ReSnapIt (titre à déterminer)

Groupe:

Aurélien Pintrand - PM1 - Alternant
Thomas Chazot - PM3 - Cherchant
Mathilde Jean - PM2 - Alternant
Lucas Zborowski - PM2 - Alternant
Pierre Ferreira - PM2 - Cherchant

Tuteur:

Pascal Lafourcade

Principe:

ReSnapIt est une application mobile cross-platform permettant de s'amuser entre amis, ou en ligne avec des inconnus, à travers des petites chasses au trésor. Le fonctionnement est le suivant:

Dans le cas d'une partie entre amis:

- Un joueur "hôte" héberge une partie, et ses amis peuvent rejoindre.
- Le joueur hôte prend une ou plusieurs photo(s) de son environnement avec son téléphone, directement depuis l'application. Il sera possible de prendre la photo avant, mais elle devra impérativement être prise depuis l'application (à cause des conditions plus bas)
- Une fois la photo prise, les autres joueurs devront essayer de retrouver l'endroit où elle a été prise, et devront prendre une photo qui ressemble le plus possible à l'originale.
- Une fois les photos prises, un score leur sera attribué en fonction de leur ressemblance avec la photo originale.

Dans le cas d'une partie en ligne:

- Les joueurs pourront, lorsqu'ils arriveront sur l'application, voir un ensemble d'images publiques, en sélectionner une et essayer de la

retrouver. Ils pourront également poster leurs propres images, pour que les autres puissent les retrouver.
- Pour éviter les images introuvables (ex: photo en intérieur), il serait possible pour les utilisateurs de noter les photos en fonction de leur difficulté et de leur intérêt.

Fonctionnement technique:

Pour réaliser une comparaison entre deux images, nous avons besoin d'un modèle de deep learning entraîné à la reconnaissance d'images. Nous en avons déjà essayé plusieurs et nous en avons trouvé un qui fonctionne relativement bien. Cependant, ce n'est pas assez. Les modèles existants ne sont pas encore assez poussés pour avoir une fiabilité suffisante. C'est pourquoi nous voulons procéder de la manière suivante:

Nous allons mettre en place un "custom score" pour comparer les images. C'est-à-dire que le score obtenu par l'algorithme ne comptera que pour une partie du score final de reconnaissance. Nous allons aussi utiliser les fonctionnalités suivantes:

- GPS: afin d'obtenir une position plus ou moins précise de l'endroit où l'utilisateur prend sa photo. Il sera alors possible d'éliminer les photos qui ressemblent à l'originale, mais prises à un endroit complètement différent.
- Boussole: permet de connaître la direction dans laquelle est prise la photo. Si jamais deux photos ont des orientations complètement différentes, ce ne sont pas les mêmes photos.
- Gyroscope: permet de déterminer l'orientation du téléphone lorsque la photo est prise. 2 photos prises avec une orientation complètement différente ne peuvent être les mêmes.

En modifiant le nombre de points obtenus par l'algorithme avec ces paramètres additionnels, il sera possible de déterminer de manière beaucoup plus précise si les deux images se ressemblent. De plus, l'avantage d'avoir plusieurs moyens de vérification d'image est que si l'un d'entre eux ne fonctionne pas correctement (ex: GPS qui indique la personne à 10m de son endroit réel), les autres sont là pour le remonter.

Technologies utilisées:

Algorithme de reconnaissance d'images: Python

La plupart des algorithmes de deep learning sont réalisés en Python, donc nous n'avons pas vraiment le choix sur ce point. (Nous nous laissons toutefois la possibilité de changer de langage si nous trouvons mieux)

Application mobile: Maui

Nous pensons réaliser l'application mobile en Maui. Sachant que nous avons dans notre groupe deux cherchants, nous préférons éviter de partir sur un nouveau framework comme le Flutter, pour ne pas perdre trop de temps.
De plus, nous avons actuellement des cours de Maui, donc nous saurons mettre en place les bonnes pratiques.

Base de données: MySQL

Bien que la base de données contienne des images, elle est très simple dans sa conception (nous aurons très peu de tables). C'est pourquoi nous utiliserons sûrement MySQL.

API: ?

Nous ne savons pas encore ce que nous allons utiliser pour les API.

Serveurs: C# ?

Nous ne sommes pas encore sûrs, mais il s'agira probablement de C#, par souci de cohérence avec l' application mobile (et éventuellement l'API si nous utilisons ASP .NET Core)

Pistes d’amélioration:

Si jamais nous avons fini l’application mobile avant la fin, nous avons pensé à des modifications que nous pourrions mettre en place:

- Possibilité pour les utilisateurs de laisser des commentaires sur les photos publiques (implique un filtre de langage et potentiellement un système de report et de vérification d’âge)
- Modération des images avec un algorithme pour éviter les contenus obscènes