# Projet Smart City IG 2016-2017

Consignes de réalisation

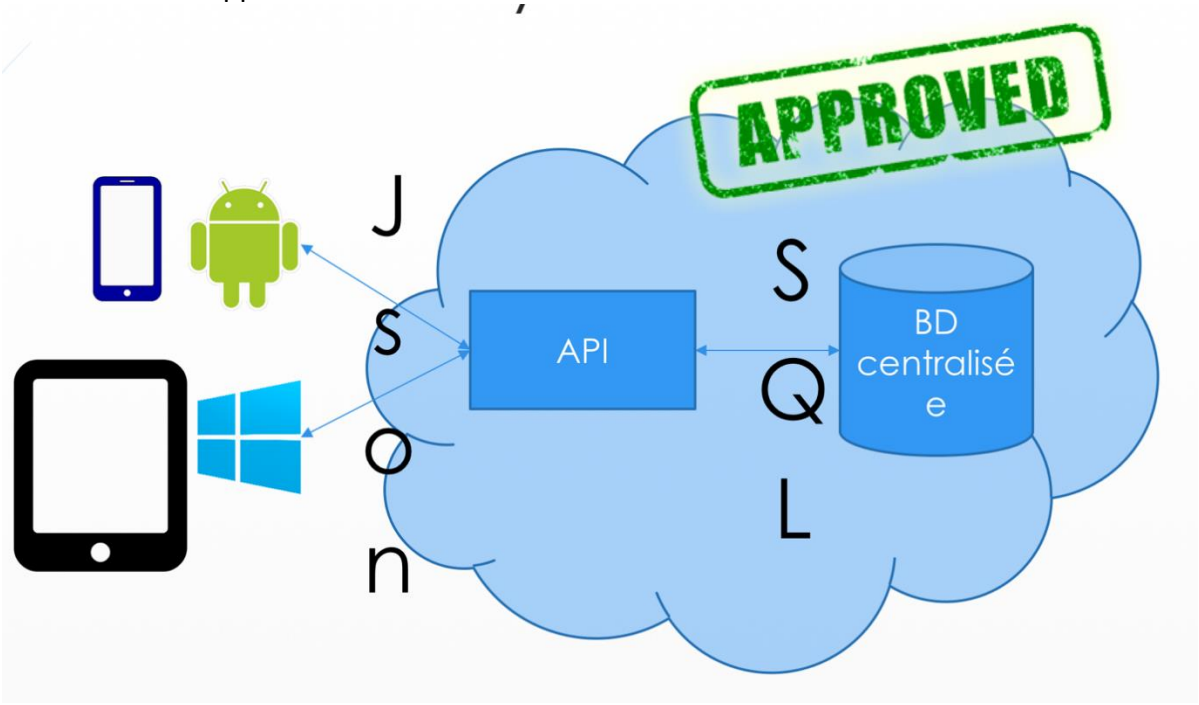
## Introduction

Ce document décrit les exigences que vous devrez respecter dans la réalisation de votre projet. Ces exigences sont à observer pour les cours suivants :

- Environnement de développement de logiciels
- Programmation mobile
- BD avancées et applications Web

## Architecture de la solution

L'architecture de votre projet est imposée. Elle vous a été présentée durant les sessions de BD avancées et applications Web.




Vous devrez concevoir :

- Un client UWP (voir cours Environnement de développement de logiciels)
- Un client Android (voir cours Programmation mobile)
- Une API basée sur ASP.NET Web API (voir cours BD avancées et applications Web)
- Une base de données (voir cours BD avancées et applications Web)

### Exigences relatives à l'API

Vos applications mobiles seront les plus minces possibles. La logique métier doit être contenue dans l'API.
L'API doit revérifier toutes les entrées soumises par les clients (l'API ne fait pas confiance au contenu envoyé par les clients). Voir support de cours.

L'API (et sa couche d'accès aux données) doit implémenter les qualités d'une couche d'accès aux données telles que vues durant les cours de BD avancées et applications Web.
Vous pouvez écrire votre couche d'accès aux données à l'aide d'un ORM, vous n'y êtes pas obligés.
L'API observera le style architectural REST.

## Exigences de l'application UWP

L'application sera de type native (C#/Xaml). Pas d'utilisation de frameworks cross-platforms tels que Xamarin, Phonegap, Cordova…

L'application devra implémenter le pattern MVVM (Data Binding, Commands, le moins de code-behind possible).

Elle interagira avec l'API par http et sérialisera/désérialisera le contenu échangé en Json.
Vous ne devez pas cibler la plateforme mobile (Windows Phone), n'essayez donc pas de faire tourner le nécessaire dans l'émulateur (vous perdriez votre temps). Contentez-vous de faire tourner l'application en « Local Machine ».

Vous aurez une solution Visual Studio qui contiendra tous les projets nécessaires, à savoir (au minimum) :

- Le projet ASP.NET web API
- Le projet UWP

Vous devrez utiliser GIT. Comme déjà mentionné, votre projet doit être taggé sous Git au plus tard le jour de la présentation (17 décembre, 23 :59). Nous devrons pouvoir visualiser dans l'historique de votre repository que vous avez utilisé ce dernier pour supporter votre travail (activité régulière).

## Exigences de l'application Android

L'application Android doit proposer les différentes fonctionnalités de votre projet Smart City. Cette application doit pouvoir s'exécuter au minimum sur un téléphone Android (aucun autre support imposé).

Les deux types d'orientation (paysage/portrait) doivent être gérés pour au moins deux écrans. Par conséquent, vous devrez prévoir des layouts différents pour l'orientation portrait et l'orientation paysage, et ce, pour au moins deux écrans.

Les accès à l'API doivent impérativement être gérés dans des tâches asynchrones.

Une attention particulière sera apportée à l'architecture de l'application (découpe en couches …). Par conséquent, n'utilisez pas la librairie Volley pour les accès à l'API car il y aurait couplage des couches, la même classe gérant à la fois des composants de la vue et l'accès aux données.

### Exigences générales relative à l'organisation de votre code

Votre code devra observer les principes d'organisation abordés durant tout votre cursus : découpe en couche, clean code, pas de duplication,  ...

### Exigences relatives à l'application et ses fonctionnalités

La validation des différents formulaires est à envisager (données obligatoires, format).
L'affichage des erreurs doit être significatif (doublons, donnée manquante…)
Vous devez gérer les requêtes http se terminant en erreur (ex : requête invalide, erreur de connexion à app-API, erreur de connexion API-DB…)
Vérifiez que l'utilisateur peut revenir en arrière s'il le souhaite (présence de boutons back pour revenir en arrière dans la partie Windows.

### Exigences relatives à la base de données

Elle sera hébergée dans le cloud ou sur le serveur SQL de l'IESN, en fonction de l'état d'avancement d'obtention des accès Azure. Le mode d'hébergement retenu vous sera communiqué ultérieurement.
Elle sera de type SQL Server
Les qualités de conception abordées durant tout votre cursus sont à respecter.

## Exigences méthodologiques

Votre projet doit porter un nom. Définissez-le assez tôt, car vous en aurez besoin rapidement (nommage de projets, namespaces, assemblies…) Impliquez vos partenaires de Marketing afin de vous aider à trouver des idées.

## Critères de cotation

La cotation est basée d'une part sur un travail de groupe et d'autre part sur un examen oral. L'évaluation du travail de groupe constituera une cote de base pour l'examen oral ; cette cote pourra être sensiblement augmentée ou diminuée en fonction des réponses données par l'étudiant aux questions posées lors de l'oral.
Lors de l'examen oral, l'étudiant devra être capable d'expliquer le code de l'application réalisée en groupe.

A l'examen oral, le projet sera présenté aux professeurs (démonstration du programme). Pensez à nous expliquer le domaine d'application, les différentes fonctionnalités, une critique de l'application existante et des suggestions d'améliorations possibles. Nous n'imposons pas de slides à ce sujet mais une présentation de projet se prépare.

Ensuite, chaque membre du groupe sera interrogé sur des points précis aussi bien théoriques que pratiques. La note de base du travail pourra être transformée en un échec si

l'étudiant est incapable de répondre aux questions posées lors de l'oral, tant au niveau théorique que pratique. Nous supposerons également que vous savez effectuer une manipulation sur GitHub. Pour l'examen, toute la matière doit être revue et maîtrisée (code smells, …), en ce compris les concepts de programmation et concepts O.O. de base. Vous devrez prouver votre intégration de ces concepts.

Une **copie sur CD** sera remise aux professeurs lors de la présentation du projet.
Ce CD sera nommé de la manière suivante :
IG3 – 2016-2017 – Smart Cities - Nom et prénom des deux membres du groupe.

Le CD doit comprendre la solution Visual Studio ou l'application Android

## Remarques

Vous allez devoir chercher beaucoup d'informations par vous-même. Votre autonomie est importante. Afin de vous mettre sur la piste, nous avons constitué une wiki qui grandit régulièrement et couvre des thèmes qui devront être abordés par plusieurs groupes. Vous pouvez faire grandir cette wiki si vous le désirez (ajout d'articles, commentaires…). Il vous est possible de demander un accès en écriture (certains l'ont déjà fait). Rappel de son URL : http://ighenallux.pbworks.com/w/page/111908059/FrontPage

Commencez par les points les plus risqués de votre système. Ex : une fonctionnalité dont vous n'êtes pas certains de la faisabilité technique, une tâche qui semblerait-il sera chronophage…

Essayez d'avoir accès à des données existantes. Envoyez des e-mails, documentez-vous sur les Open Data disponibles (http://data.gov.be/fr, http://opendata.bep.be/, http://opendata.bruxelles.be/page/home/…)
Ne vous bornez pas aux données disponibles à Namur, inspirez-vous de ce qui se fait ailleurs (et parlez-en à vos partenaires de Marketing afin qu'ils intègrent ça à la démarche de promotion). Si vous n'avez aucune donnée existante ou si vous les prenez ailleurs que dans le périmètre namurois, nous ne vous bloquerons pas ; de même si vous ne trouvez rien, vous les inventerez. Si l'idée plaît, les moyens permettant de mettre en œuvre la collecte/publication des données nécessaires seront peut-être débloqués.