**Présentation d’une architecture multimodule (QuizzTionMe)**

Notre application QuizzTionMe possède plusieurs fonctionnalités et se veut évolutive au cours du temps. De ce fait, nous avons opté pour un projet multimodule qui permet de séparer clairement les différentes parties de l’application en plusieurs modules. Chaque module de l’application est :

* Distinct
* Facile à maintenir
* Souple (ajouter des fonctionnalités)
* Réutilisable
* Testable individuellement (par exemple, le fait de tester des modules plus petits peut réduire le nombre de bug car ils peuvent être localisés avec leur module spécifique)

Un module est un sous-projet rattaché à un projet parent et grâce à maven, on peut gérer dépendances entre eux.

**Problématiques**

Un projet multimodule deviens très difficile à maintenir si les modules ne sont pas bien documentés, de facto, gérer les dépendances devient fastidieux.

Autre problématique est que l’architecture multimodule pose un problème car chaque module est dans son package, or SpringBoot scanne seulement les composants du module contenant le point d’entrée (la classe annoté @SpringBootApplication) de l’application et ses enfants. Donc, spring n’analysera pas ceux déclarés dans les autres modules et pour résoudre le problème il faut préciser à SpringBoot les emplacements à analyser via les annotations @ComponentScan(ScanBasepackageClasses= {« nomdupackage« }), @EnableJpaRepository() , @EntityScan()

**Illustration du schema de l’architecture (Figure1)**

L’application est divisée en plusieurs couches :

* La couche domain : est constitué d’objets métier
* La couche data-access : chargée de l’accès aux objets métier et de leur manipulation indépendamment du SGBD choisi.
* La couche bussiness-logic : contient les traitements représentant les règles métier
* La couche web-rest : chargée d’offrir des web services utilisables par des applications tierces

