SOUNDORA

PROJET RÉALISÉ PAR

BRUNET BASTIEN DANS LE CADRE DU TITRE DE DÉVELOPPEUR WEB ET WEB MOBILE

ÉCOLE LA PLATEFORME 2025

Introduction

Dans le cadre de ma formation de ***Développeur Web et Web Mobile****, j'ai eu l'opportunité de mettre en pratique les compétences acquises au fil des mois à travers un* ***projet de fin d'année****. Pour ce travail, j'ai choisi de développer un* ***site e-commerce dédié à la vente d'instruments de musique****.*

Ce choix s'est imposé naturellement, car il allie deux de mes grandes passions : ***la programmation informatique*** *et* ***la musique****. Depuis mon enfance, la musique occupe une place importante dans ma vie. Je pratique notamment la* ***guitare*** *et la* ***basse****, ce qui m'a permis d'acquérir une bonne connaissance des instruments, de leurs spécificités, et des besoins des musiciens, débutants comme confirmés.*

En parallèle, ma formation m'a permis de maîtriser les bases essentielles du développement web et mobile : du design d'interface à l'intégration front-end, en passant par la gestion des données et la mise en place de fonctionnalités dynamiques côté serveur. Ce projet représente donc pour moi une ***synthèse concrète de mon parcours****, mais aussi une opportunité de créer un produit qui pourrait réellement répondre aux attentes d'une communauté de passionnés.*

Ce dossier retrace les différentes étapes de la conception et du développement de ce site e-commerce, depuis l'analyse des besoins jusqu'à la mise en ligne, en passant par les choix techniques, la modélisation, et les tests.

Partie 1 : Développement Front-End

1.1 Installation et configuration de l'environnement de travail

Pour la réalisation de l'interface utilisateur de mon site e-commerce d'instruments de musique, j'ai choisi d'utiliser ***Angular 19*** *comme framework front-end. Ce choix s'explique par la robustesse de ce framework, sa gestion puissante du routing, des formulaires et de l'état applicatif, ainsi que sa parfaite compatibilité avec TypeScript.*

L'environnement de développement a été structuré de manière modulaire pour favoriser la lisibilité, la maintenabilité et l'évolutivité du projet. Voici les principaux éléments mis en place : **1.2 Maquettage des interfaces utilisateur web**

Avant de passer au développement, une étape importante de maquettage des interfaces a été réalisée.Et l'authentification avec Supabase Auth :

- *Intégration complète de l'**authentification JWT** via **Supabase Auth**.*
- *Gestion du **login**, de l'**inscription** et du **logout**, avec persistance de l'état utilisateur dans le `localStorage` et les guards Angular (`AuthGuard`) pour sécuriser les routes protégées.*

c. Paiement avec Stripe

- *Intégration de **Stripe Checkout** pour le paiement sécurisé.*
- *Lors de la validation du panier, les informations sont envoyées au backend Express qui crée une **session Stripe**. L'utilisateur est ensuite redirigé vers la page de paiement Stripe.*
- *Une fois le paiement validé, une **webhook Stripe** notifie le backend, qui met à jour la commande dans Supabase.*

d. Expérience utilisateur (UX) & fonctionnalités avancées

- ***Feedbacks en temps réel** pour les formulaires (validation de champs, messages d'erreur).*
- *Composants réactifs avec **RxJS** pour gérer les flux de données (produits, utilisateur, panier).*
- *Utilisation d'animations légères et transitions pour améliorer l'expérience visuelle.*

---

*Ce front-end, entièrement développé avec Angular et TypeScript, constitue une interface **moderne, rapide et sécurisée**, interfacée proprement avec un back-end Node.js et une base de données Supabase. L'architecture modulaire adoptée permet une **évolutivité** du projet, avec la possibilité future d'ajouter de nouvelles fonctionnalités (messagerie client, avis produit, notifications, etc.).*

. L'objectif était de définir une interface moderne, intuitive et responsive, adaptée aussi bien aux utilisateurs sur ordinateur que sur mobile.

Les maquettes ont permis de définir :

- *La **page d'accueil** avec mise en avant de produits populaires, offres spéciales et un carrousel dynamique.*
- *La **page catalogue**, avec système de filtres (catégories, prix, marques) et tri.*
- *La **fiche produit**, avec photos, description technique, avis clients, et bouton d'ajout au panier.*
- *Les pages **panier**, **commande**, et **paiement**, intégrées avec le système Stripe.*
- *Les pages **authentification** (connexion, inscription), et **espace utilisateur** (profil, historique de commandes).*

Ces maquettes ont ensuite servi de référence pour l'implémentation des composants Angular.

1.3 Réalisation des interfaces statiques

*L'étape suivante a consisté à transformer les maquettes en **interfaces statiques fonctionnelles**, à l'aide d'**Angular**, **HTML**, et **CSS (ou SCSS)**. Angular permet une approche basée sur des **composants réutilisables** et une **architecture modulaire**, facilitant le développement progressif.*

Voici quelques décisions techniques :

- *Utilisation du **système de routing Angular** pour gérer la navigation entre les pages.*
- *Mise en place d'un **design responsive** à l'aide de Flexbox, Grid, et de media queries.*

- *Utilisation de **SCSS** pour une meilleure organisation des styles et l'utilisation de variables.*
- *Intégration de **Google Fonts** et d'icônes vectorielles (SVG) pour l'esthétique générale.*

Les composants ont été regroupés par fonction (navigation, affichage produit, panier, formulaire, etc.), afin de garantir la cohérence de l'interface et la réutilisabilité du code.

1.4 Développement de la partie dynamique de l'interface

*Une fois les bases statiques en place, j'ai intégré la **logique dynamique** grâce à **TypeScript**, aux **services Angular**, et à l'**interaction avec le backend Node.js / Supabase** via des appels API sécurisés.*

Voici les principales fonctionnalités dynamiques développées :

a. Interaction avec l'API backend (Express / Supabase)

- *Les données des produits sont récupérées depuis Supabase (PostgreSQL) via une API Express personnalisée, hébergée dans le conteneur `soundora-back/`.*
- *Un **service Angular** dédié gère les appels HTTP (via `HttpClient`) pour les opérations CRUD : affichage des produits, ajout au panier, historique des commandes, etc.*

b. Authentification avec Supabase Auth

- *Intégration complète de l'**authentification JWT** via **Supabase Auth**.*
- *Gestion du **login**, de l'**inscription** et du **logout**, avec persistance de l'état utilisateur dans le `localStorage` et les guards Angular (`AuthGuard`) pour sécuriser les routes protégées.*

c. Paiement avec Stripe

- *Intégration de **Stripe Checkout** pour le paiement sécurisé.*
- *Lors de la validation du panier, les informations sont envoyées au backend Express qui crée une **session Stripe**. L'utilisateur est ensuite redirigé vers la page de paiement Stripe.*
- *Une fois le paiement validé, une **webhook Stripe** notifie le backend, qui met à jour la commande dans Supabase.*

d. Expérience utilisateur (UX) & fonctionnalités avancées

- ***Feedbacks en temps réel** pour les formulaires (validation de champs, messages d'erreur).*
- *Composants réactifs avec **RxJS** pour gérer les flux de données (produits, utilisateur, panier).*
- *Utilisation d'animations légères et transitions pour améliorer l'expérience visuelle.*

---

*Ce front-end, entièrement développé avec Angular et TypeScript, constitue une interface **moderne, rapide et sécurisée**, interfacée proprement avec un back-end Node.js et une base de données Supabase. L'architecture modulaire adoptée permet une **évolutivité** du projet, avec la possibilité future d'ajouter de nouvelles fonctionnalités (messagerie client, avis produit, notifications, etc.).*

- ***Visual Studio Code** comme éditeur principal, enrichi d'extensions Angular Essentials, Prettier, ESLint, Angular Language Service, etc.*
- ***Angular CLI** pour la création du projet (ng new soundora-frontend) et la génération de composants, services et modules.*
- *Structure du code respectant les bonnes pratiques Angular, avec séparation claire des **composants**, **services**, **modules**, **interfaces**, **environnements**, et **assets.***

Partie 2 : Développement Back-End

2.1 Mise en place d'une base de données relationnelle

*Pour la gestion des données de mon site e-commerce, j'ai opté pour **Supabase**, une plateforme open source basée sur **PostgreSQL**, qui offre une solution complète intégrant la base de données, l'authentification et le stockage.*

*J'ai modélisé la base de données en m'appuyant sur une structure **relationnelle** bien définie, comprenant notamment les tables suivantes :*

- `users` *: informations des utilisateurs (authentification via Supabase Auth)*

- `products` *: catalogue des instruments (nom, description, images, prix, stock…)*
- `categories` *: types d'instruments (basses, guitares, claviers, etc.)*
- `orders` *: commandes passées par les utilisateurs*
- `order_items` *: association entre les commandes et les produits*
- `payments` *: suivi des paiements via Stripe*

Chaque table est liée à l'autre via des ***clés étrangères****, assurant l'intégrité des données. L'outil* ***Supabase Studio*** *m'a permis de gérer visuellement les schémas et les relations, mais j'ai également utilisé des* ***scripts SQL*** *pour automatiser la création de la base et de ses contraintes.*

2.2 Développement des composants d'accès aux données (SQL & NoSQL)

Même si la majorité des données de l'application sont gérées via ***SQL (PostgreSQL)****, certaines fonctionnalités ont été développées dans une logique* ***NoSQL****, notamment pour des données semi-structurées ou temporaires, comme :*

- *Le* ***panier utilisateur côté client****, stocké dans le* `localStorage` *ou dans un store d'état (ex :* `BehaviorSubject`*) côté Angular.*
- *Les* ***logs ou métadonnées non critiques****, que l'on pourrait envisager de stocker via Supabase Storage ou un système de type Firebase dans une version future.*

Côté back-end, les composants d'accès aux données SQL ont été centralisés dans des ***services dédiés*** *au sein du dossier* `controllers/` *ou* `services/`*.*

2.3 Développement des composants métier côté serveur

Le cœur de la logique applicative réside dans le serveur ***Node.js****, propulsé par le framework* ***Express.js****, hébergé dans le dossier* `soundora-back/`*. J'ai structuré l'API en suivant le modèle MVC (Modèle-Vue-Contrôleur) pour une organisation claire et modulaire.*

Les composants métier développés incluent :

- ***Gestion des produits** : ajout, modification, suppression, affichage avec filtres dynamiques.*
- ***Gestion du panier** : traitement des articles sélectionnés côté client, avec vérification côté serveur lors de la commande.*
- ***Création de commandes** : génération d'un enregistrement `order` et de ses `order_items`, association à l'utilisateur connecté.*
- ***Paiement sécurisé via Stripe** : création d'une **session Stripe Checkout** côté serveur, avec redirection vers la page de paiement.*
- ***Webhooks Stripe** : écoute d'événements (`payment_intent.succeeded`, etc.) pour mettre à jour l'état des commandes après paiement.*
- ***Sécurisation des routes** via des **middleware JWT**, en vérifiant les tokens générés par Supabase Auth pour autoriser les opérations.*

- *Un exemple de logique métier :*

```
const createOrder = async (req, res) => {
  const { items, userId } = req.body;
  const total = calculateTotal(items);
  const { data: order, error } = await supabase
    .from('orders')
    .insert([{ user_id: userId, total_price: total }])
    .select();
  // ...
};
```

2.4 Documentation du déploiement d'une application dynamique

*Le déploiement de l'application a été réalisé à l'aide de **Plesk**, une solution d'administration de serveurs web qui facilite la mise en ligne d'applications web, la gestion des domaines, des bases de données et des certificats SSL. Ce choix permet un **déploiement plus accessible** et compatible avec les hébergements mutualisés ou VPS.*

a. Déploiement du front-end Angular

L'application Angular a été compilée en mode production à l'aide de la commande :

```
ng build --configuration production
```

Le contenu du dossier `/dist/soundora-frontend/` *généré par Angular a ensuite été* ***transféré sur le serveur via FTP ou directement via l'interface Plesk****, dans le répertoire racine du domaine ou sous-domaine configuré.*

b. Déploiement du back-end Node.js (Express)

Le back-end Express a été ***déployé manuellement sur le serveur Plesk****, en suivant les étapes suivantes :*

1. ***Création d'un sous-domaine*** *ou domaine réservé au back-end, par exemple* `api.soundora.com`*.*
2. ***Téléversement des fichiers du dossier*** `soundora-back/` *via FTP ou Plesk File Manager.*
3. ***Installation des dépendances Node.js*** *depuis Plesk (onglet Node.js) :*
   - *Sélection de la bonne version de Node.js*
   - *Exécution de* `npm install` *dans le dossier du projet*
4. ***Configuration de l'entrée de l'application*** *:*
   - *Spécification du fichier d'entrée (*`server.js`*)*
   - *Définition des variables d'environnement via l'interface (clé Supabase, clé Stripe, etc.)*
5. ***Démarrage et gestion du processus Node.js*** *via l'interface Plesk (avec redémarrage automatique en cas de problème).*

c. Configuration des variables sensibles

Les clés d'API et secrets (JWT, Supabase, Stripe, etc.) sont ***stockés dans l'interface Plesk*** *sous forme de* ***variables d'environnement*** *pour éviter de les exposer dans le code source. Cela permet de garder une configuration* ***sécurisée et maintenable****.*

d. Sécurisation et performances

- ***HTTPS** a été activé via les certificats **Let's Encrypt** fournis par Plesk.*
- *Les **en-têtes HTTP de sécurité** (CORS, Content-Security-Policy, etc.) ont été configurés au niveau de l'application Express ou via l'interface serveur.*
- *Mise en place d'un **reverse proxy** (via Plesk ou Apache/Nginx) pour diriger les requêtes vers le serveur Node.js sans exposer le port directement.*
- *Possibilité d'utiliser un **système de logs** intégré à Plesk pour surveiller l'activité de l'application et identifier les erreurs.*

e. Avantages du déploiement via Plesk

Le choix de Plesk présente plusieurs avantages :

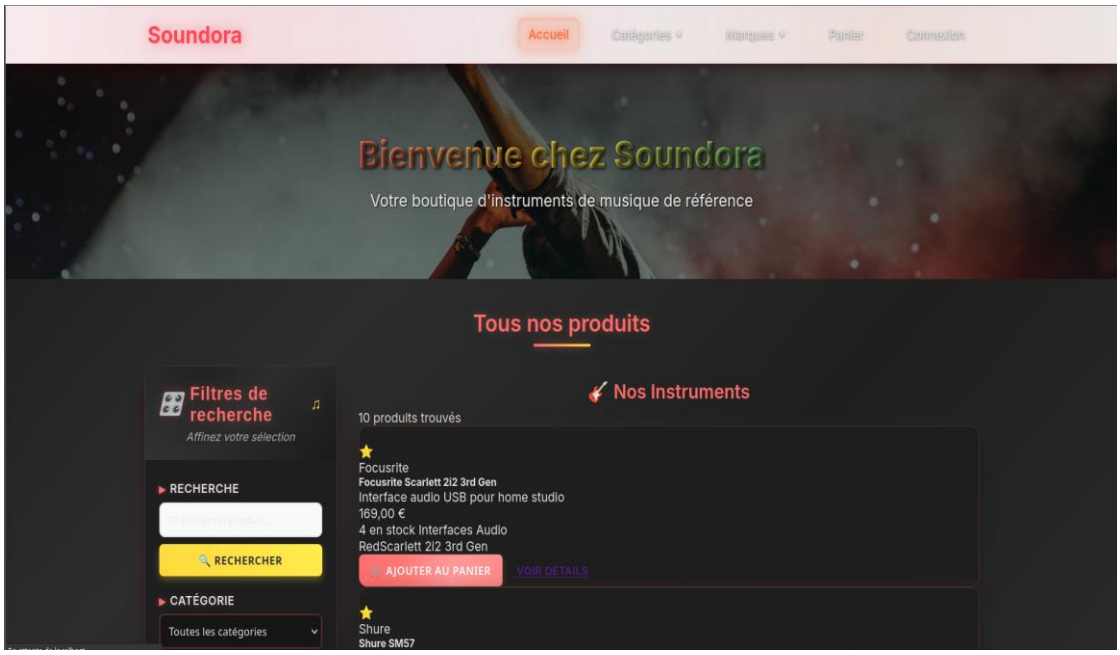
- *Interface graphique intuitive pour gérer le déploiement, sans besoin de ligne de commande avancée.*
- *Support natif de **Node.js**, **certificats SSL**, **gestion de domaines** et **bases de données**.*
- *Compatible avec les besoins d'un projet en fin de formation tout en offrant une **mise en ligne professionnelle**.*

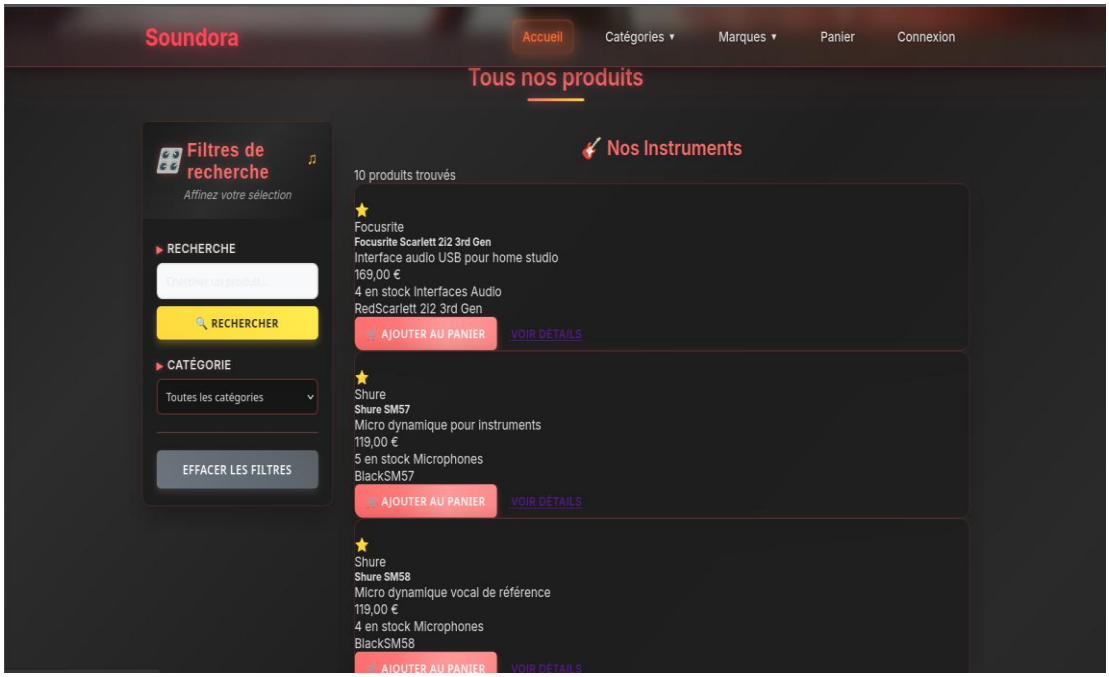
---

*Grâce à Plesk, j'ai pu déployer l'application de manière structurée, en combinant une **interface front Angular** et une **API back Express**, tout en assurant la sécurité et la stabilité de l'environnement de production.*

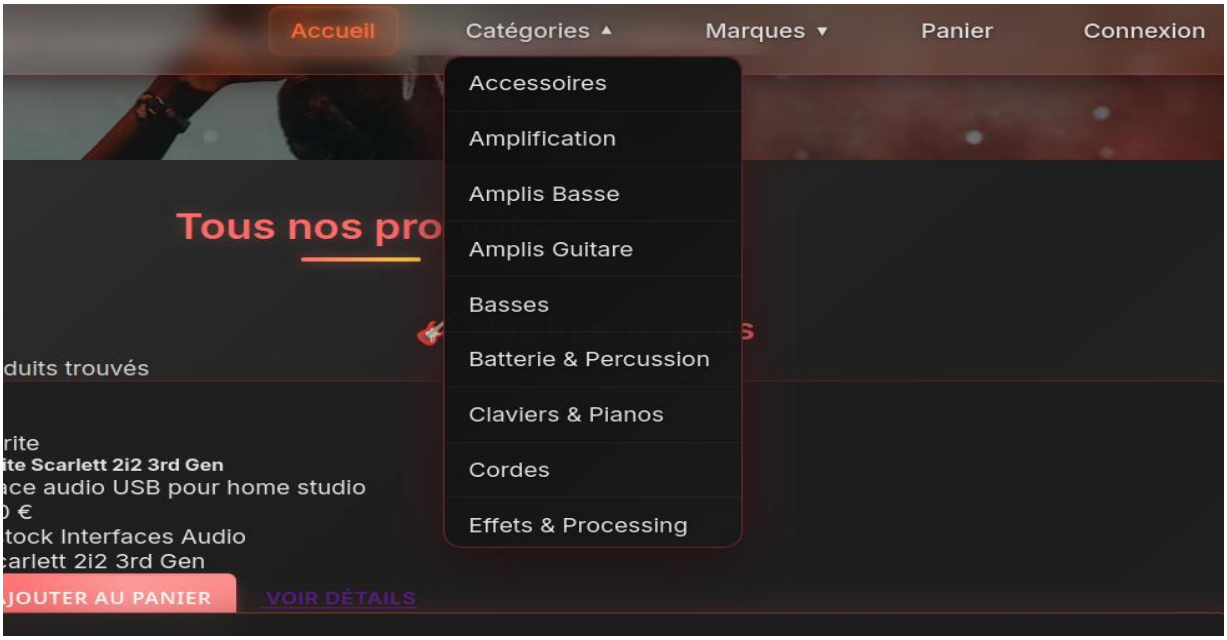
<u>Afin d'avoir un meilleur aperçu visuel du site Soundora, voici différentes captures d'écran illustrant les pages et les fonctionnalités de mon site .</u>

Accueil :

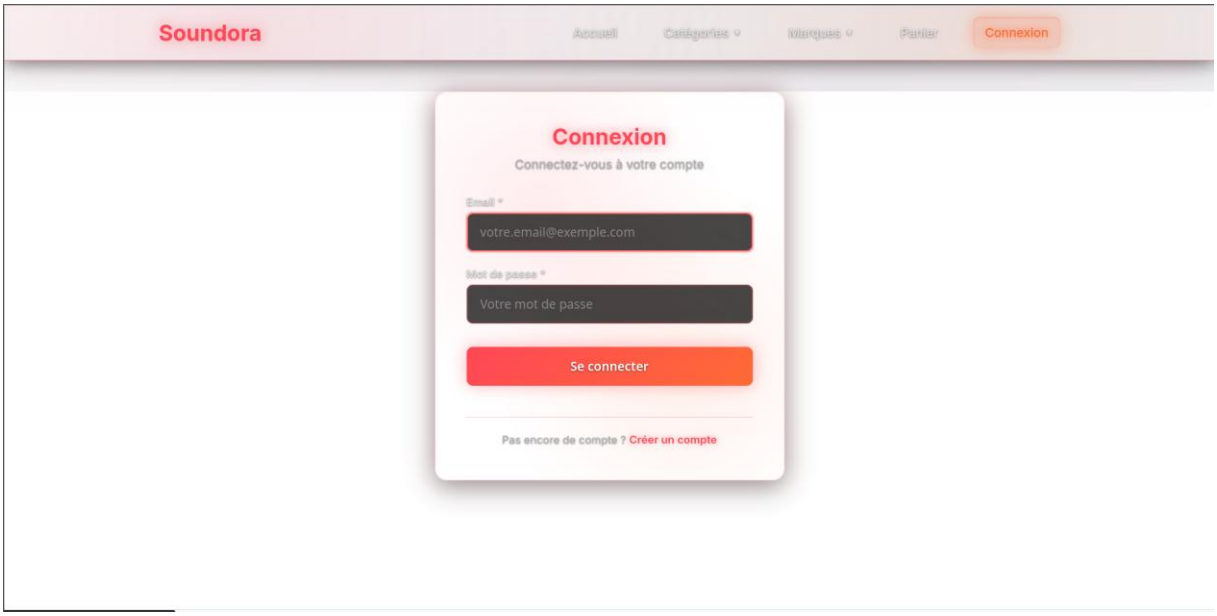




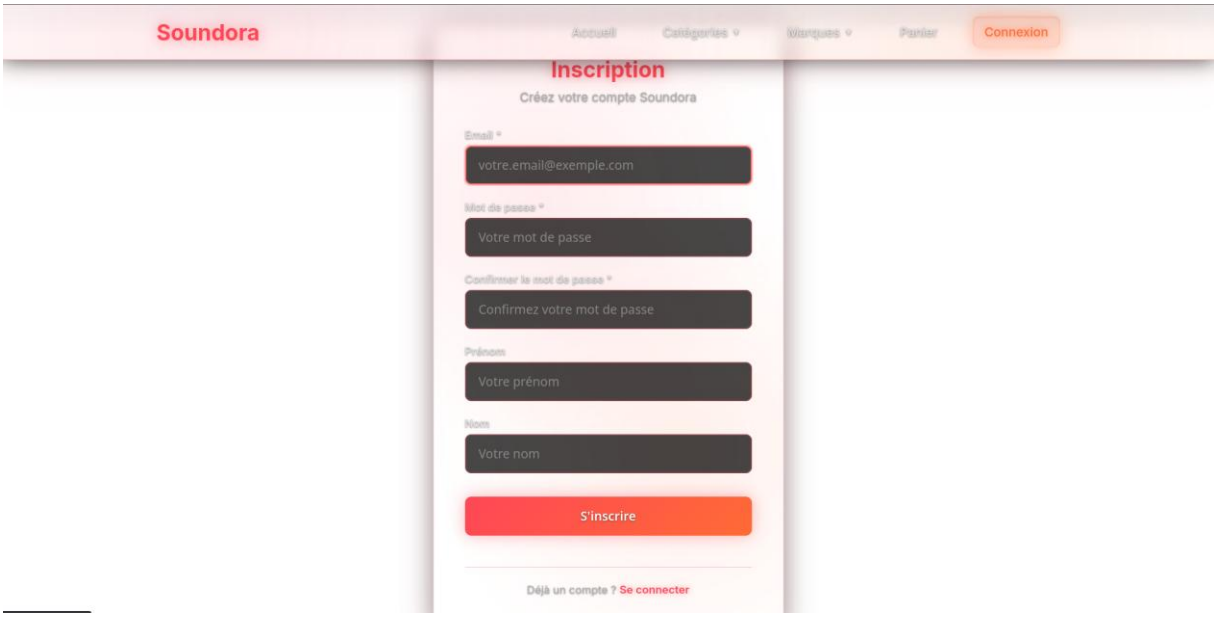
Liste déroulante des produits et des marques disponibles :



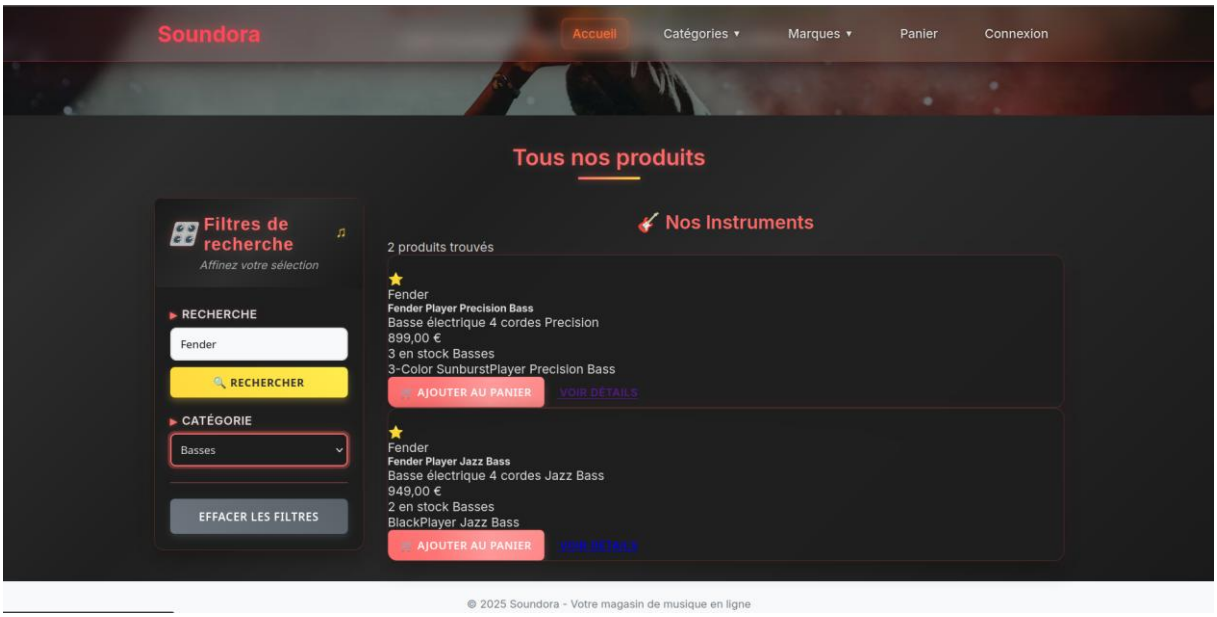
Connexion d'un utilisateur :



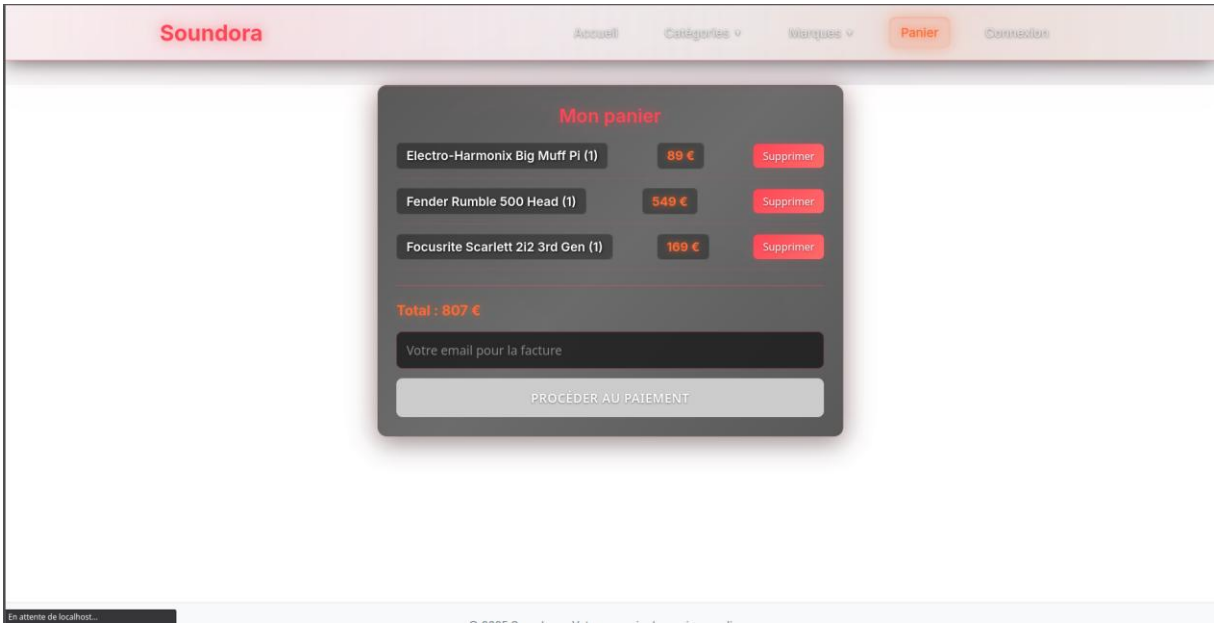
Inscription d'un utilisateur :



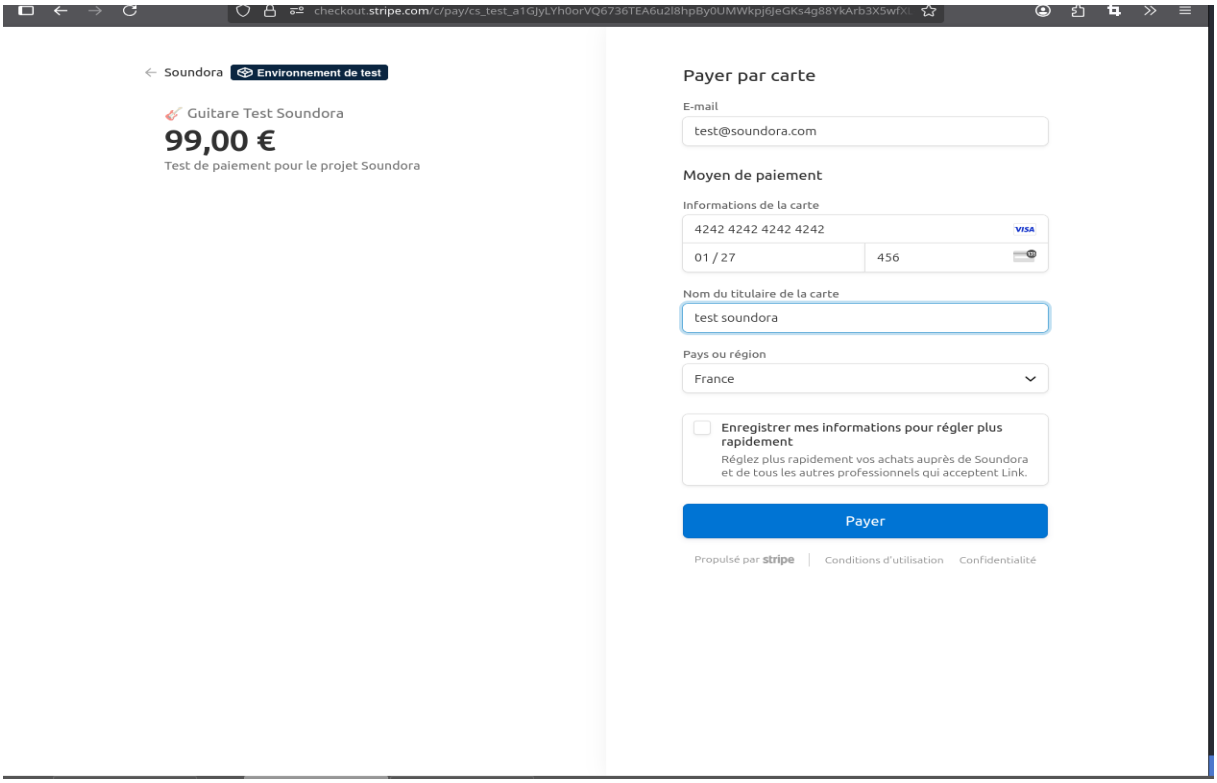
Recherche par filtres :



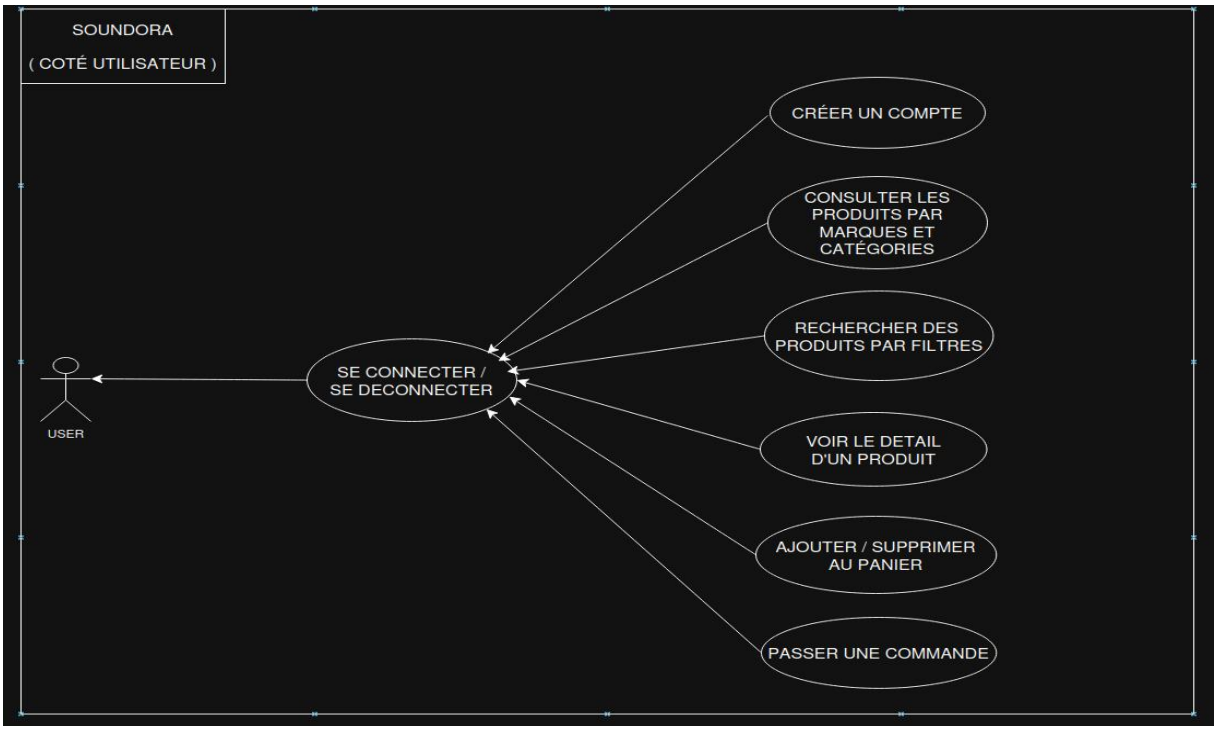
Panier de commande :



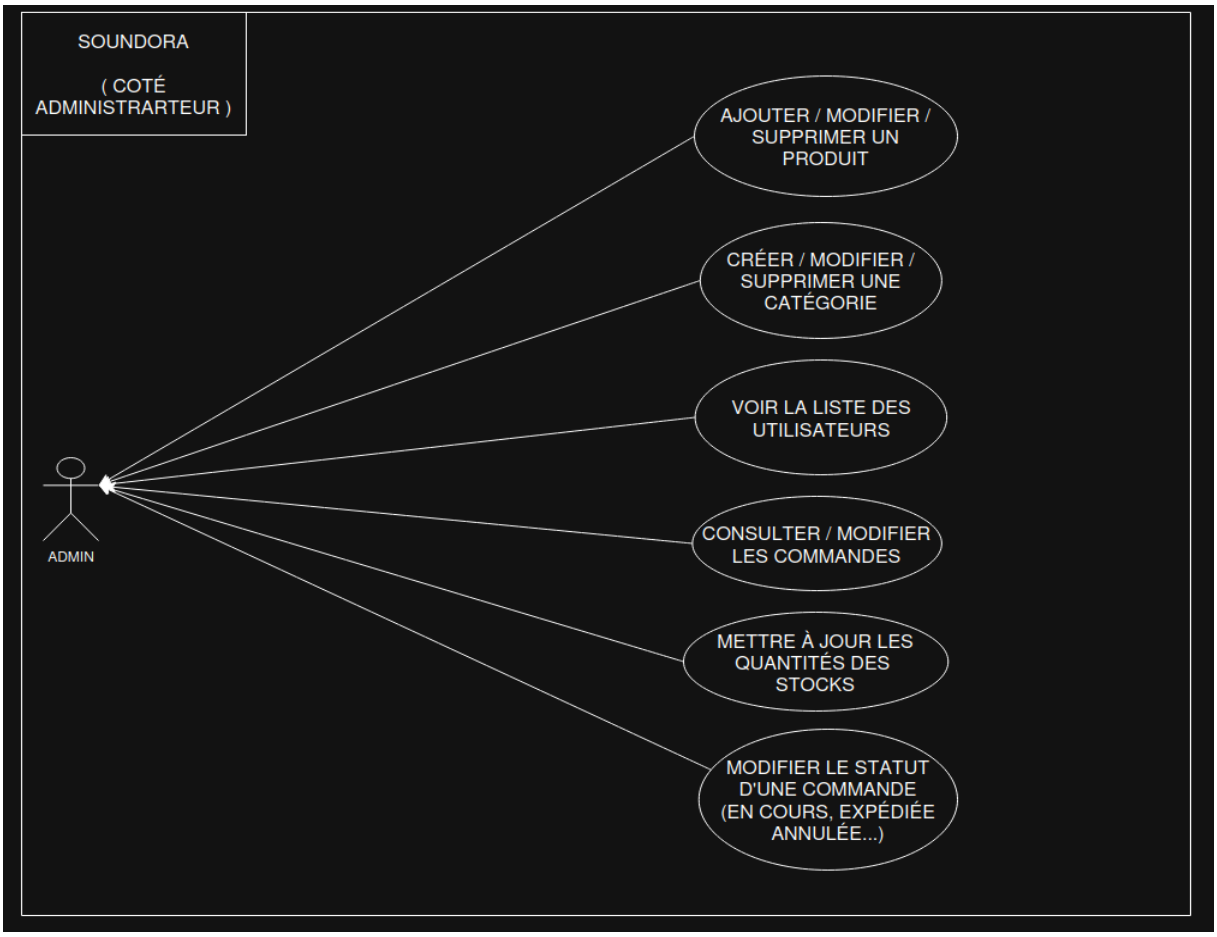
Paiement (par stripe) :

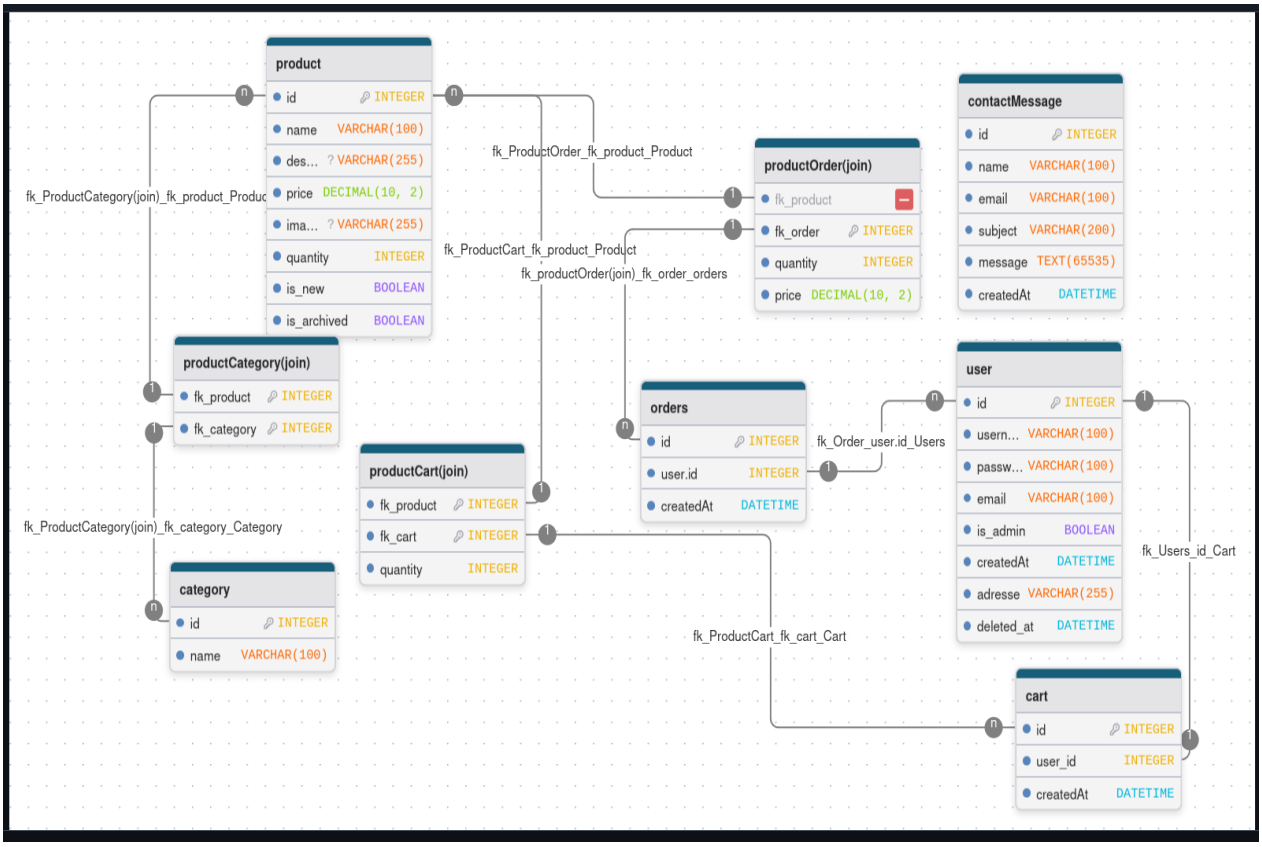


Voici les shémas de cas d’utilisitateurs / administrateurs et le Model Conceptuel de Donnée :



Administateur :





Conclusion :

Conclusion

*Le projet **Soundora**, une boutique en ligne spécialisée dans la vente d'instruments de musique, représente pour moi bien plus qu'un simple exercice de fin d'année : il s'agit de l'aboutissement concret d'un ensemble de compétences techniques et méthodologiques acquises tout au long de ma formation, et surtout d'un premier pas vers une future carrière dans le développement web.*

*La réalisation de ce projet m'a permis d'explorer en profondeur les différents aspects d'un environnement **full-stack moderne**, en m'appuyant sur une stack technique professionnelle et actuelle. Le **frontend**, développé avec **Angular 19**, m'a offert l'opportunité de mieux comprendre les principes de l'architecture component-based, la gestion des routes, ainsi que l'intégration des services pour consommer des API REST. L'utilisation de **TypeScript** m'a également permis de consolider mes compétences en typage statique et en programmation orientée objet, tout en renforçant la maintenabilité du code.*

Côté serveur, le ***backend en Node.js avec Express*** *m'a permis de mettre en place une architecture API RESTful claire et sécurisée, en gérant les différentes interactions entre le client, la base de données et les services externes. Le choix de* ***Supabase*** *pour la gestion de la* ***base de données PostgreSQL*** *ainsi que de l'****authentification via Supabase Auth et JWT*** *m'a permis de travailler avec des technologies cloud-first, à la fois modernes et flexibles. L'intégration de* ***Stripe*** *pour les paiements sécurisés m'a également confronté aux contraintes du monde réel, en particulier sur les questions de sécurité, de gestion des erreurs et d'expérience utilisateur fluide.*

L'étape du ***déploiement****, souvent négligée, a également été un véritable apprentissage. Grâce à* ***Docker*** *et* ***Plesk****, j'ai pu comprendre l'importance de l'environnement de production, de la conteneurisation, et de la gestion des services dans une perspective de déploiement stable et évolutif.*

Ce projet m'a permis de renforcer ma passion pour la programmation et a confirmé mon souhait de ***poursuivre mes études dans ce domaine****, afin de continuer à me perfectionner techniquement, mais aussi à découvrir de nouveaux paradigmes, langages, outils et méthodes de travail. J'ai conscience que le développement web est un domaine en constante évolution, où la veille technologique, la rigueur et la curiosité sont essentielles. C'est donc avec enthousiasme que je souhaite m'investir pleinement dans la suite de mon parcours, en continuant à apprendre, à expérimenter, et à relever de nouveaux défis.*

En somme, ***Soundora*** *m'a permis de prendre conscience du chemin parcouru, mais aussi de tout ce qu'il me reste à apprendre pour devenir un développeur accompli. C'est une étape clé dans ma formation, et un tremplin vers mes futures ambitions professionnelles.*