Rapport de projet : Instabot

Par Sacha Hu et Loïc Loko

Ce projet avait pour but d'automatiser des actions d'instagram. Tout d'abord nous avons commencé par coder les différentes actions possibles (dont nous avions besoin) sur instagram.
Ces actions ont donné les fonctions :

- Search : permet de rechercher un hashtag
- ClickPic : permet de cliquer sur une photo depuis le profil
- Like : permet d'aimer une photo depuis la photo ouverte
- closePic : permet de fermer la photo
- follow : permet de commencer à suivre un utilisateur et de sauvegarder son nom
- unfollow : Si l'on suit trop de personnes, cette fonction permet d'arrêter d'en suivre
- hashSort : permet de ne garder que les hashtags pertinents
- hashSearch : permet de récupérer tous les hashtags sous une photo.

Fonctionnement Général

La plupart des fonctions fonctionnent sur le même principe : déterminer où se trouve l'élément que l'on veut obtenir (sur la page web) à l'aide d'un querySelector, puis naviguer à l'aide des childNodes pour trouver l'emplacement exact. En fait, nous avons remarqué que certains éléments restent les mêmes sur toutes les pages instagram peu importe l'hashtag ou le nom d'utilisateur recherché. C'est pour cela que le querySelector se base sur les éléments qui restent inchangés d'une page à l'autre. Puis la méthode childNodes nous permet d'accéder aux éléments qui sont modifiés d'une page à l'autre. Tant qu'on sait dans quel ordre l'arbre est fait, on peut accéder à tout ce qu'on veut.
Une fois que l'on a correctement identifié la position de l'élément recherché sur la page, il ne reste plus qu'à cliquer, écrire le texte souhaité, etc.
Toutefois, le chargement d'instagram met un certain temps. Nous aurions souhaité utiliser l'observer pour obtenir en temps réel des informations sur les nouveaux nœuds créés pour savoir si l'élément que l'on souhaite obtenir à été créé mais la contrainte de temps nous en a empêché. Pour contourner ce problème nous avons écrit une fonction sleep qui permet d'attendre un laps de temps avant de renvoyer une promise resolve.

Fonctionnement asynchrone

Les processus indiqués ci-dessus nécessitent tous d'être asynchrones. En effet, on ne peut pas récupérer un élément sur la page s'il n'a pas encore été créé. Autrement dit, on ne peut pas aimer une photo que l'on n'a pas encore vu. Ainsi, toutes les fonctions sont appelées par l'intermédiaire de await dans une grande fonction asynchrone. Il est donc important que nos fonctions renvoient une promise resolve au await.

Après avoir codé cela nous avons tenté de déterminer un ordre logique dans lequel appeler les fonctions :

Tout d'abord, on effectue une recherche d'hashtag (stocké dans une liste hashtag_list). Puis, une fois arrivé sur le profil, on clique sur une photo qui s'ouvre en grand, on clique sur le bouton like et on récupère les hashtags dans une liste de hashtags temporaires appelée hashtag_list_temp. Ensuite, on ferme la photo et on réitère ce procédé en rouvrant deux ou trois photos afin de remplir la liste de hashtags temporaires. Enfin, on clique sur le bouton follow et on trie la liste de hashtags temporaires comme ceci : si un hashtag apparait plus d'une fois dans la liste alors on peut considérer qu'il s'agit d'un hashtag important et on le stocke dans hashtag_list avec la méthode shift. Sinon, cet hashtag n'est pas important. Après avoir obtenu de nouveaux hashtags dans notre liste de hashtags, on peut effectuer une nouvelle recherche.
La dernière fonction propre à instagram est unfollow. A chaque fois que l'on commence à suivre quelqu'un, on récupère son nom d'utilisateur et on le stock dans une liste. Lorsque cette liste dépasse une certaine longueur définie par l'utilisateur, la fonction unfollow permet de retirer le premier utilisateur suivi grâce à la méthode shift, de le rechercher et de cliquer sur unfollow. Cela permet de ne pas accumuler sur le compte de l'utilisateur du bot, des abonnements à des comptes qui ne l'intéressent pas.

Eviter la censure d'instagram

Ce processus de base est plutôt simple. Cependant, ce comportement étant très facile à tracer pour instagram, nous avons tenté d'introduire un peu de hasard. Pour cela nous avons essayé de rendre aléatoire le nombre de photo à aimer sur chaque profil, la position de chacune des photos à aimer (clickPic prend en argument une liste de 2 éléments dont le premier est la rangée et le deuxième la colonne), et le fait de commencer à suivre cet utilisateur ou non. La fonction getRandomIntInclusive que l'on a trouvé sur stackOverflow permet de générer un nombre aléatoire entre deux valeurs entrées en paramètre.

Nous avons profité de la nouvelle version de Instagram (qui permet de suivre un hashtag) pour ne pas différencier un hashtag d'un utilisateur.

Difficultés rencontrées :

Les principales difficultés de ce projet étaient de comprendre comment utiliser l'observer, de faire fonctionner correctement les fonctions asynchrones, et de trouver l'emplacement exact d'un objet html et d'y accéder.
Nous nous sommes beaucoup attardés sur l'observer pour tenter d'obtenir un bot qui ne perdrait pas de temps inutilement. Au final, nous n'y sommes pas arrivés et avons préféré ne pas perdre plus de temps. Nous avons également pensé utiliser la méthode hasChildNodes de javascript pour éviter d'utiliser l'observer mais nous n'avons pas eu le temps de tester cette solution.
Les fonctions asynchrones ont également posé problème. En effet, d'après la documentation trouvée sur internet, parfois il était utilisé async function pour définir la fonction asynchrone et parfois juste async. D'autre part, le renvoie de promise resolve dans les fonctions await était un problème que l'on a dû surmonter.

Obtenir les objets html sur la page instagram nous a posé quelques problèmes. Parfois parce que la méthode querySelector ne fonctionnait pas dans un cas précis (que nous ignorions) parfois parce que l'élément mutait d'une page à une autre ce qui rendait la recherche impossible. Nous avons contourné ce problème en utilisant querySelector sur l'objet le plus global restant inchangé d'une page à l'autre, puis nous nous somme orientés à l'aide de childNodes.