PHARMADOC

DOCUMENTATION TECHNIQUE

Table des matières

Structure de la PWA

```
├── css/
│   ├── bulma.css       # Librairie Graphique
│   └── style.css       # Propriétés CSS additionnel
├── favicon/            # Icônes
├── img/                # Images utilisés
├── js/
│   └── pwa.js          # Fichier JS principal
├── lib/
│   └── leaflet         # Librairie externe (map)
├── favoris.html        # Page des favoris
├── index.html          # Page d'accueil / Installation
├── map.html            # Page de la carte
├── recherche.html      # Page de recherche
└── service_worker.js   # Gestion du cache et mode offline
```

1. PWA Installation & Service Worker

main()

Fonction principale d'initialisation qui détecte si l'app est en mode PWA ou web.

Paramètres :Aucun

Retour : Aucun

Logique :

- Vérifie le mode d'affichage avec `window.matchMedia("(display-mode: standalone)")`
- Si PWA : enregistre le Service Worker et affiche un message de remerciement
- Si Web : ajoute les listeners pour l'installation PWA **Appels :**
- `registerServiceWorker()`
- `window.addEventListener("beforeinstallprompt", onBeforeInstallPrompt)`
- `window.addEventListener("appinstalled", onAppInstalled)`

onBeforeInstallPrompt(event)

Gère l'événement avant la demande d'installation PWA.

Paramètres : `event` (Événement beforeinstallprompt)

Retour : Aucun

Logique :

- Empêche le comportement par défaut
- Active le bouton d'installation
- Sauvegarde l'événement en variable globale pour utilisation ultérieure

installPwa()

Lance le processus d'installation de la PWA.

Paramètres : Aucun

Retour : Aucun (async)

Logique :

- Vérifie que l'événement `beforeinstallprompt` est disponible
- Affiche la boîte de dialogue d'installation
- Gère la réponse (accepté/rejeté)
- Désactive le bouton après utilisation

onAppInstalled()

Appelée après l'installation réussie de la PWA.

Paramètres : Aucun

Retour : Aucun

Logique : Enregistre le Service Worker après installation.

registerServiceWorker()

Enregistre le Service Worker pour la gestion du cache hors ligne.

Retour : Aucun (async)

Logique :

- Vérifie la disponibilité de l'API Service Worker
- Enregistre le fichier `./service_worker.js`
- Ajoute un listener pour détecter les mises à jour
- Gère les erreurs d'enregistrement

onUpdateFound(event)

Détecte qu'une nouvelle version du Service Worker est disponible.

Paramètres : `event` (Événement de l'enregistrement)

Retour : Aucun

Logique : Récupère le nouveau Service Worker et ajoute un listener de changement d'état.

onStateChange(event)

Gère les changements d'état du Service Worker.

Paramètres : `event` (Événement de changement d'état)

Retour : Aucun

Logique : Si l'état est "installed" et qu'il y a un contrôleur actif, une mise à jour est disponible. Active le bouton de rechargement.

reloadPwa()

Recharge l'application PWA pour appliquer les mises à jour.

Logique : Appelle `window.location.reload()`

2. Gestion de la carte et géolocalisation

initializeMap()

Initialise la carte Leaflet et demande l'accès à la géolocalisation.

Dépendances : Leaflet (L), Élément DOM `MAP_ELEMENT`

Logique :

1. Vérifie la présence de l'élément `MAP_ELEMENT`
2. Crée une instance Leaflet centrée sur [45.5, 5.7] avec zoom 13
3. Ajoute les tuiles OpenStreetMap
4. Demande la géolocalisation de l'utilisateur (Timeout: 10000ms, HighAccuracy: false)
5. Gère les erreurs

onGeolocationSuccess(position)

Traite la position réussie de l'utilisateur.

Paramètres : `position` (Objet Position contenant les coordonnées)

Variables clés : `LAT`, `LON`

Logique :

1. Extrait la latitude et longitude
2. Centre la carte sur la position (zoom 15)
3. Affiche un marqueur bleu circulaire à la position (Rayon: 10px, Bleu #3273dc)
4. Vérifie le cache des pharmacies
5. Si cache valide : affiche les pharmacies du cache. Sinon : recherche via l'API.

onGeolocationError(error)

Gère les erreurs de géolocalisation (Accès refusé, Indisponible, Timeout).

Logique : Affiche un message d'erreur spécifique, masque l'overlay de chargement et déclenche une alerte.

toggleLoadingOverlay(show)

Affiche ou cache l'écran de chargement.

Paramètres : `show` (Booléen)

Logique : Ajoute ou enlève la classe `is-hidden` sur l'élément `loadingOverlay.`

3. Recherche de médicaments

handleSearch(event)

Gère la soumission du formulaire de recherche.

API Endpoint : `https://medicaments-api.giygas.dev/medicament/{nom}`

Logique :

1. Empêche le rechargement de la page
2. Récupère le texte saisi dans `SEARCH_INPUT`
3. Si non vide : Affiche "Chargement...", construit l'URL et appelle `get()`.

get(url, fonctionSuivante)

Fonction générique d'appel API avec gestion d'erreur.

Retour : Aucun (async)

Logique :

1. Effectue un fetch à l'URL
2. Vérifie `RESPONSE.ok` et parse le JSON
3. Appelle le callback avec les données
4. En cas d'erreur : affiche une notification Bulma "danger" et log l'erreur.

displayResults(results)

Affiche les résultats de la recherche sous forme de cartes.

Paramètres : `results` (Tableau de médicaments)

Structure des cartes : Image (16:9), Titre, CIS, Statut, Bouton favoris.

Logique :

- Vide le container. Si vide : affiche message.
- Génère les cartes HTML.
- Pour chaque item : vérifie si favori, détermine l'image et l'état du bouton (Ajouter/Retirer).

4. Gestion des pharmacies

findPharmacies(lat, lon)

Recherche les pharmacies à proximité (rayon 5km) via l'API Overpass.

Requête Overpass utilisée :

```
[out:json][timeout:25];
(
node["amenity"="pharmacy"](bounds);way["amenity"="pharmacy"](bounds);
relation["amenity"="pharmacy"](bounds);
);
out center;
```

Logique : Calcule les limites géographiques (Box) et lance l'appel API via `get()`.

addPharmacyMarkers(data)

Ajoute les marqueurs des pharmacies sur la carte Leaflet.

Logique :

- Masque le chargement.
- Pour chaque pharmacie : extrait les infos (Nom, Tél, Web, Horaires), gère les géométries complexes, construit une popup HTML et ajoute le marqueur (icône `pin_pharmadoc.svg`).
- Sauvegarde les données en cache.

getPharmaciesFromCache(latitude, longitude)

Récupère les pharmacies depuis le cache localStorage.

Critères de validité :

1. Expiration : Moins de 30 minutes (TTL).
2. Proximité : Différence de position < 0.01°.

Retour : Tableau de pharmacies ou null.

savePharmaciesToCache(latitude, longitude, pharmacies)

Sauvegarde les pharmacies en cache avec un timestamp.

Clé LocalStorage : `pharmacies_cache`

Format : `{ lat, lon, timestamp, pharmacies: [] }`

5. Gestion des favoris

saveFavoriteMedication(cisCode, medicationData)

Sauvegarde un médicament dans localStorage.

Clé : `meds_{cisCode}`

Logique **:** Sérialise l'objet en JSON et sauvegarde.

getFavoriteMedication(cisCode)

Récupère un médicament favori du localStorage. *

Retour : Objet médicament ou null.

removeFavoriteMedication(cisCode)

Description : Retire un médicament des favoris (suppression de la clé).

toggleFavorite(cisCode, medicationData)

Bascule l'état favori d'un médicament.

Logique :

- Si existe : retire et vibre [200ms].
- Si n'existe pas : ajoute et vibre [100ms].
- Affiche une alerte et recharge la page de détails.

toggleFavoriteOnList(cisCode)

Ajoute/retire un médicament depuis la liste de résultats.

renderFavorisList()

Affiche la liste complète des médicaments en favoris.

Logique :

1. Itère sur localStorage pour trouver les clés `meds_`.
2. Génère les cartes (Image, Nom, CIS, Statut, Bouton suppression).
3. Si vide : affiche un message informatif.
4. Attache les event listeners.
5.

removeFavoriteAndRefresh(cisCode)

Retire un médicament, vibre l'appareil et rafraîchit l'affichage de la liste.

6. Affichage HTML

showMedicationDetails(cisCode)

Affiche la page de détails d'un médicament.

Logique :

1. Masque les résultats.
2. Cherche d'abord dans les favoris (LocalStorage).
3. Si non trouvé ou en ligne : appelle l'API (`/medicament/id/{cisCode}`).

renderDetailPage(medicament)

Construit et affiche la page détaillée.

Description :

- **En-tête :** Image, nom, CIS, labo, tags (Statut, Surveillance).
- **Infos :** Voies d'administration, AMM.
- **Composition :** Liste des substances.
- **Tableau des prix :** Prix TTC et taux de remboursement. **Interactions :** Boutons Retour, Favoris, Copier CIS.

getMedicamentImage(medicamentName)

Description : Retourne l'image appropriée selon la forme pharmaceutique (Comprimé, Sirop, Crème, Injection, Poudre, Collyre).

7. Utilitaires & Gestion Événements

copyCISToClipboard(cisCode)

Copie le code CIS dans le presse-papier via l'API Clipboard.

vibrateDevice(pattern)

Fait vibrer l'appareil (ex: `[100]`). Gère les erreurs silencieusement si non supporté.

Gestionnaires d'événements (Handlers)

- `handleSearch` : Submit recherche.
- `handleResultatsClick` : Délégation de clic pour les résultats.
- `handleFavBtnClick` : Bouton favori (page détail).
- `handleGoBackClick` / `goBackToSearch` : Navigation retour.
- `handleCopyCisClick` : Bouton copier.

Architecture et Flux de Données

Flux de recherche : Saisie Utilisateur → `handleSearch()` → `get()` (API) → `displayResults()`

Flux de localisation : Chargement Page → `initializeMap()` → `onGeolocationSuccess()` → Vérif Cache → Si non valide : `findPharmacies()` (API Overpass) → `addPharmacyMarkers()`

Stockage Local (LocalStorage) :

- Favoris : Clé `meds_{cisCode}` (JSON complet).
- Cache Pharmacies : Clé `pharmacies_cache` (JSON avec timestamp).

API Externes :

- Médicaments : `https://medicaments-api.giygas.dev`
- Cartographie : `https://overpass-api.de/api/interpreter`
- Géolocalisation & Vibration : APIs natives du navigateur.