Conception fonctionnelle

1. Présentation du contexte

L'application mobile vise à répondre aux besoins spécifiques des personnes sourdes en facilitant leur participation à des événements communautaires, sportifs ou professionnels. L'objectif est d'offrir un outil accessible et simple permettant de visualiser les événements sur une carte interactive et d'utiliser des filtres pour rechercher les activités pertinentes. L'accent est mis sur l'accessibilité, la simplicité et une interface claire adaptée à la communauté sourde.

---

2. Présentation du besoin

Problématique : Les personnes sourdes rencontrent souvent des difficultés pour accéder à des informations adaptées concernant les événements qui leur sont spécifiquement dédiés.
Solution proposée : Développer une application qui centralise les événements pour cette communauté, tout en simplifiant leur recherche et leur affichage. L'application ne nécessite pas de création de compte pour garantir une adoption rapide et intuitive.

Exigences fonctionnelles principales :

1. Une carte interactive affichant les événements géolocalisés en France.
2. Un système de filtres pour affiner les recherches (par catégorie, date ou lieu).
3. Une base de données mise à jour uniquement par les développeurs (pas de back-office).
4. Une expérience utilisateur fluide et rapide, sans inscription.

---

3. Documents de référence

- **Base de données** : Firebase Realtime Database contenant les données des événements.
- **Cahier des charges** : Instructions du client définissant les délais (3 semaines) et le budget (1 000 €).
- **Google Maps SDK Documentation** : Pour intégrer les cartes interactives.

---

4. Arborescence / Architecture du site web

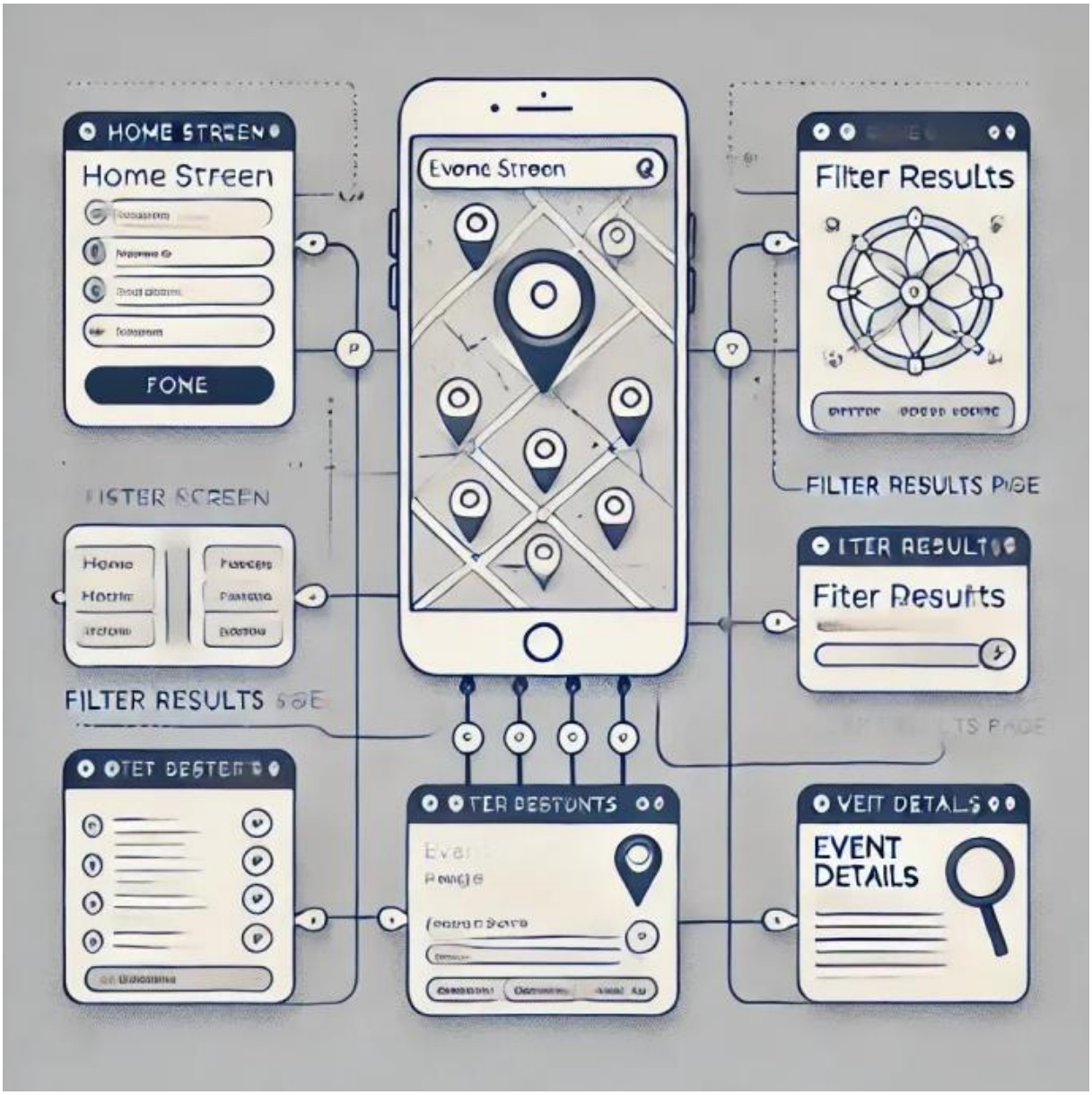
L'application suit une structure simple pour maximiser l'expérience utilisateur.

Arborescence fonctionnelle :

1. **Écran principal (Accueil)**
    - Carte interactive avec les événements géolocalisés.
    - Barre de recherche pour filtrer les événements.
2. **Page des résultats filtrés**
    - Liste des événements correspondant aux critères sélectionnés.

3. **Page d’information sur un événement**
    - Affichage des détails d’un événement : nom, catégorie, lieu, date, description.

5. Maquettes



6. Les fonctionnalités qui permettent tel processus

1. **Carte interactive :**
    - Permet de visualiser les événements géolocalisés sous forme de pins.
    - Les pins sont cliquables pour afficher des détails.
2. **Filtres de recherche :**
    - Recherche par catégorie : Travail, Sport, Communauté.
    - Recherche par localisation (ville).
    - Recherche par date.
3. **Page d'information détaillée :**
    - Présentation complète d'un événement sélectionné.
4. **Synchronisation en temps réel :**
    - Mise à jour dynamique des données via Firebase.

---

7. Glossaire (Vocabulaire technique)

- **API** : Interface de programmation permettant à des applications de communiquer entre elles (exemple : Google Maps API).
- **Firebase** : Plateforme backend-as-a-service utilisée pour gérer les données de l'application.
- **Pins** : Marqueurs placés sur une carte interactive pour indiquer la position des événements.
- **Flutter** : Framework de développement d'applications multiplateformes.
- **UML** : Unified Modeling Language, utilisé pour représenter des systèmes logiciels sous forme de diagrammes.
- **JSON** : Format d'échange de données léger utilisé pour structurer les données dans Firebase.

Conception Technique

1. Choix technique pour l'application en général

- **Framework principal** : **Flutter**
    - Justification : Flutter permet un développement multiplateforme rapide et efficace. Il garantit une compatibilité optimale avec Android et permet une éventuelle extension vers iOS.
- **Langage de programmation** : **Dart**
    - Justification : Intégré avec Flutter, il offre une syntaxe moderne et un bon support pour les animations et widgets personnalisés.
- **Base de données** : **Firebase Realtime Database**
    - Justification : Gestion des données en temps réel, simplicité d'intégration avec Flutter, et coût adapté au budget.

---

2. Composants complémentaires

- **Carte interactive** :
    - **Google Maps SDK pour Flutter**
        - Fournit des cartes précises avec la possibilité d'ajouter des "pins" interactifs pour afficher les événements.
        - Alternative en cas de problème : **OpenStreetMap** (libre et gratuit).
- **Filtres de recherche** :
    - Utilisation de la bibliothèque **Provider** pour la gestion de l'état et la synchronisation des filtres avec la base de données.
- **Design et interface utilisateur** :
    - **Material Design Widgets** pour garantir un design moderne, ergonomique et cohérent.

---

3. Compatibilité

- **Navigateurs compatibles** :
    - Non applicable, l'application est mobile native (pas de version web prévue dans ce projet).
- **Types d'appareils compatibles** :
    - **Smartphones** (Android 7.0 et versions supérieures).
    - Éventuellement compatible avec tablettes, mais non optimisé spécifiquement.

---

4. Services tiers utilisés

- **Google Maps API** : Pour la carte interactive et les géolocalisations.
- **Firebase** : Pour l'hébergement de la base de données, l'authentification éventuelle future, et les analytics simplifiés.
- **Flutter Community Plugins** : Pour les composants standard comme la gestion des connexions réseau et l'accès à des bibliothèques courantes.

---

5. Spécifications techniques du serveur

L'application utilise Firebase, qui est un service backend-as-a-service (BaaS). Par conséquent, aucun serveur d'hébergement traditionnel n'est nécessaire.

Spécifications Firebase :

- **Hébergement** : Google Cloud Platform.
- **Capacité** : Scalabilité automatique en fonction de l'utilisation.
- **Type de données** : Stockage en JSON, optimisé pour les interactions en temps réel.
- **Sécurité** : Configuration des règles de sécurité pour restreindre les accès.

---

6. Méthode utilisée pour répondre aux exigences

- **Affichage des événements sur la carte** :
    - Les événements sont récupérés depuis Firebase sous forme de coordonnées géographiques (latitude et longitude).
    - Google Maps SDK affiche les "pins" correspondants sur la carte.
- **Filtres** :
    - Les données des filtres (catégorie, localisation, date) sont synchronisées avec Firebase via des requêtes dynamiques.
    - Les résultats filtrés sont affichés sous forme de liste sous la carte.
- **Ajout des événements** :
    - Les événements sont ajoutés manuellement par les développeurs dans la base Firebase, via un outil d'administration externe (par exemple, Firebase Console).