---

1 🦊 Modèles. 2 fichiers

1 - user.js

Objet littéral central :

`auth` 🟫 – gère l'accès aux données utilisateurs et la session dans `localStorage`.

Nom	Type	Statut	Description
`generateSalt()`	🟣 Méthode	✅	Génère un salt aléatoire pour le hash des mots de passe
`hashPassword(password, salt, iterations)`	🟣 Méthode	✅	Hash SHA-256 + salt + itérations
`register(name, email, password)`	🟣 Méthode	🔴	Crée un utilisateur et l'ajoute dans localStorage ⚠ **Problème :** validation (existence email, trim) → devrait être dans **controller register.js**
`login(email, password)`	🟣 Méthode	🔴	Vérifie credentials, met à jour `loggedEmail` ⚠ **Problème :** validation / comparaison → devrait être dans **controller login.js**
`logout()`	🟣 Méthode	✅	Supprime la session (`loggedEmail`)
`getUser()`	🟣 Méthode	✅	Retourne l'utilisateur connecté
`isAuthenticated()`	🟣 Méthode	✅	Vérifie si un utilisateur est connecté

Commentaire : - Les méthodes `register` et `login` font un peu trop de logique (validation/email existant, hash + check) → techniquement, le model devrait juste **fournir les fonctions de hash et stocker/recupérer** les données.

2 - agents.js

- Contient **les données statiques des agents IA**.
- ✅ Correct, aucun changement nécessaire.

2🦊 Contrôleurs. 7 fichiers

1 - header.js

Nom	Type	Statut	Description
`DOMContentLoaded`	🟣 Méthode anonyme	✅	Remplace le bouton login par SVG, affiche/cacher login/profile/logout selon connexion
`loginBtn.addEventListener(click, …)`	🟠 Fonction	✅	Redirection vers login.html
Appel `setupLogout(logoutBtn)`	🟠 Fonction	✅	Initialise logout avec confirmation

2 - home.js

Nom	Type	Statut	Description
`showAgent(index)`	🟠 Fonction	✅	Affiche agent et met à jour DOM / onglet actif
`tabs.forEach(tab => tab.addEventListener(click, …))`	🟣 Méthode anonyme	✅	Gestion des onglets agents
`buyBtn.addEventListener(click, …)`	🟣 Méthode anonyme	✅	Acheter agent, update user.agents, localStorage, message UX
Mise à jour localStorage agents	🟠 Fonction	✅	Stocke les agents achetés
`showAgent(0)`	🟠 Fonction	✅	Affiche Daïne par défaut

Code achat agent

```
if (buyBtn) {
  buyBtn.addEventListener("click", (e) => {
    e.preventDefault();
    const agentName = buyBtn.dataset.agent;
    if (!user.agents.includes(agentName)) {
      user.agents.push(agentName);
      auth.saveUser(user);
      renderAgent(agents.find(a => a.name === agentName), user);
      purchaseMessage.textContent = `Vous avez bien acheté l'agent
${agentName} !`;
      purchaseMessage.style.color = "#4fc3f7";
      purchaseMessage.style.fontWeight = "bold";
    }
  });
}
```

🔴 Idéalement, cette logique pourrait aller dans un **controller purchase-agent.js**.

3 - login.js

Nom	Type	Statut	Description
`formLogin.addEventListener(submit, ...)`	🟣 Méthode anonyme	✅	Gère connexion avec validation et redirection

4 - logout.js

Nom	Type	Statut	Description
`setupLogout(logoutBtn)`	🟠 Fonction	✅	Logout avec confirmation et suppression session

---

5 - register.js

Nom	Type	Statut	Description
`formRegister.addEventListener(submit, ...)`	🟣 Méthode anonyme	✅	Gère inscription, validation emails/password et stockage

6 - profil.js

Nom	Type	Statut	Description
`DOMContentLoaded`	🟣 Méthodeanonyme	✅	Affiche nom/email, crée onglets agents, affiche agent sélectionné
`btn.addEventListener(click, ...)`	🟣 Méthode anonyme	✅	Affiche agent correspondant quand on clique sur onglet

7 -profil-edit.js

Nom	Type	Statut	Description
`editBtn.addEventListener(click, ...)`	🟣 Méthode anonyme	✅	Affiche / cache formulaire édition profil
`form.addEventListener(submit, ...)`	🟣 Méthode anonyme	✅	Mise à jour nom utilisateur dans localStorage

Acceptable, pas de BDD disponible

3🦊 Viewers. 3 fichiers

1 - agent-renderer.js

Nom	Type	Statut	Description
`renderAgent(agent, user)`	🟠 Fonction	✅	Met à jour DOM avec infos agent, boutons achat, messages, couleur

---

2 - home-ui.js

Nom	Type	Statut	Description
`agents`	🟫 Objet	✅	Définit agents disponibles pour UI

3 - profile-ui.js

Nom	Type	Statut	Description
`DOMContentLoaded`	🟣 Méthode anonyme	✅	Génère onglets agents possédés, affiche agent via `renderAgent`

🦊 Pages HTML

- index.html, login.html, register.html, profile.html, legal-notice.html, privacy-policy.html
- ✅ Correct, contient markup + lien vers controllers et views.

🦊 Synthèse et recommandations

Points forts

- ✅ Séparation Model / Controller / View globalement respectée.
- ✅ Inspiré de Symfony : architecture cohérente.
- ✅ Model gère données, Controller gère logique métier, View gère affichage.

Points à améliorer

- ❌ Validation inputs dans `user.js` → devrait être dans controller.
- ❌ `profil-edit.js` → utiliser un service/controller pour modifier données (localStorage acceptable ici).
- ❌ Logique achats agents dans home.js → créer **purchase-agent.js** si projet plus complexe.
- 🔹 Centraliser duplication validations inputs dans helper/service.