

Panel Web — interface superposée au jeu et intégration Unity

Le panel au-dessus du jeu

Le panel Web donne accès au support UnikBot et au centre de notifications pendant que le jeu Unity WebGL reste affiché. Angular dessine la barre d'outils et les fenêtres au-dessus du canvas Unity. L'utilisateur ouvre une vue, l'utilise puis la ferme pour retrouver le jeu.

Le composant `app-web-panel` est monté dans `AppComponent`, à côté du `router-outlet`. Il est partagé avec les autres pages du site. Dans le jeu, il reçoit la classe `game-panel-host`, qui adapte sa disposition à la barre d'outils.

Barre d'outils en jeu

La barre `game-tools-rail` apparaît lorsque l'utilisateur est authentifié, que la route est `/game`, `/demo`, `/alpha`, `/alpha2` ou `/demoLive`, et que `WebPanelService.isGameReady` vaut `true`.

- La cloche ouvre le centre de notifications et affiche son nombre de notifications non lues.
- Le bouton UnikBot ouvre le support et affiche son compteur de messages non lus.
- La poignée replie ou déploie la barre. Quand elle est repliée, un badge conserve le total des deux compteurs, plafonné visuellement à `99+`.
- Replier la barre ferme la vue ouverte. Ouvrir une vue redéploie la barre.

`AppComponent.openGamePanel()` appelle `WebPanelService.toggle()` : cliquer une seconde fois sur la vue active la ferme; choisir l'autre bouton change de vue.

État partagé et cycle d'ouverture

`WebPanelService` conserve la vue active dans un `BehaviorSubject`. `activePanel$` informe le composant du changement et `currentPanel` expose sa valeur courante. Une seule vue est active à la fois.

Valeur interne	Vue affichée
<code>null</code>	Panel fermé.
<code>notifications</code>	Centre de notifications.
<code>chat</code>	Support Web UnikBot : <code>app-page-support</code> en mode compact.

`open(panel)` sélectionne une vue, `toggle(panel)` alterne ouverture et fermeture, et `close()` remet la sélection à `null`. La touche Échap et le bouton de fermeture appellent `close()`.

Intégration avec Unity WebGL

`PageWebglComponent` héberge `#unity-canvas` et charge le build Unity. À son initialisation, il appelle `setGameReady(false)`. Son listener `window:uniksim:game-ready` passe ensuite l'état à `true` au prochain `requestAnimationFrame`, ce qui rend la barre disponible.

Le composant du panel écoute aussi `window:uniksim:open-panel`. Le champ `detail` accepte une chaîne ou un objet avec `panel`. Seules les valeurs `notifications` et `chat` sont reconnues. Cette entrée permet à un appelant de demander l'ouverture sans changer la route Angular.

```
window.dispatchEvent(new CustomEvent('uniksim:open-panel', {
  detail: { panel: 'notifications' }
}));
```

Dans la branche Game `refactor_notifications`, `MainControlStudent.Start()` et l'initialisation de `MainControlTeacher` appellent `WebGameReadyBridge.Notify()`. En build WebGL, ce bridge appelle `NotifyWebGameReady()` dans `Assets/Src/Plugins/javaLib.jslib`, qui émet `uniksim:game-ready`. Angular reçoit cet événement et affiche les outils. Dans l'éditeur Unity, le bridge écrit seulement un log.

À la destruction de `PageWebglComponent`, `setGameReady(false)` masque les outils du jeu. Le composant enregistre aussi le jeu actif avec `AccountSocketService.setGamePresence(pid, gameId)` après validation de l'accès, puis appelle `clearGamePresence()` à la sortie.

Support UnikBot

La vue support embarque `PageSupportComponent` avec `compact=true`. Ce mode conserve la présentation du jeu : le composant ne réactive pas l'en-tête et le pied de page du site.

Le support charge la conversation IA et les tickets de l'utilisateur. Il gère le fil sélectionné, les nouveaux messages, les mises à jour de tickets, les indicateurs de saisie et le statut des administrateurs. `SupportService` porte les appels métier, `SupportStateService` réconcilie les données et `SupportSocketService` fournit les événements temps réel.

La fenêtre support utilise un fond assombri cliquable pour fermer le panel. Sur ordinateur, elle peut atteindre 880 × 700 px, dans les limites du viewport. En mode jeu, elle est décalée pour laisser la barre accessible. Sur petit écran, elle occupe presque toute la surface et réserve de l'espace aux outils.

Voir [Support temps réel](#) pour les conversations et les transports.

Centre de notifications

La vue notifications est un popover positionné près de la cloche. Sa position est recalculée à l'ouverture, au redimensionnement et au scroll. Sur ordinateur dans le jeu, il apparaît à gauche de la barre. Sa largeur est limitée à 380 px; sa liste défile lorsque le contenu dépasse la hauteur disponible.

La couche externe laisse passer les interactions vers le jeu avec `pointer-events: none`; la fenêtre réactive ses propres interactions. Un clic hors de la fenêtre et de la cloche ferme la vue.

`UserNotificationService` partage la liste entre le panel et les compteurs :

Action	API Web
Charger la liste	<code>GET /notifications</code>
Marquer comme lu	<code>POST /notifications/:uid/read</code> avec <code>isRead</code>
Supprimer	<code>DELETE /notifications/:uid</code>

La première ouverture charge la liste; le bouton de rafraîchissement permet de la recharger. Le panel affiche les états de chargement, d'erreur avec nouvelle tentative, et de liste vide. Cliquer une notification non lue la marque comme lue; le bouton de suppression retire l'entrée sans déclencher ce clic de lecture.

`AccountSocketService.onNotificationCreated$` alimente `UserNotificationService.upsert()`, qui fusionne les entrées par UID. `AppComponent` recharge les notifications après reconnexion du compte. Les titres et corps utilisent `uniksim.notification_<type>_title` et `_body`, avec un texte par défaut si la traduction manque.

Côté Game, `NotificationManager.SaveNotification()` délègue à `BackEndBase.SaveNotification()`, qui envoie `POST /saveNotification`. L'affichage du panel et sa liste sont gérés côté Web. Voir

[Notifications et Push](#) pour la réception serveur, l'inbox et le push.

Fichiers à modifier

Fichier	Responsabilité
<code>uniksimFrontEnd/src/app/app.component.html</code> , <code>.ts</code> et <code>.scss</code>	Montage global, barre en jeu, boutons, compteurs et accès hors jeu.
<code>service/webPanel/web-panel.service.ts</code>	Vue active et état de disponibilité du jeu.
<code>Components/WebPanel/WebPanel.component.ts</code> , <code>.html</code> et <code>.scss</code>	Ouverture, fermeture, contenu, position et superposition.
<code>Pages/PageWebgl/PageWebgl.component.ts</code>	Chargement Unity, disponibilité et cycle du jeu actif.
<code>Pages/PageSupport/</code> et <code>service/support/</code>	Support embarqué et ses conversations.
<code>service/notifications/user-notification.service.ts</code>	Liste, compteur et mutations de notifications.
<code>service/account/account-socket.service.ts</code>	Événements du compte et jeu actif.
<code>Game/Assets/Src/Game/Notifications/NotificationManager.cs</code> et <code>Game/Assets/Src/Game/BackEnd/BackEndBase.cs</code>	Envoi des notifications depuis Unity.

Vérifier une modification

- Entrer dans le jeu : les outils attendent le signal de disponibilité.
- Ouvrir chaque vue depuis la barre, changer de vue, recliquer le bouton actif, fermer avec Échap et avec la croix.
- Replier la barre avec une vue ouverte, puis la redéplier; contrôler les compteurs.
- Contrôler le placement sur ordinateur et petit écran, et les clics qui doivent atteindre le jeu autour du popover.
- Tester le support intégré, le chargement des notifications, la lecture, la suppression et une reconnexion.
- Sortir du jeu : vérifier le retrait des outils et du jeu actif côté compte.

Sources de cette page : frontend Web de la branche `messagerie_fixes` et code Game de la branche `refactor_notifications`, inspecté le 30 septembre 2026.

Revision #4

Created 2026-09-30 14:51:56 UTC by Mathis

Updated 2026-09-30 15:22:24 UTC by Mathis