Application Windows Gargantua 8

Dossier d'architecture technique

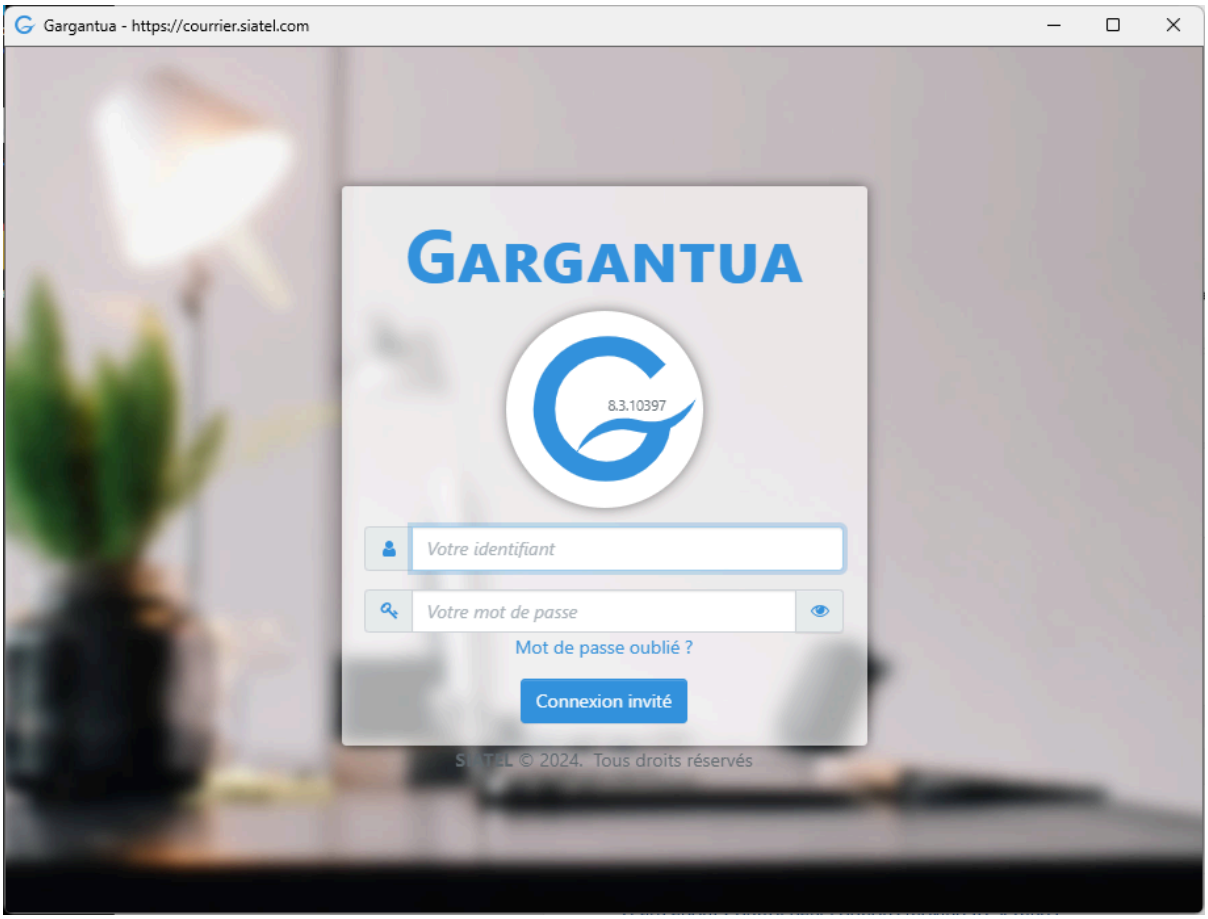
Table des matières

1. Introduction

Ce document décrit les choix techniques adoptés pour le packaging de l'application web Gargantua 8 en tant qu'application Windows. Cette approche d'encapsulation répond à des contraintes spécifiques de sécurité qui interdisent l'utilisation des éléments suivants :

- Extensions de navigateur, en raison de leurs potentielles vulnérabilités et du niveau élevé d'accès qu'elles requièrent.
- Modules externes au navigateur, car ces derniers peuvent contourner les mécanismes de sécurité standards du navigateur, qui doit continuer à être perçu comme un environnement isolé et sécurisé.

De plus, cette stratégie garantit une étanchéité totale avec le système d'exploitation, empêchant toute interaction non autorisée qui pourrait compromettre l'intégrité ou la confidentialité des données traitées par l'application.



2. Choix de la technologie d'encapsulation

Suite à différentes options étudiées, nous avons retenu et évalué de manière approfondie deux technologies d'encapsulation :

1. Chromium Embedded Framework (CEF), comme par exemple electronJS (https://www.electronjs.org/)
2. Microsoft Edge WebView2

Notre comparaison a porté sur plusieurs critères clés, incluant la compatibilité avec notre application Gargantua, les performances, l'accès aux API Windows, et la facilité d'intégration.

Evaluation du framework electronJS

Le principal avantage de cette technologie est sa compatibilité avec plusieurs systèmes d'exploitation, tels que Windows, Linux, etc. Toutefois, cet aspect présente peu d'intérêt pour le contexte qui nous intéresse, puisque l'application Gargantua est destinée à des utilisateurs finaux dont les postes de travail fonctionnent sous Windows.

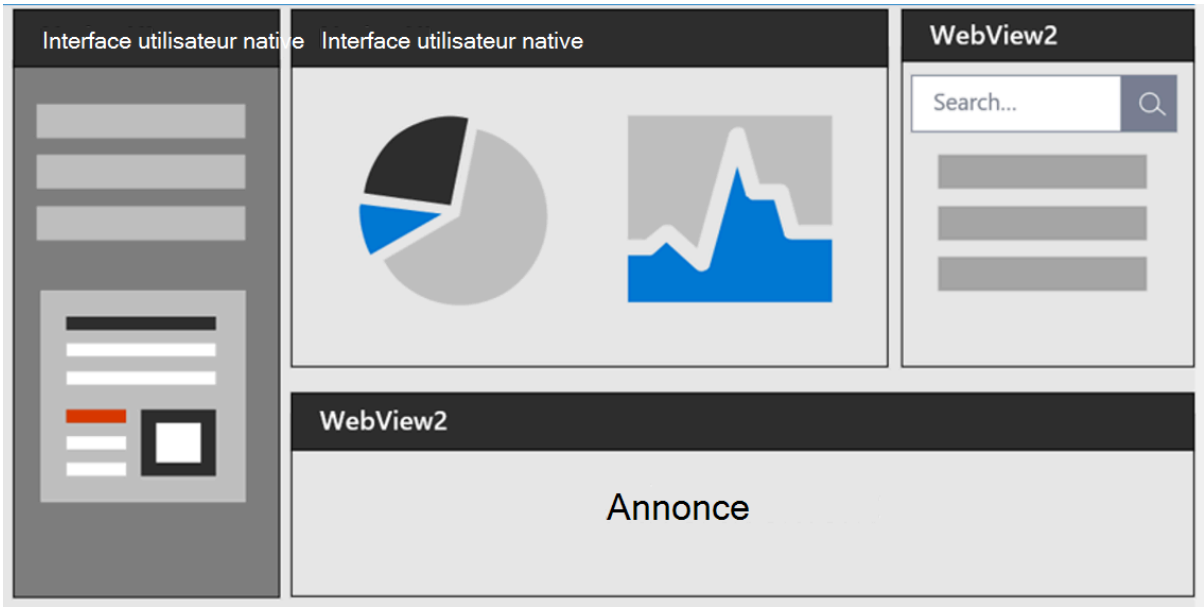
D'autre part, dans le cadre de l'application Gargantua, la plateforme ElectronJS se révèle être trop lourde, car elle inclut des modules non utiles au fonctionnement de l'application.

Evaluation du framework Microsoft Edge WebView2

L'environnement visé étant Microsoft Windows et l'utilisation du pack Microsoft Office étant essentielle dans notre logiciel, la technologie Microsoft Edge WebView2 (https://learn.microsoft.com/fr-fr/microsoft-edge/webview2/) s'est imposée comme la solution la plus adaptée. Cette solution a été privilégiée du fait de la disponibilité native de ses composants sur les plateformes Windows, qui sont précisément notre environnement cible.

Le composant Microsoft Edge WebView2 permet d'intégrer des technologies Web (HTML, CSS et JavaScript) dans des applications natives. Le composant WebView2 utilise Microsoft Edge comme moteur de rendu pour afficher le contenu web dans les applications natives.

Avec Microsoft WebView2, il est possible d'incorporer du code web dans différentes parties de l'application native ou créer l'ensemble de l'application native au sein d'une instance WebView2 unique.



L'incorporation du composant Microsoft WebView2 dans une application permet à l'application d'accéder à diverses méthodes et propriétés fournies par le biais des classes ou interfaces du composant Microsoft WebView2.

Microsoft WebView2 propose des centaines d'API qui fournissent un vaste ensemble de fonctionnalités, allant de l'amélioration des fonctionnalités de la plateforme native de l'application à la possibilité pour l'application de modifier les expériences de navigateur.

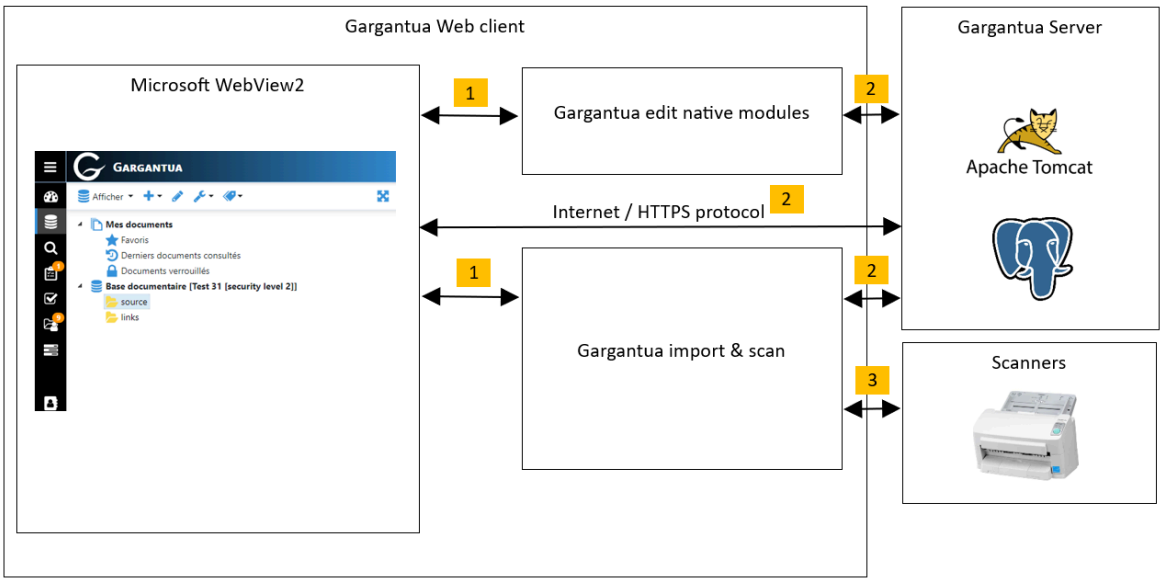
L'utilisation de cette technologie permet de conserver le même niveau d'expérience utilisateur que celui obtenu avec la version web standard de Gargantua 8 (qui repose sur l'utilisation d'extensions de navigateur), tout en évitant les contraintes de sécurité que ces extensions pourraient engendrer dans certains environnements réseau.

3. Architecture technique

Dans le cadre du POC réalisé pour valider le concept, le composant Microsoft Edge WebView2 a été inclus dans une application standard SDI (Single Document Interface) C, mais une approche MDI (Multiple document interface) est également possible.

Ce composant est dirigé automatiquement vers l'adresse du serveur Gargantua, et affiche la page d'accueil directement au démarrage.

L'architecture de l'application Gargantua Web Client est décrite dans le schéma ci-dessous :

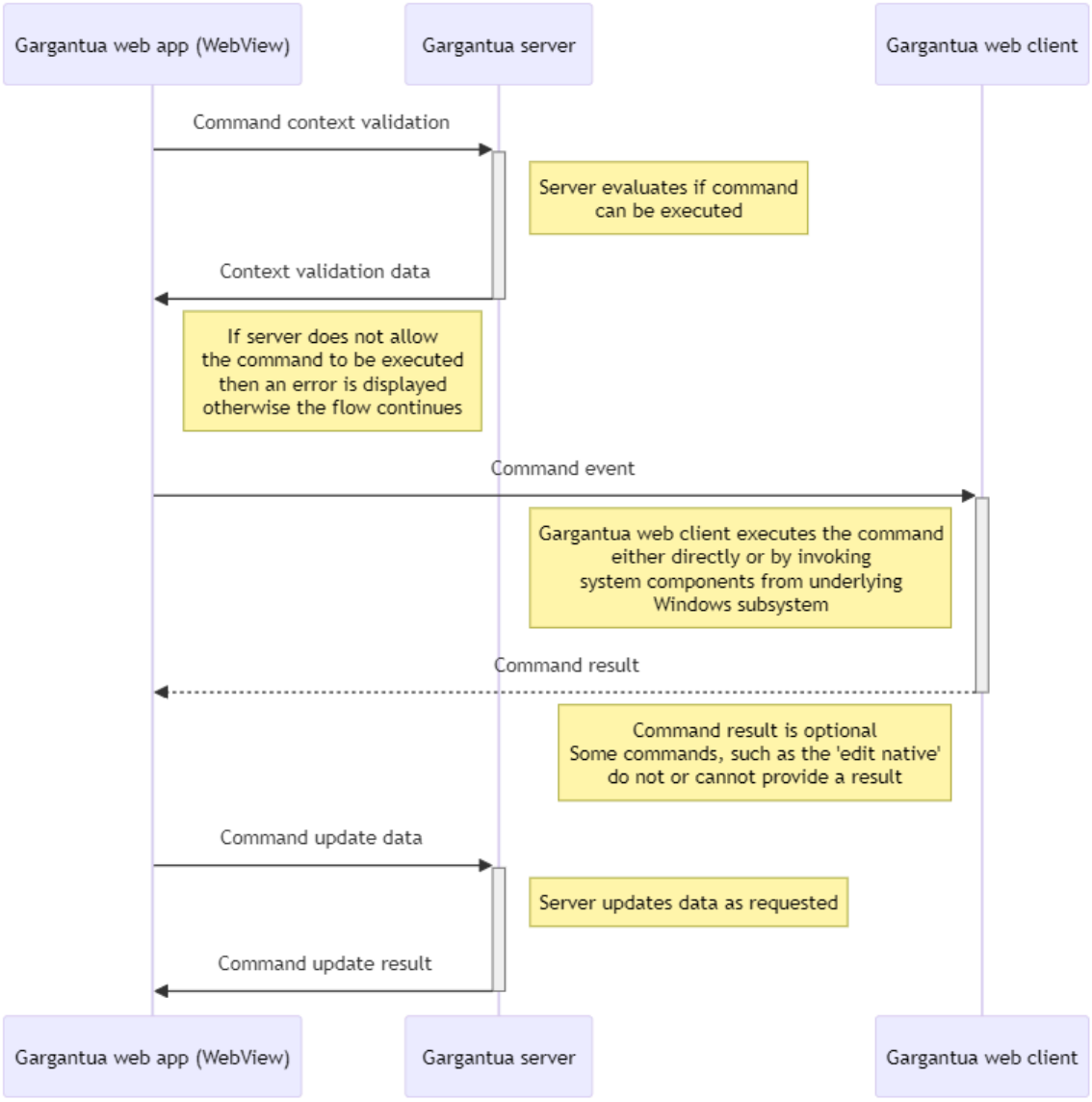


Les canaux de communication :

1. les canaux (1) utilisent des événements internes spécifiques au module Microsoft WebView2.
2. les canaux (2) sont des communications qui utilisent le protocole HTTPS.
3. le canal (3) est une communication assurée par l'OS pour les ports USB sur lesquels les scanners sont connectés.

4. Fonctionnement des événements

Le schéma ci-dessous décrit la séquence des événements qui s'exécutent lorsqu'une action utilisateur nécessite les composants spécifiques OS inclus dans l'application Gargantua Web.



Deux types de commandes sont distinguées :

1. Les commandes qui ne produisent pas de résultat pour la partie web ; dans cette catégorie on retrouve l'ouverture par l'application native ; dans ce cas, le document est sauvegardé de manière asynchrone dans Gargantua à la fin de l'opération, sans autre notification vers le client web.
2. Les commandes qui remontent un résultat, comme par exemple les signatures.

La seule différence entre ces deux catégories apparaît à la fin de la séquence d'événements.

La liste ci-dessous décrit en détail la séquence des actions qui s'applique pour une commande ne produisant pas de résultat (par exemple, l'ouverture d'un document dans son application native) :

1. Dans la page affichée, l'utilisateur choisit d'ouvrir un document dans son application native.
2. Le composant JavaScript de la page web réalise les vérifications nécessaires avant d'exécuter la commande. Si la commande ne peut pas être effectuée, les actions mises en œuvre sont les mêmes que si la page avait été ouverte dans un navigateur, y compris en ce qui concerne l'affichage des messages d'erreur.
3. Le composant JavaScript de la page web détecte que la page est affichée dans l'application Gargantua Web Client et déclenche un événement d'ouverture dans l'application native.
4. Gargantua Web Client transmet la commande au composant qui va effectuer l'ouverture de l'application native.
5. Le composant d'ouverture par l'application native télécharge le document dans un dossier temporaire.
6. Le composant d'ouverture par l'application native ouvre le document et attend la fin de l'opération.
7. Lors de la fin de l'opération, le document modifié est sauvegardé dans Gargantua en utilisant les paramètres récupérés au début de l'opération.

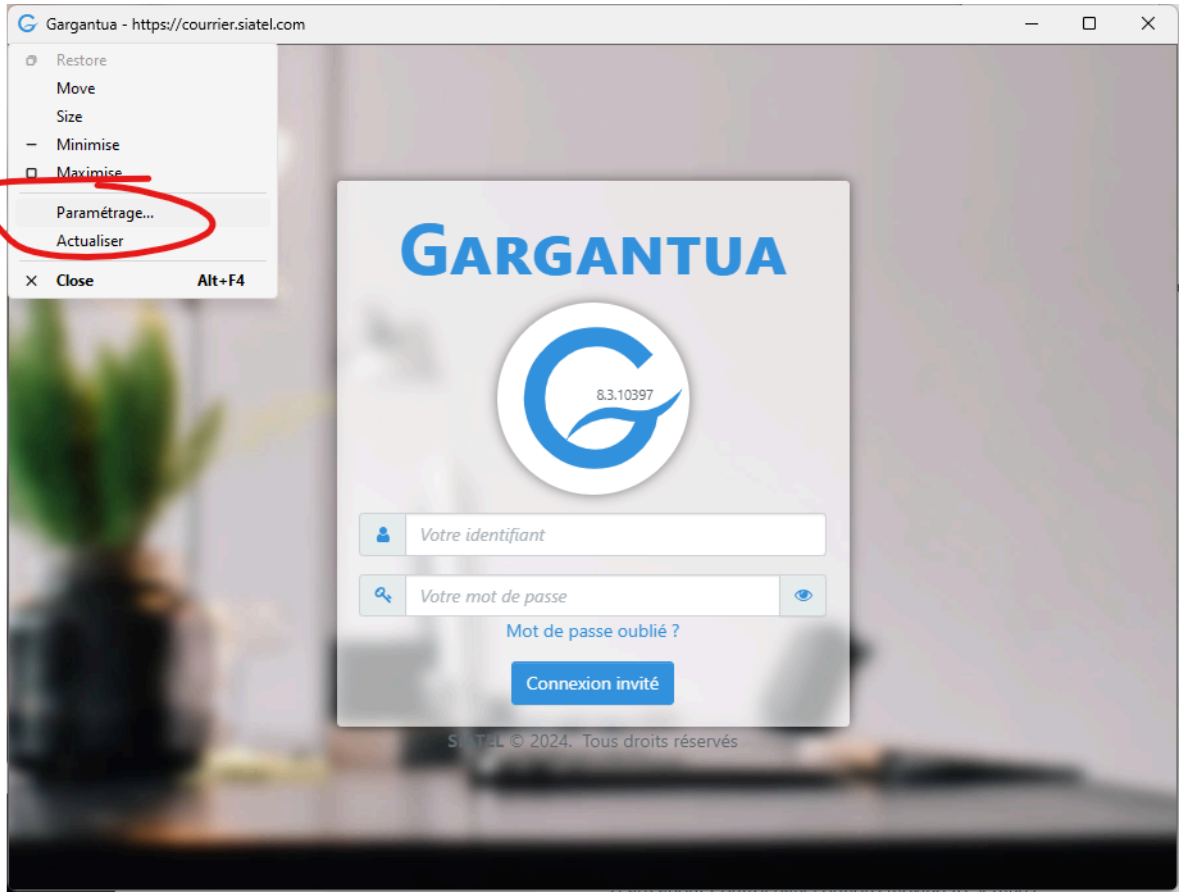
Dans le cas des opérations remontant un résultat, la séquence d'événements est la même, puis une notification avec le résultat est envoyée vers les composants JavaScript de la page afin de pouvoir effectuer les opérations spécifiques.

5. Détails techniques

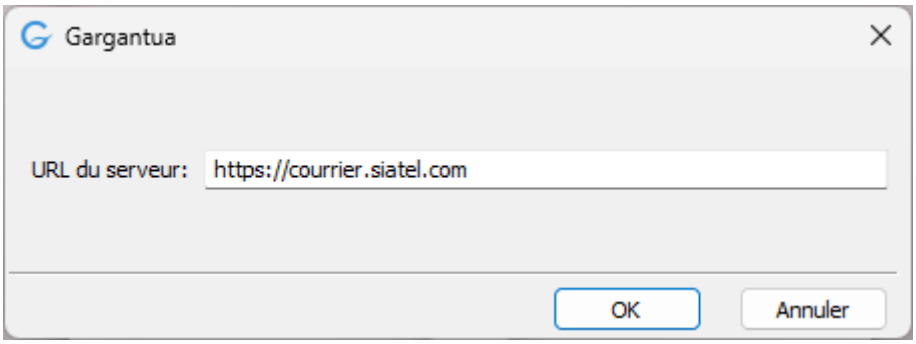
Cette section inclut quelques détails techniques sur l'architecture précédemment présentée.

5.1. Détails concernant la connexion

Dans l'application Windows Gargantua, la connexion s'effectue automatiquement vers un serveur Gargantua préparamétré ; celui-ci peut-être modifié dans la fenêtre de paramétrage accessible depuis le menu système de la fenêtre principale de l'application.



L'utilisateur peut ainsi choisir l'adresse du serveur.



5.1. Evénements Microsoft WebView2

L'envoi des événements vers le composant Microsoft WebView2 est effectué comme ceci :

```
if (isLoadedFromClientApp()) {
    if (hasResponse === true) {
          var handler = function(e) {
               if (e.data.op && e.data.op != op)
                    return;

               window.chrome.webview.removeEventListener('message', handler);

               if (e.data.message && e.data.message.length > 0 && data.silent !== true)
                    G.Alert.display(e.data);

               if (_.isFunction(data.callback))
                    data.callback(e.data);
          }

          window.chrome.webview.addEventListener('message', handler);
     }

     window.chrome.webview.postMessage({
          cmd: op,
          params: (typeof data.object === 'string') ? data.object :
JSON.stringify(data.object)
     });

     return;
}
```

6. Evolutions

Dans le cadre du POC réalisé pour valider le concept de l'application Windows Gargantua avec Microsoft Edge WebView2, seules des fonctionnalités très simples ont été mises en place, l'objectif de ce POC étant d'évaluer la technologie utilisée et de valider la conformité de l'application Gargantua Web Client aux normes de sécurité en vigueur.

Un certain nombre d'évolutions doivent être apportées. A noter que cette liste n'est pas exhaustive, et qu'une analyse des fonctionnalités est requise en amont de la phase de développement.

6.1. Connexion à plusieurs serveurs

La version du module réalisé permet et gère la connexion vers un seul serveur Gargantua. L'évolution à prévoir consiste à mettre en place une fenêtre de gestion d'une liste de serveurs et la possibilité de se connecter à un de ces serveurs Gargantua.

6.2. Mode d'affichage multi-onglets

La version du module réalisé utilise un seul onglet de navigateur.

Il est à prévoir, pour plus de confort, une interface donnant la possibilité d'ouvrir plusieurs onglets ; certains de ces onglets pourront être épinglés, d'autres pourront s'ouvrir dans le même ordre lorsque l'utilisateur ré-ouvrira l'application.