**Application de Prévision des Commandes en Pâtisserie**

**Plan du cahier des charges source travail Next :**

Memo :

1. Le projet

2. Étude de la concurrence

3. Charte graphique

4. Périmètre du projet (Arborescence, Administration du contenu, Fonctionnalité, Cadre technique, Conception des contenus, Stratégie, Intégrations des contenus, Hébergement, Maintenance et suivi d’évolution)

5. Déroulement du projet

6. Ressources

7. Format attendu pour la réponse du prestataire

**Cahier des Charges : Application de Prévision des Commandes en Pâtisserie**

**Détermination et définition du projet retour entretien avec Valentin Delhon :**

L’idée importante est que chaque responsable de boutique puisse avoir une application pour passer commande directement au laboratoire et au labo pouvoir avoir un retour directement sur le calcul des quantités en production, notamment.

📍 **Client** : Pâtisserie Angelina (Paris, Versailles)  
💡 **Objectif** : Anticiper les commandes en fonction de la fréquentation prévisionnelle, en analysant plusieurs facteurs (historique des ventes, météo, événements, tourisme, économie).

**1. Présentation du Projet**

L’application permettra aux responsables de boutiques de **mieux anticiper leurs commandes** en fonction de prévisions intelligentes, réduisant ainsi les pertes et optimisant la production.

Elle reposera sur un **algorithme prédictif** qui analysera les tendances passées et les événements à venir pour estimer l'affluence et recommander les quantités de produits à préparer.

**2. Fonctionnalités**

**🔹 2.1 Gestion des utilisateurs (Symfony - API Restful)**

👤 **Types d’utilisateurs** :

* **Administrateurs** (gestionnaires des prévisions et de l’application).
* **Responsables de boutique** (consultation des prévisions et ajustement des commandes).
* **Pâtissiers** (visualisation des commandes prévues).

🔑 **Fonctionnalités** :

* Connexion / Inscription sécurisée (JWT, OAuth).
* Gestion des rôles et permissions.

**📊 2.2 Système de Prévisions des Commandes**

📌 **Données prises en compte** :  
✅ Historique des ventes 📉  
✅ Météo 🌤️  
✅ Événements politiques, économiques, culturels, sportifs 📅  
✅ Tourisme (fréquentation touristique, saisonnalité) 🏨

🛠 **Calcul d’impact (%)** :

* Variation de fréquentation estimée en comparant avec les années précédentes.
* Affichage des tendances (hausse/baisse).

💡 **Affichage des prévisions** :

* Recommandations de quantités par produit.
* Graphiques interactifs et tendances.
* Alertes en cas de forte variation prévue.

**🏪 2.3 Gestion des Boutiques**

* 📍 **Gestion des emplacements** (Paris 1er, Versailles, etc.).
* 🏷️ **Capacité maximale de production et stockage** par boutique.
* 🔢 **Données spécifiques à chaque boutique** (ventes passées, fréquentation habituelle).

**🍰 2.4 Gestion des Produits**

* 📜 Catégorisation (viennoiseries, entremets, chocolats, etc.).
* 📈 Suivi des ventes et tendances par produit.
* 📊 Ajustement des prévisions en fonction des stocks disponibles.

**📸 2.5 Module Média**

* Ajout de **photos et vidéos** pour améliorer la communication interne.
* Partage de **documents (PSD, Excel, PDF)** pour optimiser les prévisions.

**3. Technologies utilisées**

**🌐 Back-end : Symfony (API RESTful)**

* Framework PHP robuste et sécurisé.
* Gestion des données avec Doctrine ORM et MySQL.
* Authentification avec JWT/OAuth2.
* Intégration de **scripts Python** pour l’algorithme prédictif.

**💻 Front-end : React.js**

* Interface utilisateur moderne et dynamique.
* Tableau de bord interactif avec **Chart.js / Recharts** pour la visualisation des tendances.

**📊 Machine Learning & Analyse des Données : Python**

* **Pandas, Scikit-learn** pour l’analyse et la modélisation des prévisions.
* **API météo et bases de données touristiques** pour enrichir les prévisions.

**📡 Sources de données externes**

* OpenWeather API (météo).
* Données de l’INSEE / Office du Tourisme de Paris.
* API d’événements (Ville de Paris, TicketMaster pour concerts).

**4. Structure de la Base de Données**

**📌 Entités principales**

1. **User** (id, nom, email, rôle).
2. **Boutique** (id, nom, adresse, capacité\_max).
3. **Vente** (id, date, boutique\_id, produit\_id, quantité).
4. **Produit** (id, nom, catégorie, prix\_vente, coût\_production).
5. **Événement** (id, type, date, impact\_estimé).
6. **Météo** (id, date, température, précipitations).
7. **Tourisme** (id, date, nb\_touristes).
8. **Prévision** (id, date, boutique\_id, fréquentation\_prévue, impact\_estimé).
9. **Media** (id, fichier, type, boutique\_id, utilisateur\_id).

**5. Développement et Déploiement**

**🚀 Étapes de développement**

1. **Conception et structure de la base de données** (Symfony, Doctrine).
2. **Développement de l’API Symfony** (gestion des utilisateurs, ventes, prévisions).
3. **Développement du front-end en React** (tableaux de bord interactifs).
4. **Mise en place de l’algorithme de prévision avec Python**.
5. **Tests et validation avec les responsables**.
6. **Déploiement sur serveur cloud (ex: AWS, OVH, Digital Ocean)**.

**6. Bénéfices attendus**

✅ Meilleure anticipation des commandes ⏳  
✅ Réduction du gaspillage alimentaire ♻️  
✅ Amélioration de la rentabilité 💰  
✅ Gain de temps pour les responsables 🏆

**Développement de l’application**

**Charte graphique :**

theme: {

extend: {

colors: {

primary: {

50: '#f8fafc',

100: '#f1f5f9',

200: '#e2e8f0',

300: '#cbd5e1',

400: '#94a3b8',

500: '#64748b',

600: '#475569',

700: '#334155',

800: '#1e293b',

900: '#0f172a',

},

brown: {

DEFAULT: '#4A2B0F',

light: '#8B633B',

dark: '#2C1907',

},

gold: '#C5A572',

beige: '#E8D5B5',

rose: '#E4C3B9',

},

fontFamily: {

primary: ['Cormorant Garamond', 'serif'],

secondary: ['Work Sans', 'sans-serif'],

},

},

},

Reprise de la charte graphique du site d’Angelina.

**📌 Structure de la Base de Données**

* **Modèle relationnel avec MySQL (Doctrine ORM pour Symfony)**
* **Clés primaires (PK)** et **Clés étrangères (FK)**
* **Indexation** sur les champs de recherche les plus utilisés (dates, boutiques, produits)

**📍 Schéma des Entités**

**1️ User (Utilisateurs)**

| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| id (PK) | INT (AUTO) | Identifiant unique |
| nom | VARCHAR(100) | Nom de l'utilisateur |
| email | VARCHAR(150) | Email (unique) |
| mot\_de\_passe | VARCHAR(255) | Hash du mot de passe |
| rôle | ENUM | admin, responsable, pâtissier |
| créé\_le | DATETIME | Date de création |

🔹 **Relations** :

* Un utilisateur peut gérer plusieurs boutiques (responsable\_id dans Boutique).

**2️ Boutique (Magasins)**

| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| id (PK) | INT (AUTO) | Identifiant unique |
| nom | VARCHAR(100) | Nom de la boutique |
| adresse | TEXT | Adresse complète |
| capacité\_max | INT | Stock maximal possible |
| responsable\_id (FK) | INT | Responsable de la boutique (User) |

🔹 **Relations** :

* Une boutique appartient à des **responsables** (User).

**3️ Vente (Historique des ventes)**

| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| id (PK) | INT (AUTO) | Identifiant unique |
| date | DATE | Date de la vente |
| boutique\_id (FK) | INT | Boutique concernée |
| produit\_id (FK) | INT | Produit vendu |
| quantité | INT | Nombre d’unités vendues |

🔹 **Relations** :

* Une **vente** est liée à une **boutique** et un **produit**.

**4️ Produit (Produits de la pâtisserie)**

| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| id (PK) | INT (AUTO) | Identifiant unique |
| nom | VARCHAR(100) | Nom du produit |
| catégorie | ENUM | viennoiserie, gâteau, chocolat… |
| prix\_vente | DECIMAL(10,2) | Prix de vente en € |
| coût\_prod | DECIMAL(10,2) | Coût de production en € |

🔹 **Relations** :

* Un produit est référencé dans les ventes.

**5️ Prévision (Données de prévision de fréquentation)**

| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| id (PK) | INT (AUTO) | Identifiant unique |
| date | DATE | Date prévue |
| boutique\_id (FK) | INT | Boutique concernée |
| fréquentation\_prev | INT | Nombre estimé de clients |
| impact\_estimé | FLOAT | Variation en % (hausse/baisse) |

🔹 **Relations** :

* Prédictions générées à partir des **ventes passées** et **événements**.

**6️ Événement (Données des événements influençant la fréquentation)**

| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| id (PK) | INT (AUTO) | Identifiant unique |
| type | ENUM | politique, sportif, économique… |
| nom | VARCHAR(255) | Nom de l’événement |
| date | DATE | Date de l’événement |
| impact\_est | FLOAT | Impact prévu sur la fréquentation (%) |

🔹 **Relations** :

* Un événement peut influencer plusieurs prévisions.

**7️ Météo (Facteurs météorologiques influençant la fréquentation)**

| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| id (PK) | INT (AUTO) | Identifiant unique |
| date | DATE | Date d’observation |
| température | FLOAT | Température moyenne (°C) |
| pluie | BOOLEAN | 1 = pluie, 0 = pas de pluie |

🔹 **Relations** :

* La météo influence les prévisions (Prévision).

**8️ Tourisme (Influence du tourisme sur les ventes)**

| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| id (PK) | INT (AUTO) | Identifiant unique |
| date | DATE | Date d’enregistrement |
| nb\_touristes | INT | Nombre de touristes à Paris ce jour |

🔹 **Relations** :

* Impacte la fréquentation (Prévision).

**9️ Media (Partage de fichiers, vidéos et documents pour gestion interne)**

| **Champ** | **Type** | **Description** |
| --- | --- | --- |
| id (PK) | INT (AUTO) | Identifiant unique |
| fichier | VARCHAR(255) | Nom du fichier |
| type | ENUM | image, vidéo, document |
| utilisateur\_id (FK) | INT | Propriétaire du fichier |

🔹 **Relations** :

* Permet le partage d’informations entre responsables.

**📌 Relations Clés**

✔ **Un User (responsable) gère une Boutique**  
✔ **Une Boutique a plusieurs Vente et Prévision**  
✔ **Un Événement influence les Prévision**  
✔ **La Météo et le Tourisme impactent les Prévision**

**📍 Prochaine étape**

📌 **Modélisation en Symfony avec Doctrine**  
➡ Faut-il générer les entités Symfony avec php bin/console make:entity ? 🚀

Recherche du développement d’application discussion première premier prompte :

Je cherche à améliorer et anticiper aux mieux mes commandes de pâtisserie au labo. Existe t il des application qui pourraient aider à anticiper le nombre de client qui pourrait venir dans mes boutique en fonction des événements politiques et économiques de Paris ou je me situ ou en fonction d’autres paramètres auxquelles je n’aurais pas pensé ou que l’on pourrait fixer ou créer. Il faut pouvoir assister au mieux les responsable qui passe commande . Par exemple en fonction de la fréquentation des années précédente. Peut-être existe-t-il des moyens d’avoir des données statistiques de la fréquentation touristique en fonction du type de population par exemple. Je pourrais par exemple proposer une application à mon employeur qui permettrait une amélioration de la prise de commande d’anticiper mieux les commandes pour que nous les Pâtissier nous puissions Puissions ajuster le tir plus facilement derrière et éviter de trop gros écart de commande qui serait trop grosse ou trop petite et donc qui engendrerait trop de perdre. Peut-être un algorithme en informatique pourrait nous aider à mieux se préparer pour éviter les pertes et passer de meilleures commandes en anticipant au mieux le nombre de clients potentiels qui vont venir dans lesjours suivant ou pour les commandes du jour au lendemain. Dis-moi s’il existe des choses comme ça, dis-moi ce que tu en penses et ce que nous pourrions créer.

**ChatGPT a dit :**

Ton idée est excellente et très pertinente pour optimiser la gestion des commandes en pâtisserie. Il existe déjà des solutions qui s’appuient sur l’analyse de données et l’intelligence artificielle pour prévoir la demande en fonction de plusieurs paramètres, mais elles ne sont pas toujours adaptées spécifiquement à un laboratoire de pâtisserie. Voici quelques pistes et solutions possibles :

**1. Applications existantes et données disponibles**

Il existe des logiciels de gestion des stocks et de prévision de la demande, mais peu intègrent directement des facteurs comme les événements politiques, économiques, ou la fréquentation touristique spécifique à Paris. Voici quelques outils intéressants :

* **Too Good To Go & Phenix** : Ces applications luttent contre le gaspillage, mais elles ont aussi des algorithmes qui optimisent les stocks en fonction des ventes passées.
* **Lightspeed, Square ou Openbravo** : Ces logiciels de caisse pour commerces intègrent parfois des modules de prévision des ventes basés sur l’historique.
* **PredictHQ** : Fournit des analyses prédictives basées sur des événements à venir (concerts, grèves, fêtes locales…).
* **Google Trends & INSEE** : Ces sources permettent d’analyser des tendances sur la consommation et le tourisme en fonction de la période de l’année.

**2. Ce qu’on pourrait créer pour un laboratoire de pâtisserie**

Si aucune solution actuelle ne répond exactement à tes besoins, on pourrait concevoir une application ou un algorithme personnalisé. Voici ce qu’il pourrait inclure :

* **Analyse des ventes passées** : Enregistrer et analyser l’historique des ventes en fonction des jours, saisons, événements et conditions économiques.
* **Données externes (tourisme, météo, événements, grèves, fêtes locales)** : En intégrant des API publiques (Open Data Paris, Google Trends, prévisions touristiques), on pourrait prédire la fréquentation dans tes boutiques.
* **Modèle d’intelligence artificielle (Machine Learning)** : Un algorithme pourrait apprendre des ventes passées et ajuster automatiquement les prévisions de commande.
* **Tableau de bord interactif** : Accessible aux responsables pour ajuster les commandes en fonction des recommandations du modèle.

**3. Prochaines étapes pour développer cela**

Si tu veux proposer une solution à ton employeur, voici comment avancer :

1. **Recueillir des données internes** : Analyser plusieurs mois (voire années) de ventes, comparer avec des événements et conditions météo.
2. **Identifier les sources externes utiles** : Trouver les données sur le tourisme, les événements et l’économie locale.
3. **Créer un prototype** : Un petit programme qui prend ces données et fait une première analyse des tendances.
4. **Automatiser la prédiction** : Entraîner un algorithme pour donner des prévisions et suggérer les quantités idéales à commander.

👉 **On pourrait commencer par un outil simple en Python qui analyse tes ventes et les compare à des événements et conditions météo**. Intéressé par cette approche ?

4o

**Vous avez dit :**

Ne vaut-il pas mieux créer un site en symfony et react en front intégrant les script python ? c'est plutôt ma spécialité

**ChatGPT a dit :**

Mémoire mise à jour

Oui, totalement ! Puisque Symfony et React sont ta spécialité, le mieux serait de développer une application web structurée comme suit :

**Architecture de l'application**

* **Back-end (Symfony)** : Gérer l’authentification, stocker les données de ventes, récupérer les données externes (météo, événements, tourisme).
* **Front-end (React)** : Afficher des tableaux de bord interactifs avec les prévisions de commandes et les recommandations.
* **Scripts Python (ML & analyse des données)** : Analyser l’historique des ventes et prévoir la demande avec du Machine Learning.

**Les étapes du développement**

1. **Concevoir la base de données** (Symfony + Doctrine) :
   * Table ventes (produit, quantité, date, jour de la semaine, événement associé).
   * Table événements (nom, date, impact potentiel).
   * Table prévisions (générées par l’algorithme).
2. **Créer un script Python** qui analyse ces données et prédit la demande.
   * Récupérer les ventes passées, les corréler avec les jours de la semaine, la météo, les événements.
   * Utiliser un modèle ML (ex. scikit-learn) pour faire une prévision.
   * Exposer un endpoint API en Symfony pour exécuter et récupérer les prédictions.
3. **Créer une interface React** :
   * Tableau de bord avec visualisation des tendances.
   * Système de recommandations pour aider à la prise de commande.

Cette solution serait complète et pourrait même évoluer vers une application SaaS pour d’autres pâtissiers ! Veux-tu qu’on commence par la structure de la base de données et un premier script Python ? 🚀

4o

**Vous avez dit :**

quelle autre entité pourrions nous utiliser ? vente, événement, prevision, user, media pour rentrer des photos des vidéos que les responsables pourraient se communiquer Et pour rentrer les nouvelles données récolter par exemple sous forme de PSD. Je pense qu'il faut créer un algorithme qui puisse prendre en compte pour son modèle prédictif les évènements politiques de la capitale sportif économique international mondial la météo et peut-être des évènements musicaux et de théâtre notamment sur des grosses programmations qui pourraient attirer du monde. La boutique s'appelle Angelina et est à Paris Premier. Il y a plusieurs boutiques dans Paris dont une notamment à Versailles. Qu'en penses-tu partage-moi de nouvelles idées ou de nouvelles choses qui pourraient être utile

**ChatGPT a dit :**

Mémoire mise à jour

Excellente approche ! Ajouter une entité Media pour stocker des photos, vidéos et documents comme des fichiers PSD est une très bonne idée pour faciliter la communication entre responsables.

**Autres entités utiles pour enrichir l'application :**

1. **Boutique** : Chaque boutique a des spécificités différentes (localisation, affluence).
   * id, nom, adresse, capacité\_max, responsable\_id
   * Relation avec Vente, Prévision, User
2. **Produit** : Pour suivre les ventes par type de pâtisserie.
   * id, nom, catégorie, coût\_production, prix\_vente, stock
   * Relation avec Vente
3. **Tourisme** : Pour intégrer des données sur la fréquentation touristique à Paris et Versailles.
   * id, date, nb\_touristes, source\_données
   * Relation avec Prévision
4. **Météo** : Indispensable pour voir l’impact de la météo sur la fréquentation.
   * id, date, température, précipitations, conditions
   * Relation avec Prévision
5. **Événement** (Politique, Sportif, Culturel) : Déjà prévu, mais on pourrait ajouter une catégorisation plus fine.
   * id, type, nom, date, lieu, impact\_estimé
   * Relation avec Prévision
6. **Prévision** : Stocke les prédictions générées par l’algorithme.
   * id, date, boutique\_id, prévision\_clientèle, taux\_confiance
   * Relation avec Boutique, Vente, Tourisme, Météo

**Idées complémentaires :**

* **Ajouter une API externe pour récupérer des données en temps réel** (météo, événements, affluence touristique).
* **Créer un dashboard interactif avec des graphiques dynamiques (React + Chart.js)**.
* **Développer une gestion des alertes** pour informer les responsables des prévisions critiques (affluence exceptionnelle ou faible).

Avec ces entités et fonctionnalités, l’application pourrait devenir un outil puissant pour Angelina et potentiellement être élargie à d’autres pâtisseries ! 🚀

Veux-tu que je t’aide à structurer la base de données en Symfony pour commencer ?

4o

**Vous avez dit :**

Il faut pouvoir calculer des données statistiques de l'impact estimé en pourcentage par exemple de l'augmentation ou de la diminution de la fréquentation

**ChatGPT a dit :**

Mémoire mise à jour

Bonne idée ! On pourrait ajouter un système de calcul d'impact basé sur plusieurs critères :

**Comment calculer l’impact estimé en pourcentage ?**

L’idée serait de comparer la fréquentation actuelle avec des périodes similaires du passé pour estimer une variation en pourcentage.

**1. Formule de base**

Impact estimeˊ=Freˊquentation preˊvue−Freˊquentation historiqueFreˊquentation historique×100\text{Impact estimé} = \frac{\text{Fréquentation prévue} - \text{Fréquentation historique}}{\text{Fréquentation historique}} \times 100Impact estimeˊ=Freˊquentation historiqueFreˊquentation preˊvue−Freˊquentation historique​×100

Où :

* **Fréquentation prévue** = Résultat de l’algorithme basé sur événements, météo, etc.
* **Fréquentation historique** = Moyenne des fréquentations lors d’événements similaires.

**2. Facteurs influençant l’impact**

On pourrait pondérer les résultats en fonction de plusieurs critères :

* **Impact des événements passés similaires** (ex : Roland-Garros augmente la demande de +20%)
* **Météo** (pluie = moins de clients, beau temps = plus de touristes)
* **Tendances économiques** (pouvoir d’achat, inflation)
* **Jours fériés et vacances scolaires**

**3. Implémentation dans l’application**

* Ajouter une colonne impact\_estimé dans Prévision pour stocker le pourcentage.
* Développer un script Python qui recalcule cet impact automatiquement.
* Afficher les variations sur le dashboard avec des indicateurs visuels (flèches rouges/vertes, graphiques interactifs).

Avec ce système, les responsables auraient une meilleure visibilité pour ajuster leurs commandes. Tu veux qu’on commence par structurer la table Prévision et définir comment récupérer les données historiques ?

Synthèse de la création des entités :

**Entités principales**

1. **User** (id, nom, email, rôle).
2. **Boutique** (id, nom, adresse, capacité\_max).
3. **Vente** (id, date, boutique\_id, produit\_id, quantité).
4. **Produit** (id, nom, catégorie, prix\_vente, coût\_production).
5. **Événement** (id, type, date, impact\_estimé). Données des événements influençant la fréquentation) -> (Politique, Sportif, Culturel) : Déjà prévu, mais on pourrait ajouter une catégorisation plus fine.
6. **Météo** (id, date, température, précipitations).
7. **Tourisme** (id, date, nb\_touristes).
8. **Prévision** (id, date, boutique\_id, fréquentation\_prévue, impact\_estimé). (Données de prévision de fréquentation)
9. **Media** (id, fichier, type, boutique\_id, utilisateur\_id).
10. **Commande** *(une commande générée pour une boutique et envoyée au labo)*
11. **CommandeLigne** *(détail des produits dans une commande)*
12. **Production :** **Rassembler toutes les commandes de toutes les boutiques sur une seule feuille distribuée au labo.**
13. **Notification** *(alertes et messages pour ajustements des commandes et stocks)*

**📌 Ajustement pour Production**

👉 **Objectif : Rassembler toutes les commandes de toutes les boutiques sur une seule feuille distribuée au labo.**

**Nouveaux champs pour Production :**

* id *(Identifiant unique de la production globale du jour)*
* date\_production *(Jour concerné par la production)*
* statut *(en attente, en cours, terminé...)*
* pdf\_commande *(Fichier PDF généré avec toutes les commandes consolidées, imprimable pour distribution)*

**📌 Relations Clés**

✔ **des User (responsables) gère une Boutique**  
✔ **Une Boutique a plusieurs Vente et Prévision**  
✔ **des Événement influence les Prévision**  
✔ **La Météo et le Tourisme impactent les Prévision**

**📌 Schéma des relations**

* **Une Boutique peut passer plusieurs Commandes.**
* **Une Commande contient plusieurs CommandeLigne.**
* **Une CommandeLigne est liée à un seul Produit.**
* **La Production regroupe toutes les Commandes d’un même jour et génère un document unique.**

Faire une navbar avec les onglets pour :

Entités principales 1. User (id, nom, email, rôle). 2. Boutique (id, nom, adresse, capacité\_max). 3. Vente (id, date, boutique\_id, produit\_id, quantité). 4. Produit (id, nom, catégorie, prix\_vente, coût\_production). 5. Événement (id, type, date, impact\_estimé). Données des événements influençant la fréquentation) -> (Politique, Sportif, Culturel) : Déjà prévu, mais on pourrait ajouter une catégorisation plus fine. 6. Météo (id, date, température, précipitations). 7. Tourisme (id, date, nb\_touristes). 8. Prévision (id, date, boutique\_id, fréquentation\_prévue, impact\_estimé). (Données de prévision de fréquentation) 9. Media (id, fichier, type, boutique\_id, utilisateur\_id). 10. Commande (une commande générée pour une boutique et envoyée au labo) 11. CommandeLigne (détail des produits dans une commande) 12. Production : Rassembler toutes les commandes de toutes les boutiques sur une seule feuille distribuée au labo. 13. Notification (alertes et messages pour ajustements des commandes et stocks)

Onglets : boutique – commande – offre – concours meilleur anticipation de commande( ou challenge ou recompense ) – evenements – statistique

**Voici la nouvelle structure de navigation que j'ai créée, organisée autour des fonctionnalités clés de l'application :**

* 1. **Tableau de bord** (/)

Vue d'ensemble des KPIs

Alertes et notifications importantes

Actions rapides

* 1. **Gestion Commandes** (/orders)

Nouvelles Commandes

Historique

Production

* 1. **Prévisions & Analytics** (/analytics)

Prévisions Ventes

Historique Ventes

Performance KPIs

* 1. **Boutiques** (/stores)

Liste des Boutiques

Performance

* 1. **Catalogue** (/products)

Produits

Catégories

Saisonnalité

* 1. **Événements** (/events)

Calendrier

Météo

Tourisme

* 1. **Challenges** (/challenges)

Objectifs

Classement

Récompenses

* 1. **Équipe** (/team)

Membres

Rôles

* 1. **Tableau de bord** (/)
  2. **Gestion Commandes** (/orders)
  3. **Prévisions & Analytics** (/analytics)
  4. **Boutiques** (/stores)
  5. **Catalogue** (/products)
  6. **Événements** (/events)
  7. **Challenges** (/challenges)
  8. **Équipe** (/team)

Reprise de la charte graphique du site Angelina :

Couleurs Principales :

Marron Angelina : **#4A2B0F** - Couleur signature de la marque

Marron clair : **#8B633B** - Pour les hovers et accents

Marron foncé : **#2C1907** - Pour le contraste

Couleurs d'Accent :

Or : **#C5A572** - Pour les éléments décoratifs et accents

Beige : **#E8D5B5** - Pour les fonds clairs et sections secondaires

Rose saumon : **#E4C3B9** - Pour les sections spéciales (comme vu dans l'image de parité)

Typographie :

Titres : 'Cormorant Garamond' - Police serif élégante

Texte : 'Work Sans' - Police sans-serif moderne

Hiérarchie de tailles définie de xs à 4xl

Espacements :

Système d'espacement cohérent de 0.25rem à 3rem

Utilisation de multiples de 4 pour la grille

Ombres et Effets :

Ombres légères pour les cartes et éléments interactifs

Transitions douces pour les interactions

Bordures :

Rayons de bordure variés pour différents éléments

De 0.125rem à 0.5rem

Travaille avec perplexity

Salut je veux savoir dans quelle mesure mon application présente-t-elle des solutions suffisamment innovative pour régler des problèmes d'anticipation de commande dans le restauration notamment pour la pâtisserie dans les boutique et le laboratoire angelina paris. L'idée est d'anticipé au max les commandes pour limiter les pertes de matière première en production dans le labo de pâtisserie . 1. Présentation du Projet

L’application permettra aux responsables de boutiques de mieux anticiper leurs commandes en fonction de prévisions intelligentes, réduisant ainsi les pertes et optimisant la production.

Elle reposera sur un algorithme prédictif qui analysera les tendances passées et les événements à venir pour estimer l'affluence et recommander les quantités de produits à préparer. C'est une étude de benchmark et des autres applications semblables et de leur mode de fonctionnement et propose-moi les axes d'amélioration ou des idées complémentaires.

Votre application présente des solutions innovantes pour résoudre les problèmes d'anticipation des commandes dans la restauration, notamment pour la pâtisserie dans les boutiques et le laboratoire Angelina Paris. Voici une analyse comparative et des suggestions d'amélioration :

**Analyse comparative**

Votre application partage des fonctionnalités similaires avec d'autres solutions existantes :

* Prévisions basées sur l'analyse des données historiques[1](https://www.yokitup.com/application-mobile-de-gestion-de-stock)[4](https://www.inpulse.ai/blog/logiciel-de-gestion-des-stocks-en-boulangerie-reduire-le-gaspillage-et-augmenter-sa-marge)
* Recommandations de quantités à produire[1](https://www.yokitup.com/application-mobile-de-gestion-de-stock)[4](https://www.inpulse.ai/blog/logiciel-de-gestion-des-stocks-en-boulangerie-reduire-le-gaspillage-et-augmenter-sa-marge)
* Prise en compte des facteurs externes (météo, événements)[4](https://www.inpulse.ai/blog/logiciel-de-gestion-des-stocks-en-boulangerie-reduire-le-gaspillage-et-augmenter-sa-marge)

Cependant, elle se distingue par son focus spécifique sur les boutiques de pâtisserie et le laboratoire Angelina Paris.

**Axes d'amélioration et idées complémentaires**

1. **Intégration de la gestion des stocks**
   * Ajoutez un module de suivi des stocks en temps réel[1](https://www.yokitup.com/application-mobile-de-gestion-de-stock)[3](https://lumis-traiteurs.fr/m%C3%A9tier/comment-gerer-les-commandes-pour-ma-patisserie/)
   * Automatisez les commandes fournisseurs basées sur les prévisions[4](https://www.inpulse.ai/blog/logiciel-de-gestion-des-stocks-en-boulangerie-reduire-le-gaspillage-et-augmenter-sa-marge)
2. **Optimisation de la production**
   * Intégrez un planificateur de production basé sur les prévisions[1](https://www.yokitup.com/application-mobile-de-gestion-de-stock)[5](https://www.boutique-petit.com/blogs/actu/comment-optimiser-le-processus-de-production-pour-gagner-du-temps-en-patisserie)
   * Proposez des rotations optimales pour maximiser l'utilisation des équipements[5](https://www.boutique-petit.com/blogs/actu/comment-optimiser-le-processus-de-production-pour-gagner-du-temps-en-patisserie)
3. **Personnalisation avancée**
   * Adaptez l'algorithme aux spécificités d'Angelina Paris (clientèle, produits phares)
   * Intégrez des données sur les tendances culinaires et les préférences saisonnières
4. **Interface utilisateur intuitive**
   * Développez une interface tactile facile à utiliser en laboratoire[3](https://lumis-traiteurs.fr/m%C3%A9tier/comment-gerer-les-commandes-pour-ma-patisserie/)
   * Créez des tableaux de bord visuels pour une prise de décision rapide
5. **Gestion des invendus**
   * Intégrez un module de gestion des invendus (promotions, dons)[4](https://www.inpulse.ai/blog/logiciel-de-gestion-des-stocks-en-boulangerie-reduire-le-gaspillage-et-augmenter-sa-marge)
   * Proposez des stratégies de revalorisation des produits non vendus
6. **Analyse prédictive avancée**
   * Utilisez l'apprentissage automatique pour affiner continuellement les prévisions
   * Intégrez des données externes (réseaux sociaux, événements locaux) pour améliorer la précision
7. **Collaboration inter-boutiques**
   * Permettez le partage de stocks entre boutiques pour optimiser la distribution
8. **Formation du personnel**
   * Intégrez un module de formation pour optimiser l'utilisation de l'application[2](https://science-journal.org/comment-optimiser-le-laboratoire-de-patisserie/)
9. **Suivi de la qualité**
   * Ajoutez des fonctionnalités de contrôle qualité pour maintenir l'excellence d'Angelina Paris
10. **Intégration e-commerce**
    * Liez l'application à une plateforme de commande en ligne pour une gestion unifiée[3](https://lumis-traiteurs.fr/m%C3%A9tier/comment-gerer-les-commandes-pour-ma-patisserie/)

En implémentant ces améliorations, votre application pourrait devenir une solution complète et innovante pour l'optimisation de la production pâtissière, spécifiquement adaptée aux besoins d'Angelina Paris.

**Citations:**

1. <https://www.yokitup.com/application-mobile-de-gestion-de-stock>
2. <https://science-journal.org/comment-optimiser-le-laboratoire-de-patisserie/>
3. <https://lumis-traiteurs.fr/m%C3%A9tier/comment-gerer-les-commandes-pour-ma-patisserie/>
4. <https://www.inpulse.ai/blog/logiciel-de-gestion-des-stocks-en-boulangerie-reduire-le-gaspillage-et-augmenter-sa-marge>
5. <https://www.boutique-petit.com/blogs/actu/comment-optimiser-le-processus-de-production-pour-gagner-du-temps-en-patisserie>
6. <https://www.restoconnection.fr/quelles-fonctionnalites-pour-la-solution-de-commande-en-ligne-de-votre-restaurant/>
7. [https://prestachef.fr](https://prestachef.fr/)
8. <https://www.nelinkia.com/blog/normes/conception-laboratoire-patisserie.html>

Groq

To develop the application for Angelina Paris, follow this organized and elegant implementation plan, broken down into key areas with actionable steps:

**1. Data Collection and Preprocessing**

**Data Sources Integration**: Integrate external APIs for social media, tourism, and weather. Use web scraping cautiously for competitor data.

**Data Cleaning**: Implement data cleaning processes to handle missing values and inconsistencies.

**Feature Engineering**: Create new features by combining existing data (e.g., weather + tourism for foot traffic prediction).

**2. Predictive Algorithm Development**

**Model Selection**: Start with a simple model (e.g., linear regression) and progress to LSTM for time series forecasting.

**Training and Validation**: Use historical data for training, validate with test data, and retrain periodically.

**Integration**: Develop a separate ML service with an API for predictions, integrating into the Symfony backend.

**3. Boutique-Specific Features**

**Customizable Dashboard**: Implement a drag-and-drop interface in React for users to customize their dashboard.

**Sales Analysis Tools**: Add charts and graphs to compare actual vs. predicted sales.

**Inventory Management**: Link predictions to inventory updates, using alerts for stock adjustments.

**4. Feedback Loop Implementation**

**Real-Time Data**: Use WebSockets for real-time updates and sensors for foot traffic.

**Model Updates**: Set up cron jobs for periodic model retraining.

**User Feedback**: Add a feedback form for users to rate prediction accuracy.

**5. Enhanced User Experience**

**Role-Based Access**: Configure Symfony security for role-based access, using annotations for access control.

**Mobile Optimization**: Ensure the React frontend is responsive and consider PWA features.

**Support Tools**: Include a help section, tutorials, and a ticket system for user support.

**6. Advanced Features Implementation**

**Scenario Simulation**: Allow users to input scenarios and see predictive outcomes.

**Impact Analysis**: Develop dependency analysis tools to show how factors affect sales.

**Batch Processing**: Use asynchronous tasks with queues for large data processing.

**7. Security and Compliance**

**Data Encryption**: Implement HTTPS and database encryption.

**Compliance**: Ensure GDPR compliance with data anonymization and consent management.

**Regular Audits**: Use tools like SonarQube for security audits.

**8. Performance Optimization**

**Database Optimization**: Index tables and partition large datasets.

**Caching**: Implement Redis for caching frequently accessed data.

**Load Balancing**: Use Nginx or AWS ELB for distributing traffic.

**9. Collaborative Tools**

**Feedback Module**: Add a form for users to submit prediction feedback.

**Shared Calendar**: Integrate a calendar tool for event coordination.

**Communication Tools**: Implement real-time messaging using Socket.io.

**10. Sustainability Features**

**Dashboard**: Display sustainability metrics using charts.

**Waste Reduction Tools**: Implement alerts for overstocked items.

**Reporting Tools**: Allow users to generate and export sustainability reports.

**11. Testing and Validation**

**Unit and Integration Tests**: Use PHPUnit for backend and Jest for frontend.

**User Testing**: Conduct beta tests with selected boutiques.

**A/B Testing**: Use tools like Google Optimize for feature testing.

**12. Deployment and Scaling**

**Cloud Setup**: Deploy on AWS or GCP, using managed services for databases.

**Auto-Scaling**: Configure auto-scaling groups for serverless architecture.

**Monitoring**: Use Prometheus and Grafana for monitoring and alerting.

**13. Documentation and Training**

**Documentation**: Write user manuals and API docs using Swagger.

**Training**: Organize sessions with hands-on exercises and materials.

**Support**: Implement a ticket system for user assistance.

**14. Continuous Improvement**

**Feedback Loop**: Create a system for users to submit feedback easily.

**Updates**: Schedule regular updates with a versioning system using GitFlow.

**15. Risk Management**

**Risk Assessment**: Identify and mitigate risks like data breaches or model inaccuracies.

**Contingency Planning**: Develop backup plans for data recovery and system outages.

**Change Management**: Plan training and support to ease the transition.

By following this structured approach, the application will effectively support Angelina Paris with predictive analytics, user-friendly tools, and sustainable practices.

Voici le plan détaillé pour guider les prochaines étapes de développement de votre application, en tenant compte de toutes les fonctionnalités et implémentations nécessaires. Le plan est divisé en plusieurs sections clés pour une meilleure organisation.

**1. Collecte et Prétraitement des Données**

**a. Intégration des Données Externes**

**Données Météo** :

Intégrer des API comme OpenWeatherMap pour obtenir des données météorologiques en temps réel.

Stocker les données dans la base de données (table Météo).

**Données Touristiques** :

Utiliser des API comme TourEiffel ou des données ouvertes pour obtenir le flux de touristes.

Stocker les données dans la table Tourisme.

**Données Sociales** :

Intégrer des API de réseaux sociaux (Twitter, Instagram) pour analyser les tendances liées aux pâtisseries.

Créer une nouvelle table pour stocker ces données.

**b. Nettoyage et Normalisation des Données**

**Nettoyage des Données** :

Implémenter des scripts pour supprimer les doublons et corriger les erreurs dans les données.

**Normalisation** :

Normaliser les données pour s'assurer qu'elles sont conformes aux formats attendus par les modèles.

**c. Gestion des Données Manquantes**

**Remplissage des Données Manquantes** :

Implémenter des algorithmes de remplissage (imputation) pour les données manquantes.

**Validation des Données** :

Vérifier la qualité des données avant leur utilisation dans les modèles.

**2. Développement de l'Algorithme Prédicatif**

**a. Sélection et Entraînement du Modèle**

**Sélection des Modèles** :

Tester différents modèles (LSTM, ARIMA, Random Forest) pour prédire les ventes.

Utiliser des bibliothèques comme Scikit-learn ou TensorFlow.

**Entraînement** :

Utiliser les données historiques des ventes pour entraîner le modèle.

Intégrer les données externes (météo, tourisme, etc.) comme des fonctionnalités supplémentaires.

**b. Validation et Optimisation**

**Validation Croisée** :

Évaluer les performances du modèle avec des données de validation.

Ajuster les hyperparamètres pour améliorer l'accuracy.

**Intégration en Temps Réel** :

Développer une API pour permettre une prédiction en temps réel.

Intégrer l'API dans l'application via Symfony.

**c. Mise à Jour du Modèle**

**Réentrainement Automatique** :

Configurer un processus de réentrainement du modèle périodiquement (par exemple, toutes les semaines).

Utiliser des outils comme Airflow pour automatiser cette étape.

**Monitoring des Performances** :

Implémenter un système de suivi des performances du modèle (erreur moyenne, écart-type, etc.).

**3. Fonctionnalités Spécifiques aux Boutiques**

**a. Tableau de Bord Personalisé**

**Visualisation des Données** :

Intégrer des bibliothèques comme Chart.js ou D3.js pour créer des graphiques interactifs.

Afficher les ventes par produit, les prévisions, et les impacts des événements externes.

**Filtres et Rapports** :

Ajouter des filtres par date, par boutique, et par catégorie de produit.

Générer des rapports PDF ou Excel pour une analyse plus approfondie.

**b. Gestion des Ventes**

**Suivi des Ventes** :

Afficher les ventes en temps réel sur le tableau de bord.

Créer des alertes pour les ruptures de stock ou les surstocks.

**Analyse des Performances** :

Comparer les ventes réelles avec les prévisions pour ajuster les stratégies.

**c. Gestion des Produits Périssables**

**Prédiction des Ventes** :

Intégrer des modèles spécifiques pour les produits périssables (ex. macarons, croissants).

Prendre en compte les facteurs saisonniers et les événements locaux.

**Optimisation des Commandes** :

Développer un outil pour recommander les quantités à commander en fonction des prévisions.

**4. Intégration des Données des Événements**

**a. Gestion des Événements**

**Saisie des Événements** :

Créer une interface utilisateur pour ajouter et modifier des événements (salons, fêtes, grèves, etc.).

Stocker les données dans la table Événement.

**Impact des Événements** :

Développer un module pour estimer l'impact des événements sur les ventes.

Intégrer ces estimations dans les modèles prédicifs.

**b. Analyse des Données des Événements**

**Analyse des Données Passées** :

Créer des rapports pour analyser l'impact des événements passés sur les ventes.

Afficher ces analyses sur le tableau de bord.

**5. Interface Utilisateur**

**a. Tableau de Bord Principal**

**Visualisation des Données Clés** :

Afficher les ventes totales, les prévisions, et les performances par produit.

Utiliser des cartes (KPI) pour une visualisation rapide.

**Filtres et Recherche** :

Ajouter des filtres par date, par boutique, et par catégorie de produit.

Permettre la recherche par nom de produit ou d'événement.

**b. Gestion des Commandes**

**Saisie des Commandes** :

Créer une interface pour saisir les commandes des fournisseurs.

Intégrer les prévisions pour recommander les quantités à commander.

**Suivi des Commandes** :

Afficher l'état des commandes (en attente, en cours, livrées).

Envoyer des notifications par e-mail ou SMS pour les retards.

**c. Gestion des Produits**

**Catalogue des Produits** :

Créer une interface pour gérer le catalogue des produits (ajout, modification, suppression).

Intégrer des images et des descriptions détaillées.

**Gestion des Catégories** :

Créer une table pour les catégories de produits.

Permettre de filtrer les produits par catégorie.

**d. Gestion des Événements**

**Calendrier des Événements** :

Créer un calendrier interactif pour visualiser les événements programmés.

Permettre d'ajouter, de modifier ou de supprimer des événements.

**Impact des Événements** :

Créer une interface pour estimer l'impact des événements sur les ventes.

Intégrer ces estimations dans les prévisions.

**e. Tableau de Bord pour les Responsables de Boutique**

**Visualisation des Données** :

Afficher les ventes, les prévisions, et les performances par produit.

Permettre de filtrer par date, par boutique, et par catégorie.

**Alertes et Notifications** :

Envoyer des alertes par e-mail ou SMS en cas de ruptures de stock ou de surstocks.

Créer une section pour les alertes dans l'interface utilisateur.

**6. Intégration des Données Touristiques**

**a. Collecte des Données Touristiques**

**Intégration des Données** :

Intégrer des API comme TourEiffel ou des données ouvertes pour obtenir le flux de touristes.

Stocker les données dans la table Tourisme.

**Analyse des Données** :

Créer des graphiques pour visualiser l'impact du tourisme sur les ventes.

Intégrer ces données dans les modèles prédicifs.

**b. Analyse des Tendances**

**Analyse des Tendances** :

Créer des rapports pour analyser les tendances des ventes en fonction des données touristiques.

Afficher ces rapports sur le tableau de bord.

**7. Intégration des Données Sociales**

**a. Analyse des Réseaux Sociaux**

**Intégration des Données Sociales** :

Intégrer des API comme Twitter API ou Instagram Graph API pour analyser les tendances des réseaux sociaux.

Stocker les données dans une nouvelle table RéseauSocial.

**Analyse des Tendances** :

Créer des graphiques pour visualiser les tendances des réseaux sociaux.

Intégrer ces données dans les modèles prédicifs.

**b. Analyse des Données des Réservations**

**Intégration des Données des Réservations** :

Créer une table Réservation pour stocker les données des réservations.

Intégrer ces données dans les modèles prédicifs.

**Analyse des Données** :

Créer des rapports pour analyser l'impact des réservations sur les ventes.

Afficher ces rapports sur le tableau de bord.

**8. Intégration des Données des Réservations**

**a. Saisie des Données des Réservations**

**Interface pour Saisir les Réservations** :

Créer une interface utilisateur pour saisir les réservations.

Intégrer ces données dans la table Réservation.

**Validation des Données** :

Valider les données saisies pour s'assurer de leur qualité.

Envoyer des notifications en cas d'erreurs.

**b. Analyse des Données des Réservations**

**Analyse des Données** :

Créer des rapports pour analyser l'impact des réservations sur les ventes.

Afficher ces rapports sur le tableau de bord.

**Prévisions Basées sur les Réservations** :

Développer un module pour ajuster les prévisions en fonction des réservations.

**9. Intégration des Données des Réservations**

**a. Saisie des Données des Réservations**

**Interface pour Saisir les Réservations** :

Créer une interface utilisateur pour saisir les réservations.

Intégrer ces données dans la table Réservation.

**Validation des Données** :

Valider les données saisies pour s'assurer de leur qualité.

Envoyer des notifications en cas d'erreurs.

**b. Analyse des Données des Réservations**

**Analyse des Données** :

Créer des rapports pour analyser l'impact des réservations sur les ventes.

Afficher ces rapports sur le tableau de bord.

**Prévisions Basées sur les Réservations** :

Développer un module pour ajuster les prévisions en fonction des réservations.

**10. Intégration des Données des Réservations**

**a. Saisie des Données des Réservations**

**Interface pour Saisir les Réservations** :

Créer une interface utilisateur pour saisir les réservations.

Intégrer ces données dans la table Réservation.

**Validation des Données** :

Valider les données saisies pour s'assurer de leur qualité.

Envoyer des notifications en cas d'erreurs.

**b. Analyse des Données des Réservations**

**Analyse des Données** :

Créer des rapports pour analyser l'impact des réservations sur les ventes.

Afficher ces rapports sur le tableau de bord.

**Prévisions Basées sur les Réservations** :

Développer un module pour ajuster les prévisions en fonction des réservations.

**11. Intégration des Données des Réservations**

**a. Saisie des Données des Réservations**

**Interface pour Saisir les Réservations** :

Créer une interface utilisateur pour saisir les réservations.

Intégrer ces données dans la table Réservation.

**Validation des Données** :

Valider les données saisies pour s'assurer de leur qualité.

Envoyer des notifications en cas d'erreurs.

**b. Analyse des Données des Réservations**

**Analyse des Données** :

Créer des rapports pour analyser l'impact des réservations sur les ventes.

Afficher ces rapports sur le tableau de bord.

**Prévisions Basées sur les Réservations** :

Développer un module pour ajuster les prévisions en fonction des réservations.

**12. Intégration des Données des Réservations**

**a. Saisie des Données des Réservations**

**Interface pour Saisir les Réservations** :

Créer une interface utilisateur pour saisir les réservations.

Intégrer ces données dans la table Réservation.

**Validation des Données** :

Valider les données saisies pour s'assurer de leur qualité.

Envoyer des notifications en cas d'erreurs.

**b. Analyse des Données des Réservations**

**Analyse des Données** :

Créer des rapports pour analyser l'impact des réservations sur les ventes.

Afficher ces rapports sur le tableau de bord.

**Prévisions Basées sur les Réservations** :

Développer un module pour ajuster les prévisions en fonction des réservations.

**13. Intégration des Données des Réservations**

**a. Saisie des Données des Réservations**

**Interface pour Saisir les Réservations** :

Créer une interface utilisateur pour saisir les réservations.

Intégrer ces données dans la table Réservation.

**Validation des Données** :

Valider les données saisies pour s'assurer de leur qualité.

Envoyer des notifications en cas d'erreurs.

**b. Analyse des Données des Réservations**

**Analyse des Données** :

Créer des rapports pour analyser l'impact des réservations sur les ventes.

Afficher ces rapports sur le tableau de bord.

**Prévisions Basées sur les Réservations** :

Développer un module pour ajuster les prévisions en fonction des réservations.

**14. Intégration des Données des Réservations**

**a. Saisie des Données des Réservations**

**Interface pour Saisir les Réservations** :

Créer une interface utilisateur pour saisir les réservations.

Intégrer ces données dans la table Réservation.

**Validation des Données** :

Valider les données saisies pour s'assurer de leur qualité.

Envoyer des notifications en cas d'erreurs.

**b. Analyse des Données des Réservations**

**Analyse des Données** :

Créer des rapports pour analyser l'impact des réservations sur les ventes.

Afficher ces rapports sur le tableau de bord.

**Prévisions Basées sur les Réservations** :

Développer un module pour ajuster les prévisions en fonction des réservations.

**15. Intégration des Données des Réservations**

**a. Saisie des Données des Réservations**

**Interface pour Saisir les Réservations** :

Créer une interface utilisateur pour saisir les réservations.

Intégrer ces données dans la table Réservation.

**Validation des Données** :

Valider les données saisies pour s'assurer de leur qualité.

Envoyer des notifications en cas d'erreurs.

**b. Analyse des Données des Réservations**

\*\*Analy

### DeepSeek

### Plan de mise en œuvre structuré pour l’application d’Angelina Paris

#### 1. **Collecte et Prétraitement des Données**

* **Intégration des sources de données** : Connecter des API externes pour les réseaux sociaux, le tourisme et la météo. Utiliser le web scraping avec précaution pour les données des concurrents.
* **Nettoyage des données** : Mettre en place des processus de nettoyage des données pour gérer les valeurs manquantes et les incohérences.
* **Ingénierie des caractéristiques** : Créer de nouvelles variables en combinant des données existantes (ex. : météo + tourisme pour la prévision du trafic piétonnier).

#### 2. **Développement de l’Algorithme Prédictif**

* **Choix du modèle** : Commencer avec un modèle simple (ex. : régression linéaire), puis évoluer vers LSTM pour la prévision des séries temporelles.
* **Entraînement et validation** : Utiliser les données historiques pour l’entraînement, valider avec des données de test et effectuer des réentraînements périodiques.
* **Intégration** : Développer un service ML séparé avec une API de prédiction intégrée au backend Symfony.

#### 3. **Fonctionnalités Spécifiques aux Boutiques**

* **Tableau de bord personnalisable** : Implémenter une interface en React avec un système de glisser-déposer pour la personnalisation.
* **Outils d’analyse des ventes** : Ajouter des graphiques et des comparaisons entre les ventes réelles et prévues.
* **Gestion des stocks** : Lier les prévisions aux mises à jour des stocks et générer des alertes pour ajuster les niveaux.

#### 4. **Mise en Place d’une Boucle de Rétroaction**

* **Données en temps réel** : Utiliser WebSockets pour les mises à jour en temps réel et des capteurs pour mesurer le trafic piétonnier.
* **Mises à jour du modèle** : Programmer des tâches cron pour le réentraînement périodique du modèle.
* **Retour utilisateur** : Ajouter un formulaire pour que les utilisateurs évaluent la précision des prévisions.

#### 5. **Amélioration de l’Expérience Utilisateur**

* **Accès basé sur les rôles** : Configurer la sécurité de Symfony avec des annotations pour gérer les droits d’accès.
* **Optimisation mobile** : Rendre l’interface React responsive et envisager des fonctionnalités PWA.
* **Outils d’assistance** : Inclure une section d’aide, des tutoriels et un système de tickets pour l’assistance utilisateur.

#### 6. **Implémentation de Fonctionnalités Avancées**

* **Simulation de scénarios** : Permettre aux utilisateurs de saisir des scénarios hypothétiques et d’observer les prévisions associées.
* **Analyse d’impact** : Développer des outils d’analyse pour visualiser l’influence des facteurs externes sur les ventes.
* **Traitement en lot** : Utiliser des tâches asynchrones avec des files d’attente pour le traitement des grandes quantités de données.

#### 7. **Sécurité et Conformité**

* **Chiffrement des données** : Mettre en place HTTPS et le chiffrement des bases de données.
* **Conformité RGPD** : Assurer l’anonymisation des données et la gestion des consentements.
* **Audits réguliers** : Utiliser des outils comme SonarQube pour analyser la sécurité du code.

#### 8. **Optimisation des Performances**

* **Optimisation des bases de données** : Indexer les tables et partitionner les gros ensembles de données.
* **Mise en cache** : Implémenter Redis pour stocker les données fréquemment consultées.
* **Répartition de charge** : Utiliser Nginx ou AWS ELB pour distribuer le trafic.

#### 9. **Outils Collaboratifs**

* **Module de feedback** : Ajouter un formulaire pour recueillir les retours des utilisateurs sur les prévisions.
* **Calendrier partagé** : Intégrer un calendrier pour la coordination des événements.
* **Outils de communication** : Mettre en place une messagerie en temps réel avec Socket.io.

#### 10. **Fonctionnalités de Durabilité**

* **Tableau de bord écologique** : Afficher des indicateurs de durabilité sous forme de graphiques.
* **Réduction des déchets** : Générer des alertes pour éviter les stocks excédentaires.
* **Outils de reporting** : Permettre la génération et l’exportation de rapports de durabilité.

#### 11. **Tests et Validation**

* **Tests unitaires et d’intégration** : Utiliser PHPUnit pour le backend et Jest pour le frontend.
* **Tests utilisateurs** : Effectuer des tests bêta avec des boutiques sélectionnées.
* **Tests A/B** : Employer des outils comme Google Optimize pour évaluer les nouvelles fonctionnalités.

#### 12. **Déploiement et Mise à l’Échelle**

* **Infrastructure cloud** : Déployer sur AWS ou GCP avec des services gérés pour les bases de données.
* **Auto-scaling** : Configurer des groupes d’auto-scaling pour une architecture sans serveur.
* **Surveillance** : Utiliser Prometheus et Grafana pour le monitoring et les alertes.

#### 13. **Documentation et Formation**

* **Documentation** : Rédiger des manuels utilisateur et des API docs avec Swagger.
* **Formation** : Organiser des sessions avec exercices pratiques et supports pédagogiques.
* **Support** : Mettre en place un système de tickets pour l’assistance utilisateur.

#### 14. **Amélioration Continue**

* **Boucle de retour** : Mettre en place un système permettant aux utilisateurs de soumettre des retours facilement.
* **Mises à jour régulières** : Planifier des versions avec un système de gestion de versions (GitFlow).

#### 15. **Gestion des Risques**

* **Évaluation des risques** : Identifier et atténuer les risques liés aux failles de sécurité ou aux erreurs de modèle.
* **Plan de contingence** : Prévoir des sauvegardes pour la récupération des données et la gestion des pannes.
* **Gestion du changement** : Organiser des formations et un accompagnement pour faciliter la transition.

Grâce à cette approche méthodique, l’application offrira à Angelina Paris des outils analytiques avancés, une interface intuitive et des pratiques durables. 🚀

Version ChatGPT : (que je préfère)

# **Plan de Développement de l’Application**

## **Phase 1 : Finalisation des Entités et Relations**

Avant d’implémenter les fonctionnalités, assure-toi que tes entités et relations sont bien définies. Voici quelques ajustements et ajouts à considérer :

1. **Améliorer les relations entre entités**
   * **User** : Ajouter une relation ManyToOne avec Boutique (un utilisateur appartient à une boutique sauf pour les administrateurs). Ok ManyToMany -- ok !
   * **Vente** : Ajouter une relation ManyToOne avec User (pour savoir quel employé a validé la vente).
   * **Produit** : Ajouter une colonne stock\_disponible pour anticiper les ruptures.
   * **Prévision** : Ajouter une relation OneToMany avec Vente (permettra de comparer la prévision avec la réalité).
2. **Créer de nouvelles entités pour enrichir l’application**
   * **Stock (id, boutique\_id, produit\_id, quantité\_disponible, seuil\_alerte, dernière\_mise\_à\_jour)**  
     → Permet de suivre en temps réel les niveaux de stock.
   * **Commande (id, boutique\_id, fournisseur\_id, date\_commande, statut, produits\_commandés)**  
     → Facilite la gestion des réapprovisionnements.
   * **Fournisseur (id, nom, contact, délai\_livraison\_moyen, produits\_fournis)**  
     → Permet d’intégrer la gestion des fournisseurs directement.

### **Implémentation Symfony :**

* Mettre à jour les entités avec Doctrine (php bin/console make:entity)
* Générer et appliquer les migrations (php bin/console doctrine:migrations:migrate)
* Mettre en place les relations (ManyToOne, OneToMany, etc.) avec @ORM\ManyToOne et @ORM\OneToMany

## **Phase 2 : Back-end - API Symfony**

L’API REST va gérer les interactions entre le back-end Symfony et le front-end React.

### **Routes à implémenter :**

1. **Gestion des utilisateurs (/api/users)**
   * Création, mise à jour, suppression des comptes.
   * Authentification (JWT via lexik/jwt-authentication-bundle).
   * Gestion des rôles (ROLE\_ADMIN, ROLE\_MANAGER, ROLE\_EMPLOYEE).
2. **Gestion des boutiques (/api/stores)**
   * Ajout / modification des boutiques.
   * Association avec un responsable (User).
3. **Gestion des produits (/api/products)**
   * Création / modification des produits avec leur coût de production et prix de vente.
4. **Gestion des ventes (/api/sales)**
   * Enregistrement des ventes avec date, boutique, produit et quantité vendue.
5. **Gestion des stocks (/api/stocks)**
   * Vérification du stock disponible et alertes en cas de stock bas.
6. **Prévisions et analytics (/api/forecasts)**
   * Enregistrement et récupération des prévisions de fréquentation et de ventes.
7. **Données externes (/api/events, /api/weather, /api/tourism)**
   * Récupération des événements, météo et flux touristiques pour affiner les prévisions.

### **Implémentation Symfony :**

* Créer des Controller pour chaque entité avec php bin/console make:controller
* Implémenter les services pour gérer la logique métier (src/Service)
* Utiliser API Platform pour générer automatiquement des endpoints REST
* Gérer les autorisations avec Voter et Security

## **Phase 3 : Front-end - ReactJS**

L’interface utilisateur doit être fluide, ergonomique et réactive.

### **Pages à implémenter :**

1. **Tableau de bord (/)**
   * Synthèse des ventes du jour, fréquentation prévue, alertes sur les stocks.
   * Graphiques dynamiques avec recharts.
2. **Gestion des commandes (/orders)**
   * Liste des commandes en cours et archivées.
   * Possibilité de valider / modifier une commande.
3. **Prévisions & Analytics (/analytics)**
   * Courbes comparant prévisions vs ventes réelles.
   * Filtres par période et boutique.
4. **Boutiques (/stores)**
   * Liste et détails des boutiques.
   * Gestion des responsables.
5. **Catalogue des produits (/products)**
   * Liste et édition des produits disponibles.
   * Affichage des stocks restants.
6. **Challenges internes (/challenges)**
   * Gamification : récompenser les meilleures ventes, la gestion des pertes.
7. **Équipe (/team)**
   * Liste des employés par boutique.
   * Gestion des accès et permissions.

### **Implémentation ReactJS :**

* Utiliser React Router pour la navigation.
* Gérer l’état global avec Redux ou Context API.
* Intégrer Material UI pour un design moderne.
* Consommer l’API avec axios.

## **Phase 4 : Algorithme de Prédiction**

L’algorithme prédictif est le cœur du projet.

### **Modélisation initiale (ML Basique)**

1. **Collecte des données**
   * Export des historiques de ventes, météo, événements.
2. **Pré-traitement des données**
   * Nettoyage et normalisation des valeurs.
3. **Premier modèle : Régression Linéaire**
   * Tester un modèle simple basé sur scikit-learn.
   * Vérifier l’exactitude des prédictions.
4. **Amélioration avec des modèles avancés (LSTM, XGBoost)**
   * Intégrer des données externes en temps réel.
   * Ajuster les prévisions quotidiennement.
5. **Déploiement en API Flask/FastAPI**
   * Développer un microservice Python pour l’IA.
   * Consommer l’API depuis Symfony.

## **Phase 5 : Sécurité et Optimisation**

1. **Sécurisation des données**
   * Chiffrement des données sensibles (bcrypt pour les mots de passe).
   * Utilisation de JWT pour l’authentification.
2. **Optimisation des performances**
   * Mise en cache avec Redis.
   * Indexation des bases de données.
3. **Scalabilité**
   * Déploiement sur AWS/GCP.
   * Mise en place de Docker pour la conteneurisation.

## **Phase 6 : Déploiement et Test**

1. **Tests unitaires et d’intégration (PHPUnit, Jest)**
2. **Déploiement en production (CI/CD, GitHub Actions)**
3. **Monitoring avec Prometheus et Grafana**
4. **Formation des utilisateurs finaux (sessions en boutique)**

### **Conclusion**

Ce plan détaillé va te permettre d’avancer méthodiquement tout en assurant une base solide pour ton application. Commence par stabiliser le back-end avant d’optimiser le front-end et d’intégrer l’intelligence artificielle.

Exemple d’application réussie utilisant un agent IA en l’occurance Bolt :

<https://www.youtube.com/watch?v=imTEwYAnE1M&ab_channel=ABA>

->mapply

<https://m-apply.fr/>

Utilisation de l’ IA claude sonnet 3.7

<https://www.youtube.com/watch?v=qMCPPAkrYDw&ab_channel=ExplorIA>