AWA1 – Unity Mobile AR

Introduction

Le projet est une reprise de l'ancien projet AWA de Dardan Iljazi (avec son autorisation) qui a pour but de de jouer à un quiz sur une liste de quiz disponibles. La particularité est qu'il utilise une partie réalité augmentée qui permet de récupérer un quiz via l'id donné du QR code.

Les fonctionnalités

Les principales fonctionnalités de l'application sont :

- Affichage des quiz (récupéré par un API)
- Possibilité de jouer un quiz
- Le résultat d'un quiz
- Récupérer un quiz avec un QR code

La fonctionnalité sur laquelle j'ai travaillé :

- Récupération des quiz via l'API mis à disposition

Amélioration :

- Correction des bugs / Mis à jour des packages

Fonctionnalité : Récupération des quiz via l'API mis à disposition

La première fonctionnalité était d'implémenter l'API fourni par Mr. Mottier. L'API contient des routes avec des endpoints mais aussi des routes dite « Partially Nested » (Nommé ainsi dans l'application par Dardan). Par exemple, pour avoir la liste des questions pour un quiz donné, nous avons cette route à disposition « http://api.kwiz.mycpnv.ch/api/quizzes/{quizzId}/questions ». Cette route est considérée comme endpoint dans l'application. Mais pour le retour des réponses, celui-ci se trouve dans la même route que les questions. Cela demande une modification dans le modèle de notre API pour avoir en sorte de récupérer les données en endpoint et en partially nested.

Dardan avait travaillé sur cette partie-là mais elle n'était pas complète. Il a cependant documenté le « Comment faire » pour mettre en place son propre API. C'est sur cette fonctionnalité que j'ai passé le plus de temps dessus. La majeure partie a été de me replonger dans le code de Dardan et de comprendre les différentes classes utilisées et comment elles fonctionnaient entre elles. J'ai dû aussi lire les documents qui a mis à disposition sur comment le l'application fonctionne, comprendre le workflow mis en place, comment mettre en place son propre API.

Une fois que j'avais compris la base (sur Unity et le code), je pouvais mettre en place mon modèle API. L'un des premiers problèmes que j'ai rencontrés est que le mode test/play sur Unity ne marchait pas avec l'API. Il fallait que je passe en build sur le téléphone pour que cela passe ou que je debug ce problème.

L'autre problème fût que je pensais que l'API était en endpoint entièrement alors qu'il n'y a pas de endpoint pour les réponses des questions. J'ai dû un peu remodeler et fusionner le endpoint avec la partially nested. C'est là où j'ai passé passablement de temps à chercher dans la documentation de Dardanet ses anciens modèles qu'il a mis à disposition comme exemple. Une fois que cette fusion était faite, les quiz récupérés de l'API fonctionnaient. On peut lancer un quiz et jouer ce dernier. Le résultat (le nombre de question juste ou faux) est affiché à la fin du quiz.

Voici le code ajouté pour avoir accès aux réponses de la question.

```
KwizApiModel.QuestionsInAPI questionsData = JsonUtility.FromJson<KwizApiModel.QuestionsInAPI>(json_questions_with_answers);

Answers answers = new Answers();
// Let's begin to search the questionId we need in this case
foreach (QuestionInAPI questionData in questionsData.data)
{
    Debug.Log(questionId);
    if (questionId.ToString() == questionData.id.ToString())
    {
        // Now that we found the question with the questionId we need, let's get all answers
        foreach (AnswerInAPI answerData in questionData.answers)
        {
            answerData.MapAPIValuesToAbstractClass();
            answers.AddAnswer(answerData);
        }

        return answers;
    }
}
```

Mes différentes sources :

La documentation de Dardan : https://github.com/Seni-J/Abstract_Quiz_AWA/tree/master/docs

La documentation de Unity : https://docs.unity3d.com/2020.1/Documentation/Manual/index.html

Amélioration : « Correction des bugs / Mis à jour des packages »

Ceci n'est pas une amélioration à proprement parler mais c'est aussi une partie de ce que j'ai fait durant ce projet. La première installation des packages a causé quelques soucis dans le code que j'ai dû fixer. Ceci était dû à certaines fonctions qui n'avaient plus le même nom ou qui étaient dépréciées. La mise à jour avec Unity 2020.1 a aussi créé des conflits mais cela a été corrigé plutôt rapidement.

L'un des bugs que j'ai eu était un problème de cache sur la connexion. Ce problème avait lieu car je ne pouvais pas charger une liste de quizz via l'API car la liste n'avait pas changé dans la base. Du coup l'application me renvoyait une erreur 304 (Not modified qui indique qu'il n'y a pas besoin de retransmettre les ressources demandées). À cause de cette erreur, la liste ne s'affichait pas. Pour corriger cette erreur, j'ai dû cibler le fichier qui posait problème en mettant des debug points. Une fois que j'avais le fichier, je devais trouver un moyen de ne pas contrôler le cache et le moyen était de définir le contrôle du cache.

Voici la modification apportée à l'un des bugs :

```
HttpWebRequest req = (HttpWebRequest)WebRequest.Create(url);
req.CachePolicy = new HttpRequestCachePolicy(HttpRequestCacheLevel.NoCacheNoStore);
req.Accept = "text/xml,text/plain,text/html,application/json";
/*req.Headers.Add("Cache-Control", "no-cache, no-store, must-revalidate");
req.Headers.Add("Pragma", "no-cache");
req.Headers.Add("Expires", "0"); */
req.Method = "GET";
req.Timeout = 2000;


if (GameManager.Instance.GetApiManager().HasToHaveToken())
{
    string postData = null; // Workaround to pass the ref postData (this make no sense in GET request) --> Has to be changed later
    SetTokenAccordingToEmplacement(ref req, "GET", ref postData, GameManager.Instance.GetApiManager().GetTokenttpEmplacement());
}

HttpWebResponse result = (HttpWebResponse)req.GetResponse();


Stream ReceiveStream = result.GetResponseStream();
StreamReader reader = new StreamReader(ReceiveStream, System.Text.Encoding.UTF8);
return reader.ReadToEnd();
```

Les sources :

StackOverflow : https://stackoverflow.com/questions/1366848/httpwebrequest-getresponse-throws-webexception-on-http-304/1366869#1366869

La documentation Microsoft sur le WebRequest : https://docs.microsoft.com/en-us/dotnet/api/system.net.webrequest?view=net-5.0

La documentation sur le Cache-Control par Mozilla : https://developer.mozilla.org/fr/docs/Web/HTTP/Headers/Cache-Control

La documentation de Unity.

Conclusion

La reprise du projet fût plus complexe que prévue. J'ai passé beaucoup de temps à comprendre le code et la documentation. J'ai eu pas mal de problèmes à corriger dans le code de base qui m'a aussi pris pas mal de temps. Le confinement n'a pas aidé non plus car le dernier commit n'était pas passé et j'ai dû faire le trajet + réinstallation de Unity.

Au final, l'application tourne sur mobile (version Android) avec l'affichage des quiz, possibilité de participer et répondre aux questions et l'affichage du résultat. Scanner un QR Code fonctionne pour charger un quiz fonctionne aussi.