esprit
Se former autrement
HONORIS UNITED UNIVERSITIES

Architecture des SI II (Spring Framework)

Etude de cas gestion “Station de SKI”

ESPRIT - UP ASI

Année universitaire 2022/2023

EUR-ACE® Délivrée par la Commission des Titres d'Ingénieur

CONFÉRENCE DES GRANDES ÉCOLES

cdio

AUF Agence Universitaire de la Francophonie

IFGU

Etude de cas “Station de SKI”

Objectifs

Développer et déployer une application web d’entreprise en utilisant le Framework Spring.

Acquis d’apprentissage

AA1 : Identifier les différentes couches d’une architecture N-tiers

AA3 : Construire une application par l’intermédiaire d’un outil de gestion de projet

AA4 : Appliquer la notion de l’injection de dépendance

AA6 : Evaluer les différentes couches du projet Spring Data JPA

AA7 : Développer des services pour la manipulation des données

AA8 : Exposer des Web Services REST: Spring MVC REST

Etude de cas “Station de SKI” Énoncé

- Une station de ski offre à ses abonnées des cours de ski par une équipe de moniteurs.
- Chaque **piste** de la station est identifiée par un numéro et possède un nom de piste, et on connait sa couleur (qui indique la difficulté), ainsi que sa longueur en kilomètres et son pente (dénivelé).
- Des cours de ski sont proposés, identifiés par un numéro. Chaque **cours** (Collectif ou Particulier) est adapté à un niveau de ski (1ere étoile, 2eme étoile…), un support (par exemple ski, snowboard…) et il est associé à un créneau (1: matin ou 2: après-midi) et à un prix.
- Le moniteur associé à un cours est également enregistré.

Etude de cas “Station de SKI” Énoncé

- La participation à un cours nécessite une **inscription** de la part des skieurs, en précisant le numéro de la semaine concernée (entre 1 et 52).

- Les **skieurs** sont identifiés par un numéro, et on connaît leur nom, prénom, date de naissance et ville de résidence. Chaque skieur peut effectué un abonnement selon son choix (annuel, mensuel ou semestriel).

- Les **moniteurs** sont identifiés par un numéro, et on connaît leur nom, prénom et leur date de recrutement. Le diagramme de classes suivant, décrit la conception de notre étude de cas.

Etude de cas “Station de SKI” Énoncé

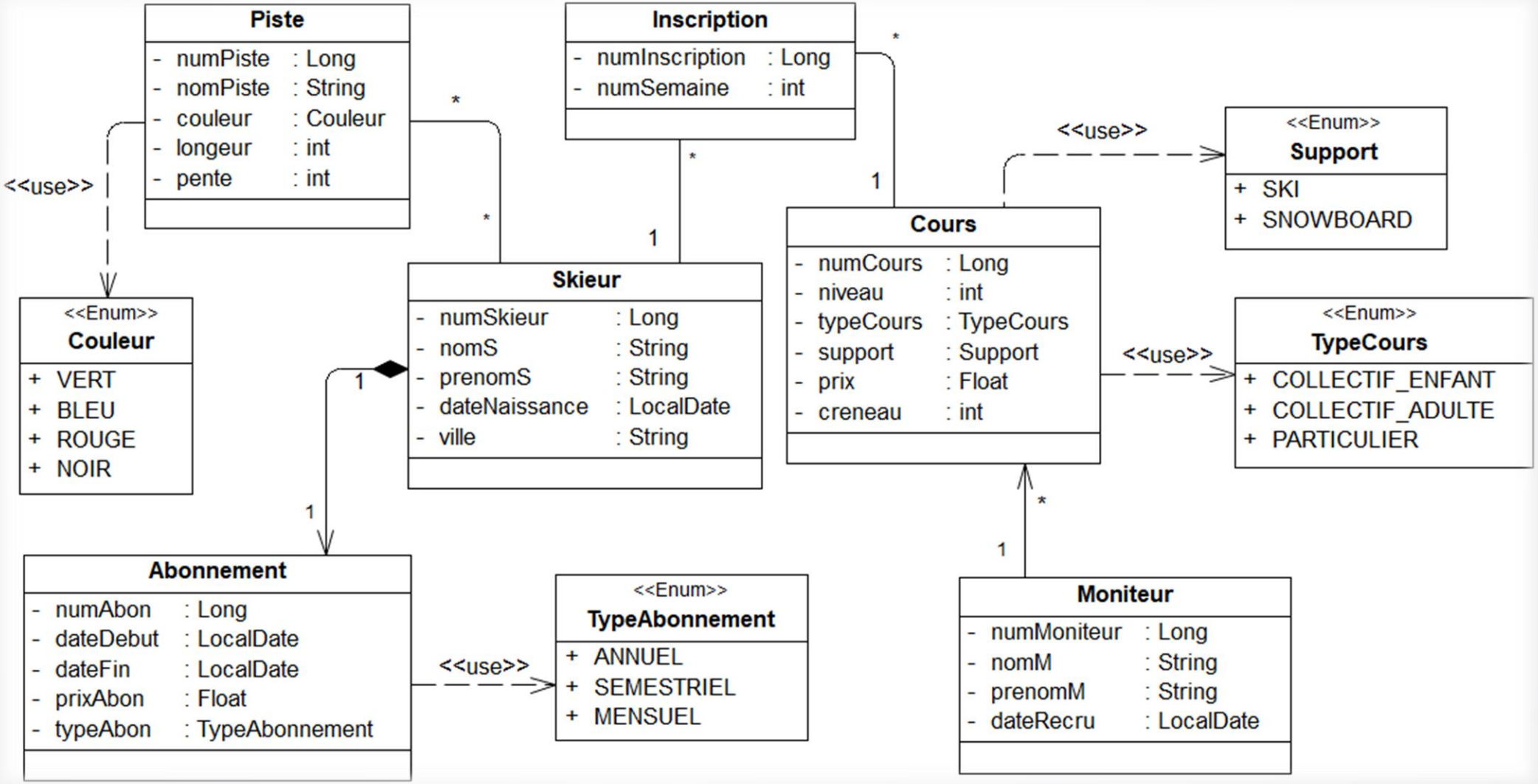

Figure 1. Diagramme de classes Etude de cas gestion “Station de SKI”

Etude de cas “Station de SKI”

Travail demandé

1. Implémentation du modèle entité - association
2. Création de la couche de persistance de données de chaque entité (CrudRepository, JPQL, …)
3. Implémentation de la couche métier (Service)
4. Exposition des services en utilisant des APIs REST : couche présentation (RestController)

N.B : Une bonne qualité du code est fortement recommandée

Etude de cas “Station de SKI”

Partie 1 Spring Data JPA – Première entité

- Créer les entités se trouvant dans le diagramme des classes (sans les associations) et vérifier qu’ils ont été ajoutés avec succès dans la base de données.

Etude de cas “Station de SKI”

Partie 2 Spring Data JPA – Associations

- Supprimer les tables existantes dans la base de données.
- Créer les associations entre les différentes entités.
- Générer la base de données de nouveau et vérifier que le nombre de tables créées et le mapping des associations sont corrects.

Etude de cas “Station de SKI”

Partie 3 Spring Data JPA CRUD Repository- Le langage JPQL - JPA Repository

- Créer les CRUD des différentes entités indiquées dans les slides suivants en respectant les signatures suivantes

Entité Piste

List<Piste> retrieveAllPistes();

Piste addPiste(Piste piste);

Piste updatePiste (Piste piste);

Piste retrievePiste (Long numPiste);

Etude de cas “Station de SKI”

Partie 3 Spring Data JPA CRUD Repository- Le langage JPQL - JPA Repository

- Créer les CRUD des différentes entités indiquées dans les slides suivants en respectant les signatures suivantes

Entité Cours

List<Cours> retrieveAllCourses();

Cours addCours(Cours cours);

Cours updateCours (Cours cours);

Cours retrieveCours (Long numCours);

Etude de cas “Station de SKI”

Partie 3 Spring Data JPA CRUD Repository- Le langage JPQL - JPA Repository

- Créer les CRUD des différentes entités indiquées dans les slides suivants en respectant les signatures suivantes

Entité Moniteur

List<Moniteur> retrieveAllMoniteurs();

Moniteur addMoniteur(Moniteur moniteur);

Moniteur updateMoniteur (Moniteur moniteur);

Moniteur retrieveMoniteur (Long numMoniteur);

Etude de cas “Station de SKI”

Partie 3 Spring Data JPA CRUD Repository- Le langage JPQL - JPA Repository

- Créer les CRUD des différentes entités indiquées dans les slides suivants en respectant les signatures suivantes

Entité Skieur

List<Skieur> retrieveAllSkieurs();

Skieur addSkieur(Skieur skieur);

void removeSkieur (Long numSkieur);

Skieur retrieveSkieur (Long numSkieur);

N.B : - Pour l’ajout de Skieur, il faut créer en même temps son abonnement (l’entité associée Abonnement)
- Pour la suppression de Skieur, il faut supprimer en même temps son abonnement

Etude de cas “Station de SKI”

Partie 4 Spring MVC

- Exposer les services implémentés dans la partie 3 avec Postman et/ou Swagger pour les tester.

Etude de cas “Station de SKI”

Partie 5 Services avancés

- On souhaite ajouter une nouvelle inscription et l’affecter à un Skieur donné.
- Créer un service permettant l’ajout d’une inscription et l’affectation à un skieur et exposer le en respectant la signature suivante :

Inscription addRegistrationAndAssignToSkier(Inscription inscription, Long numSkieur)

Etude de cas “Station de SKI”

Partie 5 Services avancés

- On désire affecter une Inscription donnée à un Cours.
- Créer un service permettant l’assignation d’une inscription à un cours et exposer le en respectant la signature suivante :

Inscription assignRegistrationToCourse(Long num Inscription, Long numCours)

Etude de cas “Station de SKI”

Partie 5 Services avancés

- On désire affecter un Skieur à une Piste.
- Créer un service permettant l’assignation d’un skieur à une piste et exposer le en respectant la signature suivante :

```
Skieur assignSkierToPiste(Long numSkieur, Long numPiste);
```

Etude de cas “Station de SKI”

Partie 5 Services avancés

- On désire ajouter un Moniteur et l’affecter à un Cours donné.
- Créer un service permettant l’ajout et l’affectation d’un moniteur à un cours et exposer le en respectant la signature suivante :

```
Moniteur addInstructorAndAssignToCourse(Moniteur moniteur, Long numCours);
```

Etude de cas “Station de SKI”

Partie 5 Services avancés

- On désire ajouter un Skieur et l’affecter à un cours donné :
  - Il faut créer en même temps son abonnement (l’entité associée Abonnement) et sa nouvelle inscription (l’entité associée Inscription) et l’affecté au cours donné.
  - **N.B :** L'ensemble des objets liés au Skieur seront inclus et encapsulés avec lui.
- Créer le service adéquat et exposer le en respectant la signature suivante :

Skieur addSkierAndAssignToCourse(Skieur skieur, Long numCours);

Etude de cas “Station de SKI”

Partie 5 Services avancés

- On souhaite récupérer les skieurs selon leur type d’abonnement.
- Créer un service permettant de lister les skieurs selon un type d’abonnement et exposer le en respectant la signature suivante :

```
List<Skieur> retrieveSkiersBySubscriptionType(TypeAbonnement typeAbonnement);
```

Etude de cas “Station de SKI”

Partie 5 Services avancés

- On souhaite récupérer la liste des abonnements selon un type d’abonnement spécifique, triée en fonction de leur date de début.
- Créer le service adéquat et exposer le en respectant la signature suivante :

Set<Subscription> getSubscriptionByType(TypeSubscription type)

Etude de cas “Station de SKI”

Partie 5 Services avancés

- On souhaite afficher les abonnements qui ont été créés entre deux dates données.
- Créer le service adéquat et exposer le en respectant la signature suivante :

```
List<Abonnement> retrieveSubscriptionsByDates(LocalDate startDate,
                                              LocalDate endDate);
```

Etude de cas “Station de SKI”

Partie 5 Services avancés

- On désire ajouter une inscription et l’affecter à un skieur et à un cours donnés.
  - **N.B :** L’inscription à un cours donné ne doit pas dépassé 6 Skieurs/Cours si le type de cours est Collectif (COLLECTIF_ENFANT ou COLLECTIF_ADULTE).
  - L'âge de Skieur est encore vérifié au moment de l’affectation.
- Créer le service adéquat et exposer le en respectant la signature suivante :

Inscription addRegistrationAndAssignToSkierAndCourse(Inscription inscription, Long numSkieur, Long numCours);

Etude de cas “Station de SKI”

Partie 5 Services avancés

- On souhaite afficher les numéros des semaines où un moniteur a donnée des cours selon un support donné.
- Créer le service adéquat et exposer le en respectant la signature suivante :

```
List<Integer> numWeeksCourseOfInstructorBySupport(Long numInstructor,
                                                  Support support);
```

Etude de cas “Station de SKI”

Partie 6 Spring Scheduler

- Nous souhaitons créer un service programmé automatiquement permettant d’avertir le responsable de la gestion des abonnements dont la date de fin est prévue pour les 7 prochains jours afin de vérifier s’il faut renouveler ou mettre fin aux abonnements des Skieurs concernés.
- Le résultat permet d’afficher le numéro de l’abonnement en question et les informations associées au skieur concerné (numSkieur, firstName et lastName).
- Créer le service programmé adéquat en respectant la signature suivante :

```
void retrieveSubscriptions()
```

Etude de cas “Station de SKI”

Partie 6 Spring Scheduler

- Nous souhaitons créer un service programmé automatiquement permettant d’avertir le responsable de la station de ski chaque mois de **revenu récurrents mensuel** .
- L’**MRR** (*monthly recurring revenue*) est un indicateur qui prédit les revenus mensuel, sur la base du nombre d’abonnements souscrits.
- Créer le service programmé adéquat en respectant la signature suivante :

```
public void showMonthlyRecurringRevenue()
```

Etude de cas “Station de SKI”

Si vous avez des questions, n'hésitez pas à nous contacter :

Département Informatique
UP Architecture des Systèmes d’Information (ASI)
Bureau E204