Evaluation Architecture d'Application

contact@ls-a.fr
https://www.ls-a.fr

Table des matières

Contexte

Vous êtes amené(e)s à travailler sur un nouveau projet de retouche d'image social, Puzzix, mélange entre "Fuzz" et "Pix". Le principe est le suivant : à partir d'une image fortement déformée, faire deviner à ses ami(e)s l'image d'origine, ainsi que les transformations (si possible dans l'ordre) qu'a subi l'image.
Après avoir téléversé une image, un(e) internaute peut ajouter de nombreuses retouches sur celle-ci : rotation, redimensionnement, filtres de couleur, effets type kaléidoscope et ainsi de suite. Pour cela, l'internaute pourra effectuer des glisser/déposer d'effets représentés par des petites boîtes, ainsi que définir des valeurs, dans l'esprit du logiciel en ligne Scratch.

Architecture matérielle

Pour ce projet, votre entreprise a divisé l'architecture matérielle en différents serveurs :

- Un serveur appelé "Alice" délivrant le frontend de l'application : formulaire d'envoi d'images, glisser/déposer des blocs et construction des étapes de transformation ;
- Un serveur appelé "Bill" recevant les étapes de transformation après soumission du formulaire : c'est le backend de l'application. Son rôle est de stocker les étapes et les données des utilisateurs ;
- Un serveur appelé "Claire", invoqué par Bill pour stocker les images d'origine. C'est un serveur disposant de beaucoup d'espace disque ;
- Un serveur appelé "Diego", invoqué par Bill pour effectuer toute transformation d'image lié à des filtres ;
- Un serveur appelé "Elise", invoqué par Bill pour effectuer toute transformation d'image lié à des effets.

Notes

Définitions :

- Un filtre : quand un pixel donné de l'image d'origine se retrouve au même endroit après transformation. Exemple : un filtre de couleur, de flou, de netteté, retirer les yeux rouges, etc.
- Un effet : quand un pixel donné de l'image d'origine est déplacé après transformation. Exemple : une rotation, un écrasement, un effet miroir, kaléidoscope, etc.

Les serveurs "Diego" et "Elise" peuvent être clonés. En effet, ils sont d'un fonctionnement simple : recevoir une image, y appliquer une modification, renvoyer l'image. Aucun stockage n'est associé à ces serveurs. Ce fonctionnement permettra de faciliter le traitement des images lors de périodes de forte charge. En effet, si l'application est fortement sollicitée, on pourra créer de nouveaux "Diego" et "Elise", tant que "Bill" conserve une liste de tous ses interlocuteurs.

Exemple de parcours utilisateur

Fabrice, un utilisateur :

- Visualise la page de connexion du réseau social Puzzix (page fournie par le serveur Alice) ;
- Rentre ses identifiants et soumet le formulaire (données envoyées au serveur Bill qui vérifie la connexion et renvoie un succès ou échec à Alice) ;
- Fabrice échoue à entrer ses identifiants : sur la page, un message d'erreur s'affiche ;
- Fabrice rentre de nouveau ses identifiants et réussit : il est redirigé vers un canevas avec les blocs représentant les retouches d'image ;
- Fabrice téléverse quatre images, et sélectionne "la bonne", à savoir celle que ses ami(e)s devront trouver. Il fait des glisser/déposer des blocs de filtres et d'effets pour constituer la séquence de modification d'images (ici, tout est réalisé en frontend. Les données sont envoyées à l'étape suivante)
- Quand Fabrice est satisfait, il clique sur "Valider". (Toutes les données sont alors envoyées vers Bill, qui envoie à Claire les images, et qui stocke le séquencage des opérations).
- Quand les ami(e)s de Fabrice se rendent sur la page du challenge, l'image altérée apparaît : il faudra retrouver la bonne image, et l'ordre des opérations (ici, Alice demande à Bill de générer l'image altérée : Bill, pour cela, réalise toute la séquence d'altérations, en demandant d'abord l'image d'origine à Claire, puis en appelant successivement Diego et Elise).

Dans une documentation, vous devrez :

- Représenter l'architecture sous la forme d'un schéma, en présentant les responsabilités de chaque serveur ;
- Sélectionner des modèles éprouvés de communication entre les différents serveurs, voire proposer des outils implémentant ces modèles ;
- Proposer la nomenclature des APIs web entre ces serveurs : méthodes, URL, données en entrées, données en sortie ;
- Bonus : proposer une autre manière d'organiser le projet, qui vous semblerait plus efficace, tant qu'elle inclut plusieurs serveurs, et détailler pourquoi.

En code, vous devrez proposer **au choix** :

- Le code permettant de stocker les opérations d'altération puis d'appeller successivement les différents serveurs, en fonction de cette séquence ;
- Le code permettant au serveur Bill de référencer plusieurs clones de serveurs Diego et Elise, pour envoyer les demandes de transformation d'une manière qui vous semble équilibrée.

Modalités de rendu

Le rendu doit être réalisé en groupe de 1 à 3 personnes.
La note portera à moitié sur la documentation, à moitié sur le code. Le soin apporté à ces deux rendus permettra parfois d'obtenir un bonus ou un malus en fonction de cette qualité.