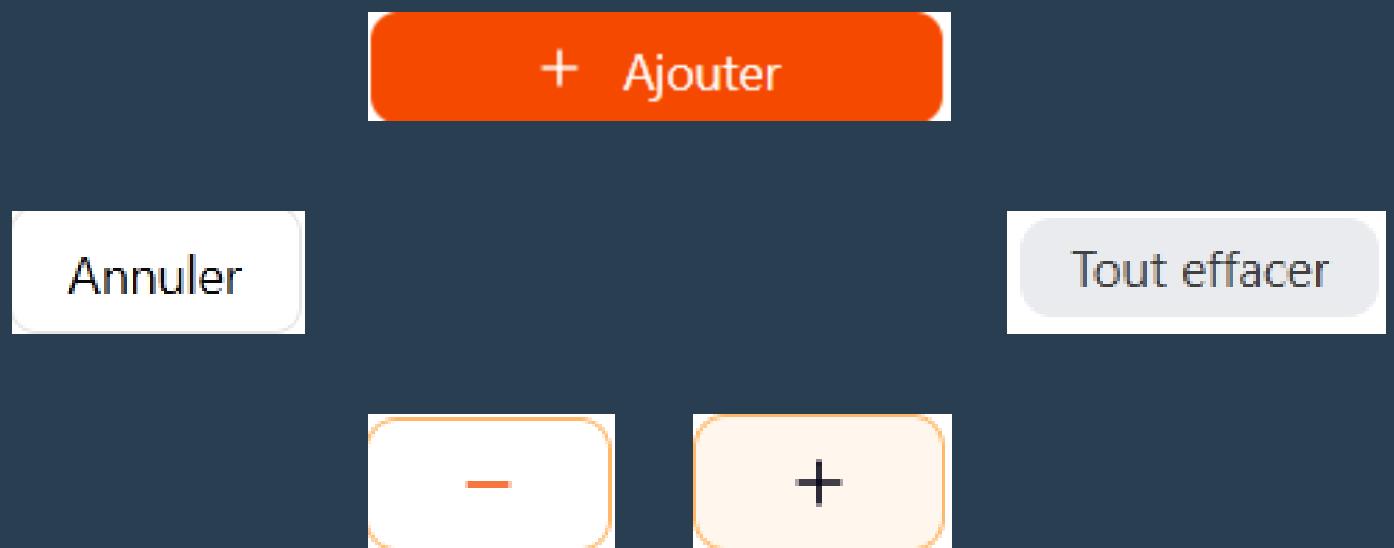
```
/* Contenu avec classes responsive Tailwind */
<div className={deviceType === 'smartphone' ? 'p-2' : 'p-3'}>
  <div className="flex items-start justify-between gap-2 mb-2">
    <h4 className={`${text-neutral-900 line-clamp-2 flex-1
      ${deviceType === 'smartphone' ? 'text-xs' : ''}`}>
      {dish.name}
    </h4>
    <span className="text-orange-600 whitespace nowrap">
      {dish.price.toFixed(2)}
    </span>
  </div>

  {/* Boutons avec variants différents selon le mode */}
  {isTablet && hasQuantity ? (
    <div className="flex items-center justify-between gap-2">
      <Button variant="outline" size="sm" onClick={handleRemoveClick}>
        <Minus className="w-4 h-4" />
      </Button>
      <span className="font-medium">{quantity}</span>
      <Button variant="default" size="sm" onClick={handleAddClick}>
        <Plus className="w-4 h-4" />
      </Button>
    </div>
  ) : (
    <Button variant="default" size="sm" onClick={handleAddClick}>
      <Plus className="w-4 h-4" />
      Ajouter
    </Button>
  )}
</div>
</div>
);
```

# Exemple UI : Variantes de boutons

```
// components/ui/button.tsx
const buttonVariants = cva(
  "inline-flex items-center justify-center ...",
  {
    variants: {
      variant: {
        default: "bg-primary text-primary-foreground hover:bg-primary/90",
        destructive: "bg-destructive text-white hover:bg-destructive/90",
        outline: "border bg-background hover:bg-accent",
        ghost: "hover:bg-accent hover:text-accent-foreground",
      },
      size: {
        default: "h-9 px-4 py-2",
        sm: "h-8 px-3",
        lg: "h-10 px-6",
        icon: "size-9",
      },
    },
    defaultVariants: { variant: "default", size: "default" },
  }
);
```

```
// Utilisation TypeScript avec props typées
function Button({ variant, size, className, ...props }: ButtonProps) {
  return (
    <button
      className={cn(buttonVariants({ variant, size, className }))} {...props}
    />
  );
}
```



# Exemple UI : Variantes de boutons (disclaimer)



Filtres Actifs

∅ Régime alimentaire

Vegetarian icon Végétarien Vegan icon Végan Gluten-free icon Sans gluten

♡ Caractéristiques

Light icon Léger Local products icon Produits locaux Spicy icon Épicé

🌐 Type de cuisine

French flag icon Française Italian flag icon Italienne Asian flag icon Asiatique Mediterranean flag icon Méditerranéenne

# Conclusion

- Conception axée sur les besoins identifiés des utilisateurs du système
- Adaptations diverses proposées pour s'adapter au support ou aux situations d'urgence, critères d'âge ou préférences des utilisateurs
- Bonne réutilisabilité globale du code mais nombreuses possibilités d'amélioration pour “humaniser” et uniformiser le code
- Application fonctionnelle prête à communiquer avec un back réel