

as # 📁 Dossier Projet - SOUNDORA

## Titre Professionnel DWWM (Développeur Web et Web Mobile)

---

### SOMMAIRE

---

1. [Présentation du projet](#)
  2. [Contexte et objectifs](#)
  3. [Cahier des charges](#)
  4. [Stack technique](#)
  5. [SEO et UI/UX](#)
  6. [Architecture de l'application](#)
  7. [Modèle de données \(MCD/MLD\)](#)
  8. [Wireframes et maquettes](#)
  9. [Les 8 compétences professionnelles](#)
  10. [Extraits de code commentés](#)
  11. [Difficultés rencontrées et solutions](#)
  12. [Veille technologique](#)
  13. [Bilan et perspectives](#)
- 

## 1. Présentation du projet

---

### Qu'est-ce que Soundora ?

**Soundora** est une plateforme e-commerce dédiée à la vente d'instruments de musique et d'accessoires audio. Le projet propose une expérience utilisateur moderne, intuitive et sécurisée.

#### Informations générales

Information	Détail
<b>Nom du projet</b>	Soundora
<b>Type</b>	Site e-commerce
<b>Secteur</b>	Vente d'instruments de musique
<b>Durée du projet</b>	12 semaines
<b>Développeur</b>	BRUNET Bastien

#### Fonctionnalités principales

- Catalogue de produits (guitares, basses, batteries, amplis, pédales, etc.)
- Recherche et filtres par catégories/marques
- Système d'authentification (inscription, connexion)

- Panier d'achat
  - Paiement sécurisé via Stripe
  - Interface responsive (mobile et desktop)
  - Pages informatives (À propos, FAQ, Contact, Services)
- 

## 2. Contexte et objectifs

---

### Contexte

Dans le cadre de la formation Développeur Web et Web Mobile, j'ai réalisé ce projet de site e-commerce pour mettre en pratique les compétences acquises :

- Développement front-end avec un framework moderne
- Développement back-end avec Node.js
- Conception et gestion d'une base de données
- Intégration de services tiers (paiement, authentification)

### Objectifs pédagogiques

Objectif	Description	Statut
Front-end	Créer une interface utilisateur moderne et responsive	<input checked="" type="checkbox"/>
Back-end	Développer une API RESTful sécurisée	<input checked="" type="checkbox"/>
BDD	Concevoir et implémenter une base de données relationnelle	<input checked="" type="checkbox"/>
Auth	Mettre en place un système d'authentification	<input checked="" type="checkbox"/>
Paiement	Intégrer un système de paiement en ligne	<input checked="" type="checkbox"/>
Déploiement	Préparer l'application pour la production	<input checked="" type="checkbox"/>

### Public cible

- Musiciens amateurs et professionnels
  - Débutants cherchant leur premier instrument
  - Studios d'enregistrement
- 

## 3. Cahier des charges

---

### 3.1. Fonctionnalités Frontend (Angular)

Fonctionnalité	Description	Priorité
Catalogue produits	Listing, détails, images	Haute
Recherche	Barre de recherche avec filtres	Haute

Fonctionnalité	Description	Priorité
Panier	Ajout, suppression, modification quantité	Haute
Authentification	Inscription, connexion, déconnexion	Haute
Paiement Stripe	Checkout sécurisé	Haute
Responsive	Adaptation mobile/tablet/desktop	Haute
Pages statiques	FAQ, À propos, Contact, Services	Moyenne

### 3.2. Fonctionnalités Backend (Node.js/Express)

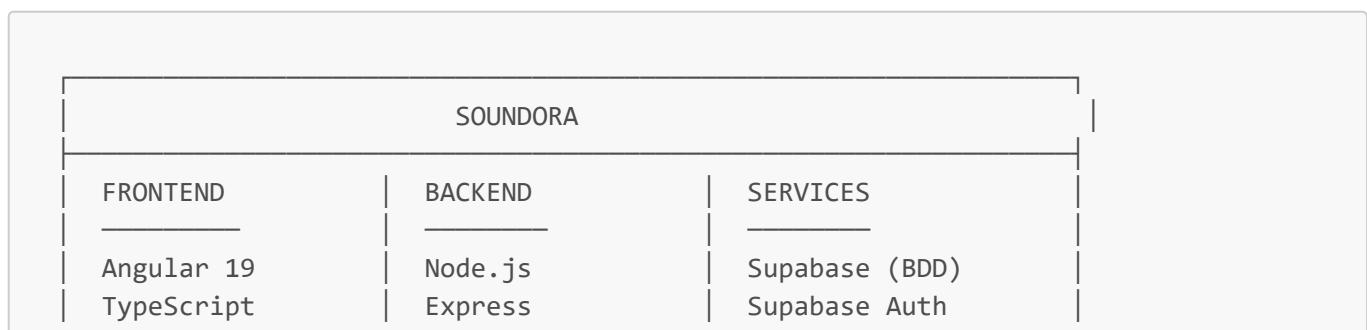
Route API	Méthode	Description
/api/products	GET	Liste des produits avec filtres
/api/products/:slug	GET	Détail d'un produit
/api/categories	GET	Liste des catégories
/api/brands	GET	Liste des marques
/api/auth/register	POST	Inscription utilisateur
/api/auth/login	POST	Connexion utilisateur
/api/cart	GET/POST/PUT/DELETE	Gestion du panier
/api/orders	POST	Création de commande
/api/stripe/create-checkout	POST	Création session Stripe

### 3.3. Contraintes techniques

- Responsive** : Mobile-first, breakpoints à 768px et 1200px
- Sécurité** : JWT, validation des entrées, protection CORS
- Performance** : Pagination, lazy loading des images
- SEO** : URLs propres (slugs), meta descriptions

## 4. Stack technique

### 4.1. Vue d'ensemble



HTML5/CSS3	API REST	Stripe (Paiement)
RxJS	JWT	GitHub (Versioning)

## 4.2. Technologies utilisées

### Frontend

Technologie	Version	Utilisation
<b>Angular</b>	19+	Framework SPA
<b>TypeScript</b>	5.x	Langage typé
<b>RxJS</b>	7.x	Programmation réactive
<b>HTML5</b>	-	Structure des pages
<b>CSS3</b>	-	Styles, Flexbox, Grid

### Backend

Technologie	Version	Utilisation
<b>Node.js</b>	18+	Runtime JavaScript
<b>Express</b>	4.x	Framework web
<b>Supabase JS</b>	2.x	Client base de données
<b>Stripe</b>	-	API de paiement
<b>JWT</b>	-	Authentification

### Base de données

Technologie	Utilisation
<b>Supabase</b>	Backend-as-a-Service
<b>PostgreSQL</b>	Base de données relationnelle

### Outils de développement

Outil	Utilisation
<b>VS Code</b>	IDE principal
<b>Git/GitHub</b>	Versioning
<b>Postman</b>	Tests API
<b>Chrome DevTools</b>	Debug frontend

## 5. SEO et UI/UX

### 5.1. Stratégie SEO (Search Engine Optimization)

Optimisations techniques mises en place

Technique SEO	Implémentation	Bénéfice
<b>URLs propres (slugs)</b>	/products/fender-stratocaster-player au lieu de /products/123	URLs lisibles et mémorisables
<b>Structure HTML sémantique</b>	<header>, <nav>, <main>, <article>, <footer>	Meilleure compréhension par les moteurs
<b>Balises meta</b>	<title> et <meta description> dynamiques	Affichage optimisé dans les résultats
<b>Attributs alt</b>	Images avec descriptions alt="Guitare Fender Stratocaster"	Accessibilité + indexation images
<b>Hiérarchie des titres</b>	H1 > H2 > H3 respectée	Structure logique du contenu
<b>Performance</b>	Lazy loading, compression images	Core Web Vitals améliorés
<b>Mobile-first</b>	Design responsive prioritaire mobile	Favorisé par Google (Mobile-First Index)

Exemple de balises meta dynamiques

```
// Dans un composant Angular - Mise à jour du titre de page
import { Title, Meta } from "@angular/platform-browser";

export class ProductDetailComponent {
  constructor(
    private titleService: Title,
    private metaService: Meta,
  ) {}

  ngOnInit() {
    // Titre dynamique selon le produit
    this.titleService.setTitle(`#${this.product.name} | Soundora`);

    // Meta description
    this.metaService.updateTag({
      name: "description",
      content: `Achetez ${this.product.name} - ${this.product.short_description}.
Livraison rapide.`,
    });
  }
}
```

## Structure URL optimisée

Bonnes pratiques :

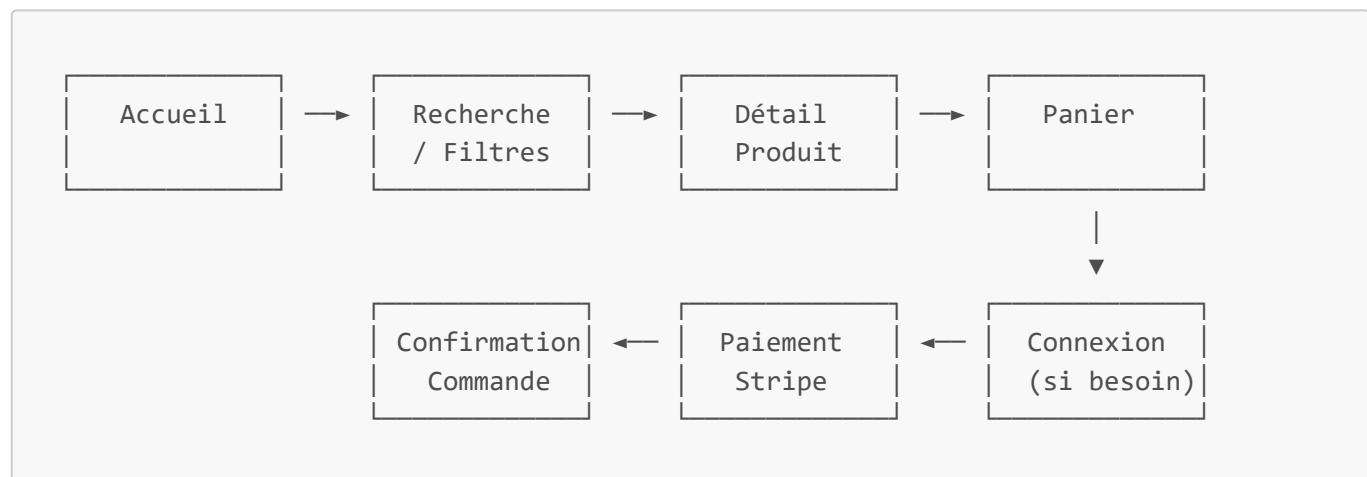
/products/guitares	→ Liste guitares
/products/guitares/fender-stratocaster	→ Détail produit
/categories/amplis	→ Catégorie amplis
/brands/gibson	→ Marque Gibson

À éviter :

/products?id=123  
/p/45678  
/product-detail.php?ref=ABC

## 5.2. Principes UI/UX appliqués

### Parcours utilisateur optimisé



### Principes UX appliqués

Principe	Application dans Soundora
<b>Clarté</b>	Interface épurée, textes lisibles, hiérarchie visuelle claire
<b>Cohérence</b>	Même style de boutons, couleurs et espacements sur toutes les pages
<b>Feedback</b>	Messages de confirmation (ajout panier, connexion réussie, erreurs)
<b>Accessibilité</b>	Contrastes suffisants, navigation au clavier, attributs ARIA
<b>Rapidité</b>	Chargement rapide, indicateurs de loading pendant les requêtes
<b>Mobile-first</b>	Conception prioritaire pour mobile, adaptation desktop ensuite

### Charte graphique

Élément	Valeur	Utilisation
---------	--------	-------------

Élément	Valeur	Utilisation
<b>Couleur primaire</b>	#1a1a2e	Navigation, textes principaux
<b>Couleur accent</b>	#e94560	Boutons CTA, prix, éléments importants
<b>Couleur secondaire</b>	#16213e	Arrière-plans, sections
<b>Police principale</b>	Segoe UI, Roboto	Corps de texte
<b>Police titres</b>	Segoe UI Semibold	Titres et en-têtes
<b>Border radius</b>	8px	Cartes, boutons, inputs
<b>Espacements</b>	8px, 16px, 24px, 32px	Grille de 8px

## Composants UI réutilisables

Composant	Description
<b>Card produit</b>	Image, nom, prix, bouton ajout panier
<b>Bouton primaire</b>	Rouge (#e94560), hover avec effet
<b>Bouton secondaire</b>	Bordure, fond transparent
<b>Input field</b>	Label flottant, validation visuelle
<b>Toast notification</b>	Messages temporaires en bas de page
<b>Loading spinner</b>	Indicateur de chargement
<b>Badge</b>	Quantité panier, promotions

## Design responsive

```

/* Breakpoints utilisés */
/* Mobile (default) */
.container {
  padding: 16px;
}

/* Tablet - 768px */
@media (min-width: 768px) {
  .products-grid {
    grid-template-columns: repeat(2, 1fr);
  }
}

/* Desktop - 1200px */
@media (min-width: 1200px) {
  .products-grid {
    grid-template-columns: repeat(4, 1fr);
  }
  .container {

```

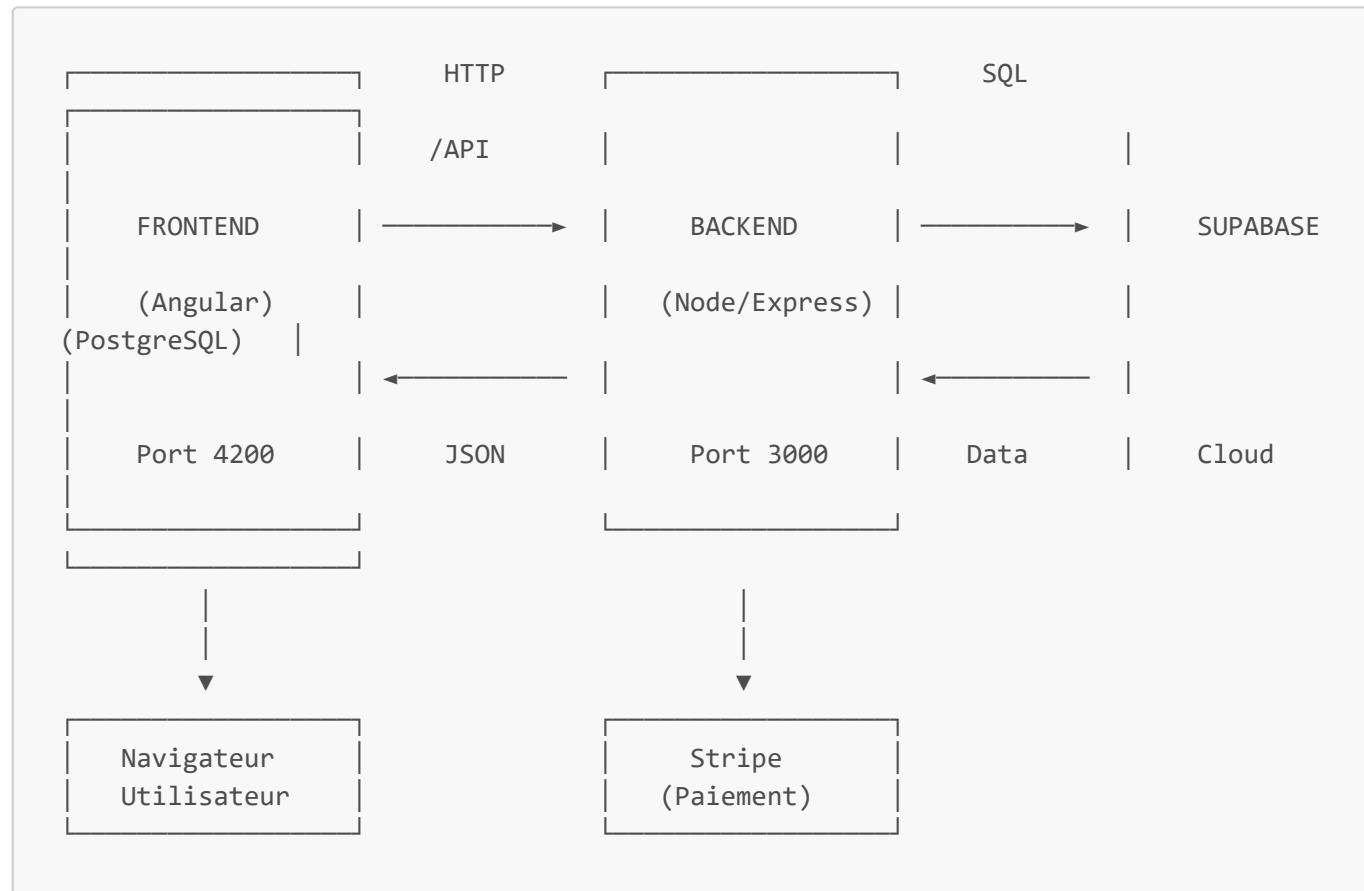
```
max-width: 1400px;
}
}
```

## Accessibilité (A11Y)

Critère	Implémentation
<b>Contraste</b>	Ratio minimum 4.5:1 pour les textes
<b>Focus visible</b>	Outline sur les éléments interactifs
<b>Labels</b>	Tous les inputs ont un label associé
<b>Alt text</b>	Toutes les images ont une description
<b>Navigation clavier</b>	Tab pour naviguer, Enter pour valider
<b>ARIA</b>	aria-label, role sur les éléments complexes

# 6. Architecture de l'application

## 5.1. Architecture globale



## 5.2. Structure du projet Frontend

```

soundora-frontend/
├── src/
│   ├── app/
│   │   ├── components/          # Composants Angular
│   │   │   ├── navbar/           # Navigation principale
│   │   │   ├── top-navbar/        # Navigation secondaire
│   │   │   ├── product-list/      # Liste des produits
│   │   │   ├── product-detail/    # Détail d'un produit
│   │   │   ├── login/             # Page de connexion
│   │   │   ├── register/          # Page d'inscription
│   │   │   ├── contact/           # Page de contact
│   │   │   ├── about/              # Page À propos
│   │   │   ├── faq/                # Page FAQ
│   │   │   └── services/          # Page Services
│   │   ├── services/            # Services Angular
│   │   │   ├── auth.service.ts     # Authentification
│   │   │   ├── product.service.ts  # Gestion produits
│   │   │   ├── cart.service.ts     # Gestion panier
│   │   │   └── category.service.ts # Gestion catégories
│   │   └── app.routes.ts         # Configuration des routes
│   └── assets/
│       └── images/               # Images du site
└── angular.json                 # Configuration Angular

```

## 5.3. Structure du projet Backend

```

soundora-backend/
├── server.js                  # Point d'entrée de l'application
├── config/
│   ├── supabase.js              # Configuration Supabase
│   └── stripe.js                # Configuration Stripe
├── controllers/                # Logique métier (MVC)
│   ├── authController.js        # Authentification
│   ├── productSupabaseController.js # Produits
│   ├── categoryController.js    # Catégories
│   ├── brandController.js       # Marques
│   ├── cartController.js        # Panier
│   ├── orderController.js       # Commandes
│   └── stripeController.js      # Paiements
├── middleware/
│   └── checkSupabaseAuth.js     # Vérification JWT
└── routes/
    └── api.js                   # Définition des routes API

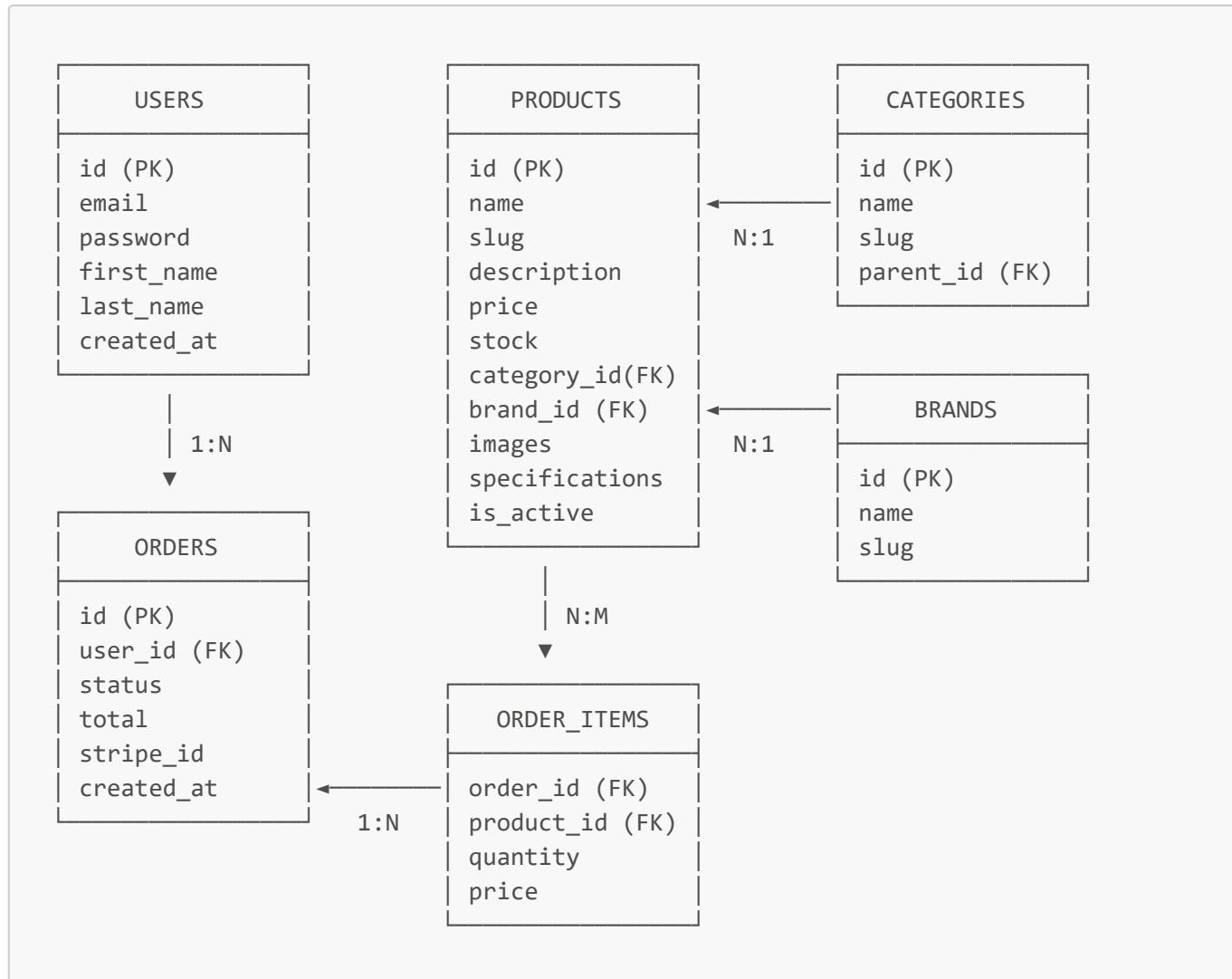
```

---

## 7. Modèle de données (MCD/MLD)

---

### 6.1. MCD (Modèle Conceptuel de Données)



## 6.2. Tables principales

Table **products**

Colonne	Type	Description
id	UUID	Identifiant unique
name	VARCHAR	Nom du produit
slug	VARCHAR	URL-friendly name
description	TEXT	Description longue
short_description	VARCHAR	Description courte
price	DECIMAL	Prix en euros
stock	INTEGER	Quantité en stock
category_id	UUID (FK)	Catégorie
brand_id	UUID (FK)	Marque
model	VARCHAR	Modèle

Colonne	Type	Description
color	VARCHAR	Couleur
images	JSONB	Tableau d'URLs
specifications	JSONB	Caractéristiques
is_active	BOOLEAN	Produit visible
is_featured	BOOLEAN	Produit mis en avant
created_at	TIMESTAMP	Date de création

## Table categories

Colonne	Type	Description
id	UUID	Identifiant unique
name	VARCHAR	Nom de la catégorie
slug	VARCHAR	URL-friendly name
parent_id	UUID (FK)	Catégorie parente (nullable)
is_active	BOOLEAN	Catégorie visible

## Table brands

Colonne	Type	Description
id	UUID	Identifiant unique
name	VARCHAR	Nom de la marque
slug	VARCHAR	URL-friendly name
description	TEXT	Description
is_active	BOOLEAN	Marque visible

## 8. Wireframes et maquettes

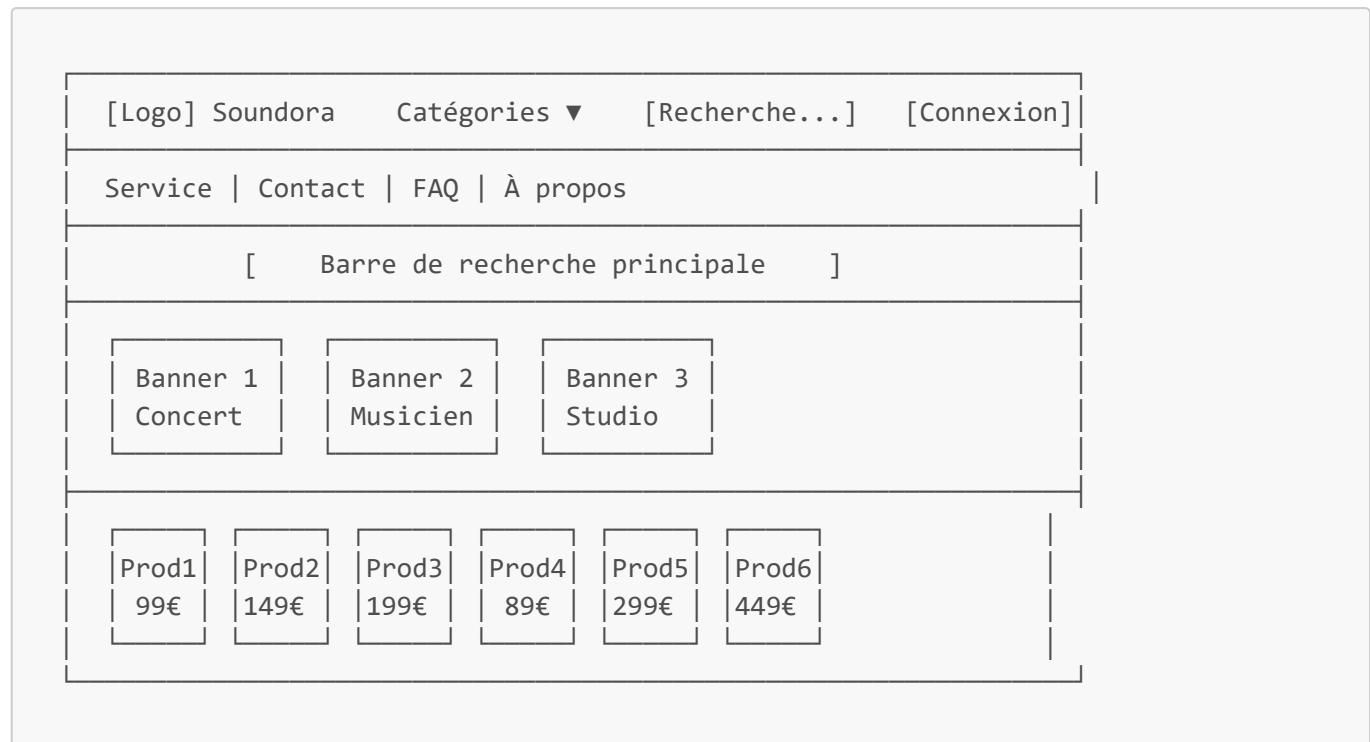
### À RÉALISER

**Note :** Cette section doit contenir vos wireframes. Vous pouvez les créer avec :

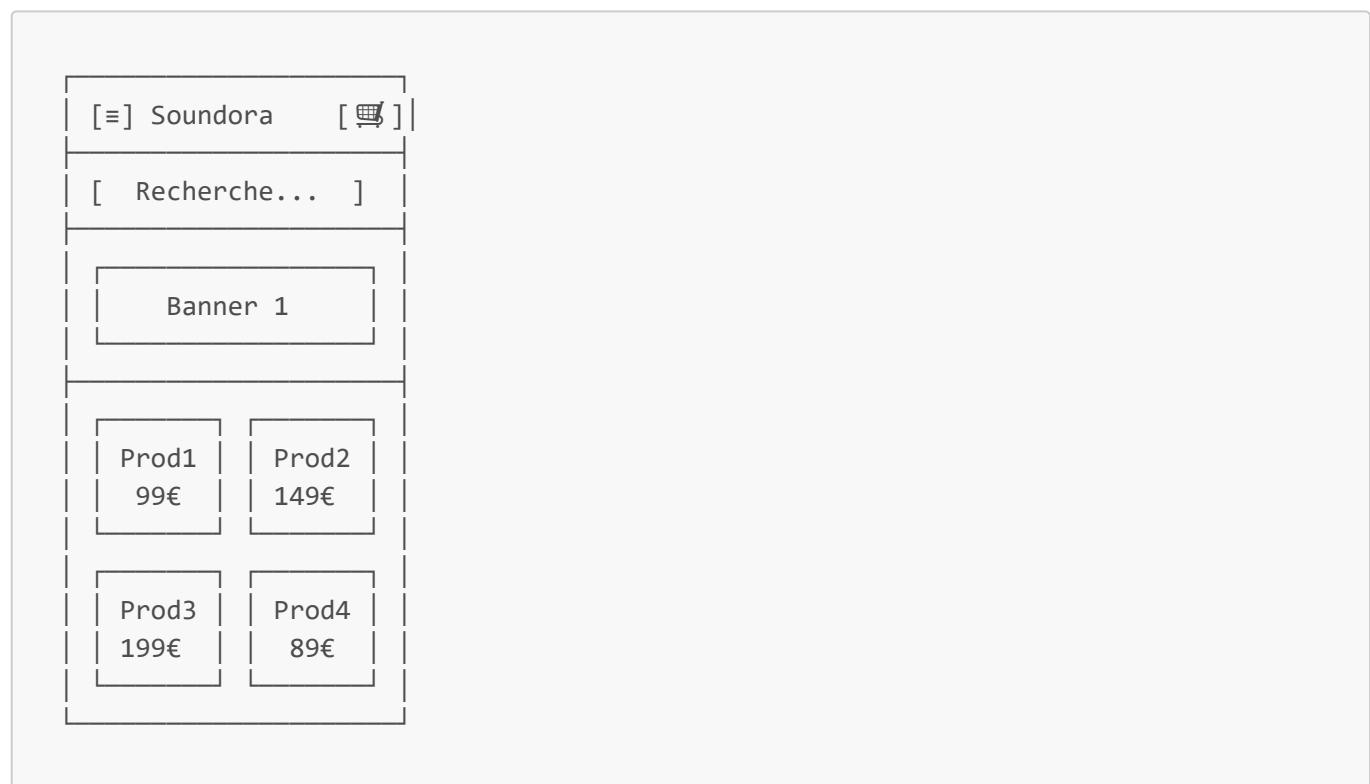
- **Figma** (gratuit, en ligne)
- **Balsamiq** (spécialisé wireframes)
- **Excalidraw** (gratuit, simple)
- **Papier/crayon** (puis scanner)

## Pages à maquetter

## 7.1. Page d'accueil (Desktop)



## 7.2. Page d'accueil (Mobile)



## 7.3. Page détail produit

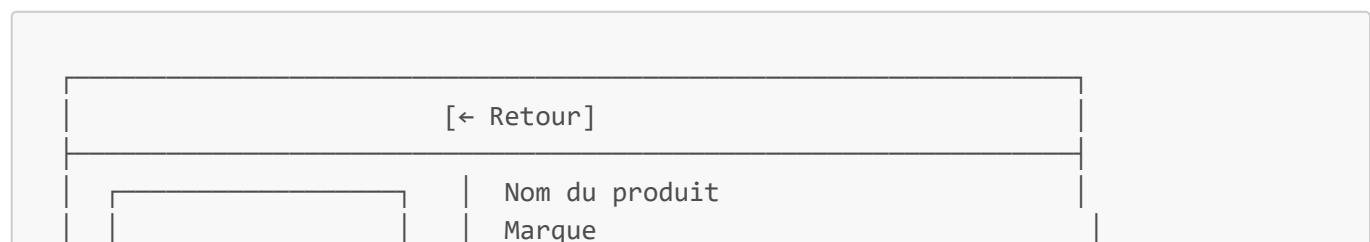


IMAGE PRODUIT	Prix: 599 €
Description du produit...	
[ AJOUTER AU PANIER ]	

#### 7.4. Page connexion

CONNEXION
Email
<input type="text"/>
Mot de passe
<input type="password"/>
[ SE CONNECTER ]
Pas encore de compte ? S'inscrire

---

## 9. Les 8 compétences professionnelles

---

### CP1 - Maquetter une application

**Ce que j'ai fait :**

- Création de wireframes pour les pages principales
- Définition de la charte graphique (couleurs, typographie)
- Parcours utilisateur défini

**Preuves :** Wireframes ci-dessus (Section 7)

---

### CP2 - Réaliser une interface utilisateur web statique et adaptable

**Ce que j'ai fait :**

- Structure HTML5 sémantique (`<header>`, `<nav>`, `<main>`, `<footer>`)

- CSS3 avec Flexbox et Grid
- Design responsive avec media queries

### Exemple de code :

```
<!-- Structure sémantique HTML5 -->
<header class="navbar">
  <nav class="navbar-menu">
    <a routerLink="/products">Catalogue</a>
  </nav>
</header>
<main class="product-list">
  <!-- Contenu principal -->
</main>
```

```
/* Responsive avec CSS Grid */
.products-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));
  gap: 1.5rem;
}

/* Media query pour mobile */
@media (max-width: 768px) {
  .navbar-menu {
    display: none;
  }
  .menu-burger {
    display: block;
  }
}
```

---

## CP3 - Développer une interface utilisateur web dynamique

### Ce que j'ai fait :

- Composants Angular avec TypeScript
- Gestion des événements (click, submit)
- Communication API avec HttpClient
- Programmation réactive avec RxJS

### Exemple de code :

```
// Service Angular - Appel API avec Observable
getProducts(page: number, limit: number, filters?: any):
Observable<ProductsResponse> {
  let params = new HttpParams()
```

```
.set('page', page.toString())
.set('limit', limit.toString());

if (filters?.search) {
    params = params.set('search', filters.search);
}

return this.http.get<ProductsResponse>(this.apiUrl, { params });
}
```

---

## CP4 - Réaliser une interface utilisateur avec une solution e-commerce

### Ce que j'ai fait :

- Catalogue de produits avec filtres
- Panier d'achat fonctionnel
- Processus de checkout
- Intégration Stripe pour les paiements

### Fonctionnalités e-commerce :

- Affichage des produits par catégorie/marque
- Recherche de produits
- Gestion du panier (ajout, suppression, quantité)
- Paiement sécurisé

---

## CP5 - Créer une base de données

### Ce que j'ai fait :

- Conception du MCD (Modèle Conceptuel de Données)
- Crédit des tables dans Supabase (PostgreSQL)
- Définition des relations (clés étrangères)
- Indexation pour les performances

### Tables créées :

- `products` - Catalogue produits
- `categories` - Catégories (avec sous-catégories)
- `brands` - Marques
- `orders` - Commandes
- `order_items` - Lignes de commande

---

## CP6 - Développer les composants d'accès aux données

### Ce que j'ai fait :

- Requêtes CRUD avec Supabase JS SDK

- Jointures entre tables
- Filtrage et pagination
- Gestion des erreurs

### Exemple de code :

```
// Requête Supabase avec jointures et filtres
let query = supabase
  .from("products")
  .select(
    '*',
    category:categories!category_id(id, name, slug),
    brand:brands!brand_id(id, name, slug)
  ,
  { count: "exact" },
)
.eq("is_active", true);

// Filtre par catégorie
if (category) {
  query = query.eq("category_id", categoryData.id);
}

// Pagination
const from = (page - 1) * limit;
const to = from + limit - 1;
query = query.range(from, to);
```

## CP7 - Développer la partie back-end d'une application web

### Ce que j'ai fait :

- API RESTful avec Express
- Architecture MVC (Model-View-Controller)
- Routes organisées par fonctionnalité
- Middlewares de sécurité

### Structure de l'API :

```
// routes/api.js
router.get("/products", productController.getAllProducts);
router.get("/products/:slug", productController.getProductBySlug);
router.post("/auth/register", authController.register);
router.post("/auth/login", authController.login);
router.post("/orders", checkAuth, orderController.createOrder);
```

## CP8 - Élaborer des composants dans une application e-commerce

### Ce que j'ai fait :

- Authentification avec Supabase Auth + JWT
- Intégration Stripe (checkout, webhooks)
- Sécurisation des routes sensibles
- Validation des données

### Exemple - Authentification :

```
// Connexion avec Supabase Auth
const { data, error } = await supabase.auth.signInWithEmailAndPassword({
  email: email,
  password: password,
});

// Retour du token JWT
res.json({
  success: true,
  user: data.user,
  access_token: data.session.access_token,
});
```

### Exemple - Paiement Stripe :

```
// Création d'une session Stripe Checkout
const session = await stripe.checkout.sessions.create({
  payment_method_types: ["card"],
  line_items: cartItems.map((item) => ({
    price_data: {
      currency: "eur",
      product_data: { name: item.name },
      unit_amount: item.price * 100,
    },
    quantity: item.quantity,
  })),
  mode: "payment",
  success_url: `${frontendUrl}/success`,
  cancel_url: `${frontendUrl}/cancel`,
});
```

---

## 10. Extraits de code commentés

### 9.1. Composant Angular - Liste des produits

```
/**  
 * ProductListComponent  
 * Affiche la liste des produits avec filtres et pagination  
 */  
  
@Component({  
  selector: "app-product-list",  
  standalone: true,  
  imports: [CommonModule, RouterModule],  
  templateUrl: "./product-list.component.html",  
})  
export class ProductListComponent implements OnInit {  
  // État du composant  
  products: Product[] = []; // Liste des produits  
  isLoading: boolean = false; // Indicateur de chargement  
  currentPage: number = 1; // Page actuelle  
  totalPages: number = 0; // Nombre total de pages  
  
  constructor(  
    private productService: ProductService, // Service d'accès aux données  
    private route: ActivatedRoute, // Accès aux paramètres URL  
  ) {}  
  
  ngOnInit(): void {  
    // Écoute les changements de paramètres URL  
    this.route.queryParams.subscribe((params) => {  
      this.selectedCategory = params["category"] || "";  
      this.loadProducts(); // Recharge les produits  
    });  
  }  
  
  loadProducts(): void {  
    this.isLoading = true;  
  
    // Appel au service (Observable)  
    this.productService  
      .getProducts(this.currentPage, 12, {  
        category: this.selectedCategory,  
      })  
      .subscribe({  
        next: (response) => {  
          this.products = response.data;  
          this.isLoading = false;  
        },  
        error: (err) => {  
          console.error("Erreur:", err);  
          this.isLoading = false;  
        },  
      });  
  }  
}
```

## 9.2. Contrôleur Express - Produits

```
/**  
 * getAllProducts - Récupère les produits avec filtres et pagination  
 * GET /api/products?page=1&limit=10&category=guitares&search=fender  
 */  
export const getAllProducts = async (req, res) => {  
  try {  
    // Extraction des paramètres de requête  
    const { page = 1, limit = 10, category, search } = req.query;  
  
    // Construction de la requête Supabase  
    let query = supabase  
      .from("products")  
      .select()  
      .  
      *,  
      category:categories!category_id(id, name, slug),  
      brand:brands!brand_id(id, name, slug)  
      ,  
      { count: "exact" },  
    )  
    .eq("is_active", true);  
  
    // Filtre par catégorie (si fourni)  
    if (category) {  
      const { data: cat } = await supabase  
        .from("categories")  
        .select("id")  
        .eq("slug", category)  
        .single();  
  
      if (cat) {  
        query = query.eq("category_id", cat.id);  
      }  
    }  
  
    // Recherche textuelle (si fournie)  
    if (search) {  
      query = query.or(`name.ilike.%${search}%,description.ilike.%${search}%`);  
    }  
  
    // Pagination  
    const from = (page - 1) * limit;  
    query = query.range(from, from + limit - 1);  
  
    // Exécution  
    const { data, error, count } = await query;  
  
    if (error) throw error;  
  
    // Réponse
```

```

res.json({
  success: true,
  data: data,
  pagination: {
    currentPage: parseInt(page),
    totalPages: Math.ceil(count / limit),
    total: count,
  },
});
} catch (error) {
  res.status(500).json({ success: false, message: error.message });
}
};

```

## 11. Difficultés rencontrées et solutions

Difficulté	Contexte	Solution mise en œuvre
<b>Images non affichées</b>	Les URLs en BDD étaient fictives	Création d'un mapping local <code>productImageMap</code> qui associe chaque modèle à son image locale
<b>Gestion des catégories parent/enfant</b>	Filtrer par catégorie parente devait inclure les sous-catégories	Requête récursive pour récupérer les IDs des sous-catégories puis utilisation de <code>IN</code>
<b>Authentification persistante</b>	Session perdue au rafraîchissement	Utilisation de <code>localStorage</code> pour stocker le token et <code>BehaviorSubject</code> pour l'état utilisateur
<b>CORS</b>	Requêtes bloquées entre frontend et backend	Configuration du middleware <code>cors</code> dans Express avec les bons origins
<b>Responsive navbar</b>	Menu trop large sur mobile	Implémentation d'un menu burger avec toggle

## 12. Veille technologique

### Sources utilisées

Source	Type	Utilisation
<b>MDN Web Docs</b>	Documentation	Référence HTML/CSS/JavaScript
<b>Angular.io</b>	Documentation officielle	Guide et API Angular
<b>Stack Overflow</b>	Forum Q&A	Résolution de problèmes
<b>Supabase Docs</b>	Documentation	API Supabase, Auth, RLS
<b>Stripe Docs</b>	Documentation	Intégration paiement

Source	Type	Utilisation
<b>GitHub</b>	Projets open source	Exemples de code
<b>Dev.to / Medium</b>	Blogs tech	Articles et tutoriels

## Méthode de veille

1. **Quotidienne** : Newsletters (JavaScript Weekly, Angular Blog)
  2. **Hebdomadaire** : Lecture d'articles sur Dev.to
  3. **Projet** : Recherche ciblée sur Stack Overflow pour les problèmes spécifiques
- 

## 13. Bilan et perspectives

### Ce que j'ai appris

- Développement d'une SPA complète avec Angular
- Création d'une API RESTful avec Node.js/Express
- Utilisation d'un BaaS (Supabase) pour la BDD et l'auth
- Intégration d'un système de paiement (Stripe)
- Gestion de projet avec Git/GitHub
- Architecture MVC et bonnes pratiques

### Points forts du projet

- Interface utilisateur moderne et responsive
- Code bien structuré et commenté
- Authentification sécurisée
- Paiement fonctionnel

### Améliorations futures

Amélioration	Description	Priorité
Tests unitaires	Ajouter des tests avec Jasmine/Jest	Haute
PWA	Transformer en Progressive Web App	Moyenne
Admin dashboard	Interface de gestion des produits	Haute
Avis clients	Système de notation et commentaires	Moyenne
Wishlist	Liste de souhaits utilisateur	Basse
Mode sombre	Thème dark complet	Basse

## Conclusion

Le projet Soundora m'a permis de mettre en pratique l'ensemble des compétences du titre DWWM. J'ai pu développer une application web complète, du maquettage à l'intégration de services tiers, en passant par la

conception de la base de données et le développement front/back.

---

## Annexes

---

- **Lien GitHub** : [URL du repository]
  - **Site en ligne** : [URL de démo si disponible]
  - **Captures d'écran** : Voir dossier [/documentation/screenshots/](#)
- 

*Document rédigé dans le cadre du Titre Professionnel DWWM Date : Février 2026*