# Application de réalité augmentée

## Introduction

Dans l’optique d’apporter une nouvelle façon de visualiser les données d’une entreprise. Mr Tripied m’a proposé de réaliser une application de réalité augmentée pour afficher les graphiques réalisés sous power bi.

L’application en plus d’afficher différents graphiques. Devras les afficher en fonction des coordonnées GPS retourné par le smartphone de l’utilisateur. Cette fonctionnalité permet d’afficher des données très précises et cohérente avec l’endroit où se trouve l’utilisateur.

## Recherche

Avant de commencer à développer une quelconque application, j’ai réalisé plusieurs recherches pour étudier la faisabilité du projet. Le premier point sur lequel j’ai travaillé, a été power bi. Il fallait savoir s’il était possible d’afficher un rapport, ou n’importe quel graphique. La démarche complète et les résultats seront détaillés plus tard.

Une fois solution trouver, j’ai pu m’intéresser à la partie réalité augmentée. Sur ce morceau de la solution, les choix on était plus simple. Comme je connaissais déjà pas mal de framework qui permettent de réaliser des applications de réalité augmentées.

Power BI

Comme expliqué plus haut, la première chose que j’ai fait, a été de réaliser des recherches pour savoir s’il était possible d’afficher des rapports power bi dans une application extérieur.

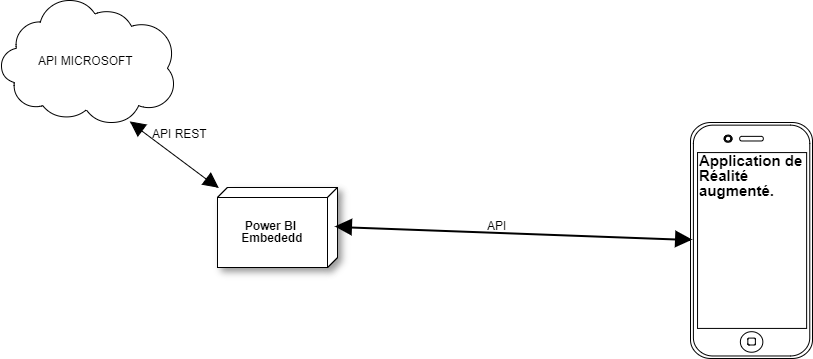
À la suite des recherches, nous découvrons une api de Microsoft, tout simplement nommé Power BI embedded. Cette api permet de récupérer n’importe quel rapport, diagramme d’un rapport sous forme d’une iframe, qui peut donc être affiché très simplement dans une application.

Réalité augmentée

Ayant déjà réalisé des applications de réalité augmentées à l’aide de Unity, il a été plus simple d’identifier les possibles problèmes. La première question était de savoir avec quelle technologie l’application serait développée. Je me suis naturellement tournée vers Unity, car c’est une des technologies que je maîtrise le mieux. Et que Unity permet de réaliser des applications de réalité augmentées très simplement. La seule chose qui a dû être vérifié, été si Unity était possible d’afficher une iframe dans une scène.

## Travaux réalisés

Une fois les recherches finis, et les différents choix concernant les technologies utilisaient etc… J’ai réalisé des diagrammes et notamment des diagrammes de flux. C’est diagramme nous permettent de représenter le fonctionnement de l’application.



Comme on peut le voir dans le diagramme ci-dessus l’application se séparent en trois grandes parties différentes. La première est la partie power bi embedded, cette solution sera sur un serveur externe. Qui va permettre de décharger le smartphone de cette opération et donc d’avoir de meilleur temps de réponse.

La deuxième partie est le côté de l’application qui sera sur les smartphones. Cette partie sera donc en charge d’afficher les rapports power bi en réalité augmentée.

Et enfin la dernière partie de l’application permet de faire le lien entre la solution power bi embedded et la solution sur smartphone. Elle sera représentée par une API, qui sera donc appelé par la solution de réalité augmentée sur smartphone et renverra une iframe contenant un rapport entier ou simplement un graphique.

Power Bi

La solution permettant de récupérer les iframes des différents rapports, fait appel à une api développez par Microsoft. Cette api peut être implémenté en .net, node.js, java, php etc…  
Pour réaliser une application la plus simple à maintenir, l’application a été réalisée en C# .net.

Ce choix se justifie simplement, l’api étant développé par Microsoft elle est entièrement compatible avec .net et donc le C#, mais également car Unity3D utilise le C# comme langage de programmation. Et ainsi dans le futur les deux solutions pourront être réunies très facilement.

### Etape 1

La première étape dans la création de cette solution est d’enregistrer notre application dans une Azure AD. Cela a pour but de nous fournir deux tokens d’identification. C’est token nous servirons pour s’authentifier lors de la récupération des rapports. Pour enregistrer notre application rien de plus simple il faut se rendre sur le site web suivant : <https://dev.powerbi.com/apps>  
Une fois sur ce site, il vous suffit de renseigner le nom de l’application, ainsi que l’url de redirection (si vous travaillez en local avec IIS ou wamp/xamp, l’url seras sous la forme suivante : localhost :1234).

### Etape 2

Cette étape consiste à donner à s’authentifier au prêt de l’Azure AD, au quel nous avons enregistré notre application. Pour réaliser cela nous devons envois les paramètre suivant :

var @params = new NameValueCollection

{

{"response\_type", "code"},

//Un des deux token que l’on as récupérer en enregistrant l’application.

{"client\_id", Properties.Settings.Default.ClientID},

// https://analysis.windows.net/powerbi/api

{"resource", Properties.Settings.Default.PowerBiAPI},

//adresse de redirection

{"redirect\_uri" , "http://localhost:13526/Redirect"}

};

C’est paramètres sont envoyés en POST à l’adresse suivante : <https://login.windows.net/common/oauth2/authorize/>.  
Une fois la réponse obtenue par l’api de Microsoft, vous serez redirigé vers l’adresse de redirection que vous avez indiquée lorsque vous avez enregistré votre application à power bi embedded.

### Etape 3

Maintenant que nous avons récupérer les autorisations dont nous avions besoin, nous pouvons commencer à récupérer les dashboards accessibles avec le compte qui nous à permit de se connecter au service power bi embedded. Pour cela nous devons dans un premier temps configuré une requête web. Cette requête à pour adresse : <https://analysis.windows.net/powerbi/api/dashboard> . Ce qui vas permettre à l’api de nous renvoyer une list JSON avec tous les dashboards dont l’utilisateur à accès. Ce fichier JSON est constitué de deux informations un ID unique pour chaque dashboard et leurs noms.   
Pour ce qui est des rapports et les graphiques, la manipulation est identique seule l’adresse change.

Url Rapport : <https://api.powerbi.com/v1.0/myorg/reports>  
Url Graphique(Tiles) : <https://api.powerbi.com/v1.0/myorg/dashboards/id/tiles> (l’id correspond à l’id du Dashboard que l’on souhaite utiliser.)

### Etape 4

Maintenant que nous avons une liste des rapports et graphique que l’on veut, nous devons les afficher. Pour cela nous allons créer un script javascript (ou si vous utilisez ASP.net juste une nouvelle méthode) et y utiliser la méthode **powerbi.embed()** qui prend deux paramètre, le premier est le conteneur HTML qui va accueillir notre rapport et le deuxième est une liste de paramètre qui se présente sous cette forme en js :

var config= {

    type: 'report',

    tokenType: tokenType == '0' ? models.TokenType.Aad : models.TokenType.Embed,

    accessToken: txtAccessToken,

    embedUrl: txtEmbedUrl,

    id: txtEmbedReportId,

    permissions: permissions,

    settings: {

        filterPaneEnabled: true,

        navContentPaneEnabled: true

    }};

Une fois la requêtes effectuée la méthode powerbi.embed() vas intégrer directement dans votre page HTML votre rapport.

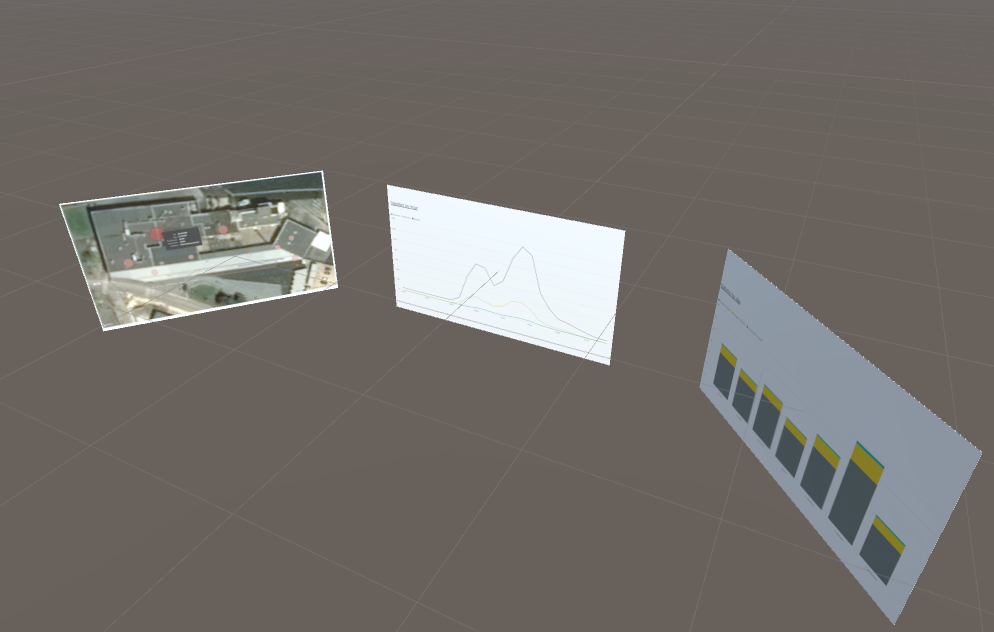
Réalité augmentée

La solution de réalité augmentée a donc pour fonction d’afficher n’importe quel rapports ou/et graphiques power bi. Pour cela la solution est constituée de deux parties distinctes. La première permet de gérer les rapports à afficher dans l’espace. Pour cela l’application vas dans un premier temps réaliser un arrière-plan avec la vidéo renvoyée par la caméra du smartphone. Cela permet de créer l’environnement de réalité augmentée.

  
Une fois l’environnement créé, nous devons encore afficher les rapports. Pour cela il a fallu définir l’endroit où instancier un rapport. Nous avons décidé très simplement de l’instancier en face du smartphone. Cela nous permet de se retrouver avec un visuel comme cela :



La deuxième partie de l’application de réalité augmentée est la partie qui gère les rapports power bi une fois qu’ils sont affichés. Pour cela nous avons décidé de les gérer sous forme d’un menu constitué de plusieurs tuiles. Ces tuiles contiennent les rapports ou graphiques dont nous avons besoin.

  
Le menu peut contenir n’importe quelle type d’objets et peut également être constitué d’autant de tuiles que l’on désire.