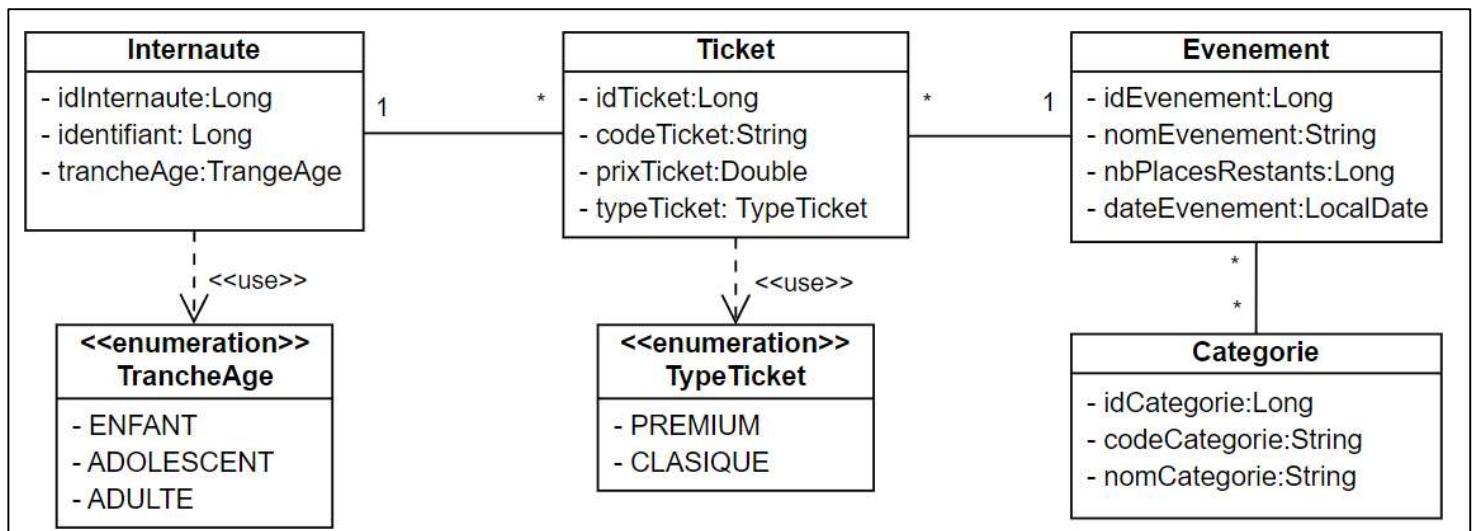


## TP2 Microservices

On vous propose d'implémenter une application de vente de tickets d'évènements en ligne. Un internaute (utilisateur d'internet) peut accéder aux tickets des différents événements. Ci-dessous le diagramme des classes :



### I.1. Entités/associations :

Implémenter les entités qui permettent de générer le schéma de la base de données comme illustré dans le diagramme de classes sachant que :

- Les identifiants sont auto-générés avec la stratégie « **IDENTITY** ».
- Les énumérations **TrancheAge** et **TypeTicket** doivent être stockée en tant que chaînes de caractères dans la base de données.

- La relation bidirectionnelle **Internaute** --- **Ticket** modélise le fait qu'un internaute peut acheter plusieurs tickets et qu'un ticket concerne un seul internaute.
- La relation bidirectionnelle **Ticket** --- **Evenement** modélise le fait qu'un ticket est associé à un seul événement et qu'un événement offre plusieurs tickets.
- La relation bidirectionnelle **Evenement** --- **Categorie** modélise le fait qu'un événement peut être associée à plusieurs catégories et qu'une catégorie peut concerner plusieurs événements (**Categorie** est le **Child**).

## I.2. Services :

Développer le code nécessaire dans une classe annotée par **@RestController** qui fait appel aux différents services. (Exposition des services avec Spring REST MVC, et Tests avec **Postman** ou **Swagger**). Voici les Services demandés :

- A. En respectant la signature de la méthode suivante, ajouter les internautes ci-dessous :

**Internaute ajouterInternaute(Internaute internaute);**

	Identifiant	trancheAge
<b>Internaute</b>	Salim souissi	ADOLESCENT
	Aziza Salhi	ADULTE

- B. En respectant la signature de la méthode suivante, ajouter les événements ci-dessous avec les catégories associées (L'événement 'summer vibes' avec les catégories associées 'c1' et 'c2' et l'évènement 'how to get a job in one week' avec la catégorie associée 'c3') :

**Evenement ajouterEvenement(Evenement evenement);**

	<b>nomEvenement</b>	<b>nbPlacesRestantes</b>	<b>DateEvenement</b>
<b>Evenement</b>	summer vibes	2	2024-08-10
	<b>Categorie</b>	<b>codeCategorie</b>	<b>nomCategorie</b>
		c1	Divertissement
<b>Evenement</b>		c2	Loisir
	<b>nomEvenement</b>	<b>nbPlacesRestantes</b>	<b>DateEvenement</b>
	how to get a job in one week	1	2024-09-02
<b>Categorie</b>	<b>Categorie</b>	<b>codeCategorie</b>	<b>nomCategorie</b>
		c3	Professionnel

- C. En utilisant **Spring Scheduler**, proposer une méthode qui se déclenche **toutes les 15 secondes** et qui affiche sur la console la liste des événements pour chaque catégorie comme illustré dans l'exemple ci-dessous :

```
void listeEvenementsParCategorie();
```

INFO 20024 --- [ scheduling-1] tn.esprit.spring.Services.ServiceClass : Categorie Divertissement
INFO 20024 --- [ scheduling-1] tn.esprit.spring.Services.ServiceClass : Evenement summer vibes planifié le 2024-08-10
INFO 20024 --- [ scheduling-1] tn.esprit.spring.Services.ServiceClass : Categorie Loisir
INFO 20024 --- [ scheduling-1] tn.esprit.spring.Services.ServiceClass : Evenement summer vibes planifié le 2024-08-10
INFO 20024 --- [ scheduling-1] tn.esprit.spring.Services.ServiceClass : Categorie Professionnel
INFO 20024 --- [ scheduling-1] tn.esprit.spring.Services.ServiceClass : Evenement how to get an it job in one week planifié le 2024-09-02

- D. En respectant la signature de la méthode suivante, ajouter les tickets ci-dessous et les affecter aux événements et aux internautes indiqués dans le tableau **en mettant à jour** à chaque fois **le nombre de places restantes** dans la table « Evenement » :

**NB :** Si le nombre de tickets à ajouter dépasse le nombre de places restantes dans un événement, aucun ticket ne peut être sauvegardé dans la base de données et une exception sera déclenchée comme suit :

```
throw new java.lang.UnsupportedOperationException("nombre de places demandées
indisponible")
```

Le déclenchement de l'exception sur la console :

```
java.lang.UnsupportedOperationException Create breakpoint : nombre de places demandées indisponible
at tn.esprit.spring.Services.ServiceClass.ajouterTicketsEtAffecterAEvenementEtInternaute(Se
at tn.esprit.spring.RestControllers.RestControllerClass.ajouterTicketsEtAffecterAEvenementE
at jakarta.servlet.http.HttpServlet.service(HttpServlet.java:590) ~[tomcat-embed-core-10.1.
at jakarta.servlet.http.HttpServlet.service(HttpServlet.java:658) ~[tomcat-embed-core-10.1.
```

**List<Ticket> ajouterTicketsEtAffecterAEvenementEtInternaute(List<Ticket> tickets, Long idEvenement, Long idInternaute );**

	codeTicket	prixTicket	typeTicket	Evenement	Internaute
Ticket	sv1	35	CLASIQUE	summer vibes	Salim souissi
	sv2	35	CLASIQUE		
	tick1	10	CLASIQUE	how to get a job in one week	Aziza Salhi

E. En respectant la signature de la méthode suivante, afficher le montant à récupérer pour un événement donné selon le type du ticket :

**Double montantRecupereParEvtEtTypeTicket(String nomEvt, TypeTicket typeTicket);**

F. En respectant la signature de la méthode suivante, afficher l'identifiant de l'internaute le plus actif (celui qui a acheté le plus de tickets) :

**String internauteLePlusActif();**

G. En utilisant **Spring AOP**, implémenter un aspect qui permet d'afficher le message « Le nombre de places restantes dépasse le nombre de tickets demandés » si la méthode ajouterTicketsEtAffecterAEvenementEtInternaute de la question « D » retourne une exception