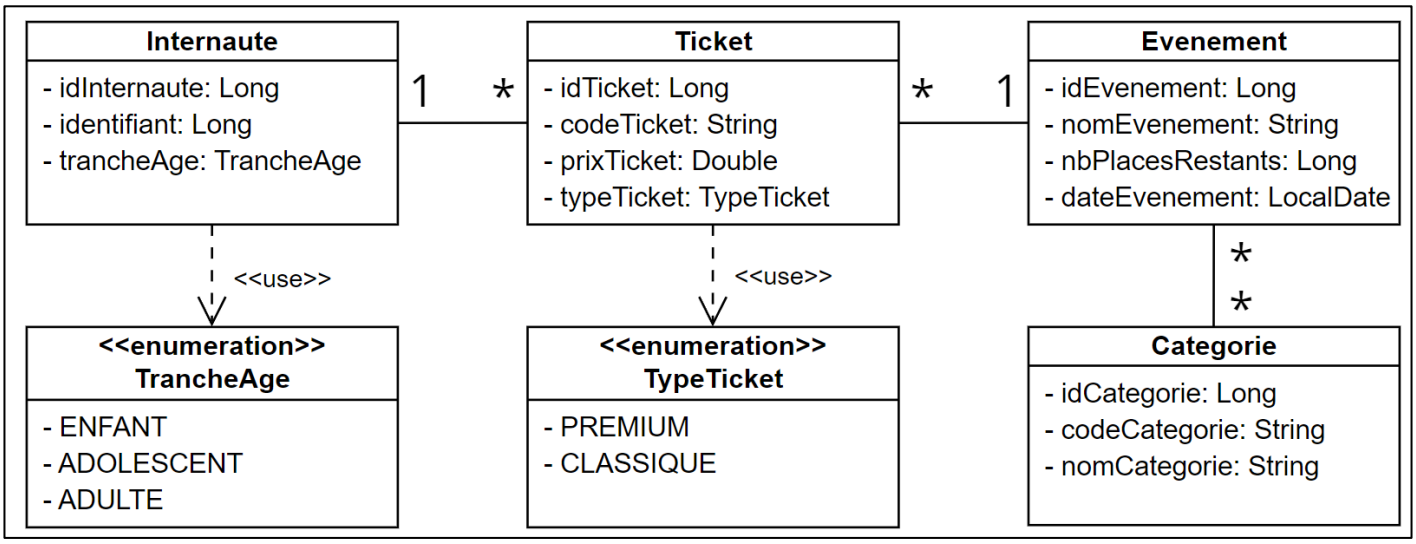
esprit Se former autrement HONORIS UNITED UNIVERSITIES	Semestre : 1 ■ 2 □ Session : Principale ■ Rattrapage □

Module : **ARCHITECTURE DES SI II (FRAMEWORK SPRING)**

Enseignants : **Équipe Spring**	Classes : **4ème SAE**	Nombre de pages : **4 pages**
Documents autorisés : **OUI**	Calculatrice autorisée : **NON**	Internet autorisés : **NON**
Date : **08/01/2024**	Heure : **09h00**	Durée : **01h30min**

NB : La validation de l'épreuve est appliquée sur la base d'un code source exécutable. Aucun code source non fonctionnel n'est comptabilisé lors de la validation

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 (6 points) :

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 évènement offre plusieurs tickets.
- La relation bidirectionnelle **Evenement** --- **Categorie** modélise le fait qu'un événement peut être associé à plusieurs catégories et qu'une catégorie peut concerner plusieurs événements **(Categorie est le child)**.

I.2. Services (14 points) :

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 **(1 point)** :

Internaute ajouterInternaute (Internaute internaute) ;

Internaute	**identifiant**	**trancheAge**
	Salim souissi	ADOLESCENT
	Aziza Salhi	ADULTE

B. En respectant la signature de la méthode suivante, ajouter les événements ci-dessous avec les catégories associées **(2 points) :**
   - ✓ L'événement 'summer vibes' avec les catégories associées 'c1' et 'c2'.
   - ✓ L'événement 'how to get a job in one week' avec la catégorie associée 'c3'.

Evenement ajouterEvenement(Evenement evenement);

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

C. En utilisant **Spring Scheduler**, proposer une méthode qui se déclenche **toutes les 15 secondes** et qui affiche sur la console, avec **l'outil de journalisation SLF4J**, la liste des événements pour chaque catégorie comme illustré dans l'exemple ci-dessous **(1.5 points)** :

void listeEvenementsParCategorie() ;

```
tn.esprit.spring.Services.ServiceClass   : Categorie Divertissement
tn.esprit.spring.Services.ServiceClass   : Evenement summer vibes planifié le 2024-08-10
tn.esprit.spring.Services.ServiceClass   : Categorie Loisir
tn.esprit.spring.Services.ServiceClass   : Evenement summer vibes planifié le 2024-08-10
tn.esprit.spring.Services.ServiceClass   : Categorie Professionnel
tn.esprit.spring.Services.ServiceClass   : Evenement how to get a 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 » **(2.5 points)** :

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 indisponibe")

Le déclenchement de l'exception sur la console :

```
java.lang.UnsupportedOperationException Create breakpoint : nombre de places demandées indisponibe
    at tn.esprit.spring.Services.ServiceClass.ajouterTicketsEtAffecterAEvenementEtInternaute(Se
    at tn.esprit.spring.RestControllers.RestControllerClass.ajouterTicketsEtAffecterAEvenementE
```

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

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

E. 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 **(1.5 points)**.

Le message affiché sur la console :

```
tn.esprit.spring.Aspect.AspectClass      : Le nombre de places restantes dépasse le nombre de tickets demandés
```

F. En respectant la signature de la méthode suivante et **en utilisant KEYWORD**, Afficher le nombre d'internautes de la tranche d'âge « ADULTE » ayant participé à des événements entre « 2024-08-01 » et « 2024-10-01 » **(1.5 points)** :

Long nbInternauteParTrancheAgeEtDateEvenement(TrancheAge trancheAge, LocalDate dateMin, LocalDate dateMax);

G. En respectant la signature de la méthode suivante et **en utilisant JPQL**, afficher le montant à récupérer pour un événement donné selon le type du ticket **(2 points)** :

Double montantRecupereParEvtEtTypeTicket(String nomEvt,TypeTicket typeTicket);

H. 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) **(2 points)** :

String internauteLePlusActif();

Bon courage 😭😊