

RA-IL 2

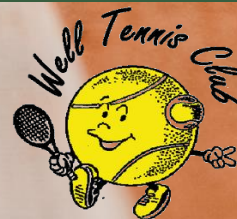
PRÉSENTATION

WELL TENNIS CLUB

KETZINGER MICHELI PEDRON

RAOUF

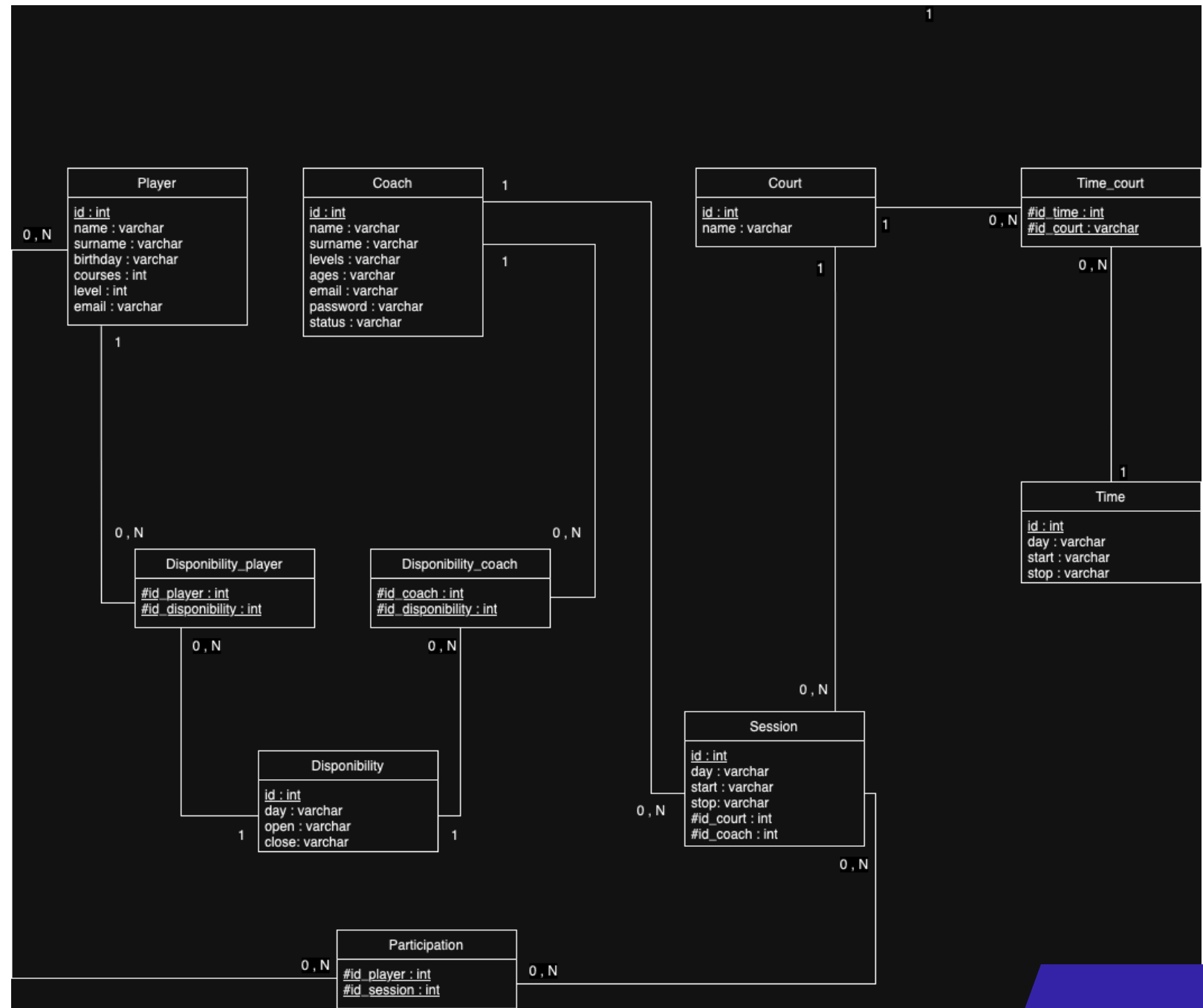
MICHAËL BORNE - OLIVIER PERRIN



Objectif de l'itération - 2

- Finalisation de la base de données.
- Intégration complète de la base de données avec l'API Spring Boot.
- Transition finale vers un projet Vue.js avec Tailwind.
- Création des pages d'inscription, de connexion et de réinitialisation de mot de passe.
- Mise en place du lien entre l'API et le Frontend.
- Implémentation des premières requêtes vers l'API.
- Test du fonctionnement de TimeFold.
- Intégration de TimeFold au projet avec ajout de contraintes.

Base de données



Création des tables par des scripts via liquibase

- 00-INIT-PLAYER.xml
- 01-INIT-COACH.xml
- 02-INIT-COURT.xml
- 03-INIT-DISPO.xml
- 04-INIT-DISPOJ.xml
- 05-INIT-DISPOC.xml
- 06-INIT-TIME.xml
- 07-INIT-TIMEC.xml
- 08-INIT-SESSION.xml
- 09-INIT-PARTICIPATION.xml

Exemple de la création de player

```
<createTable tableName="player">
  <column name="id" type="int" autoIncrement="true">
    <constraints primaryKey="true"/>
  </column>
  <column name="name" type="varchar(200)">
    <constraints nullable="false"/>
  </column>
  <column name="surname" type="varchar(200)">
    <constraints nullable="false"/>
  </column>
  <column name="birthday" type="varchar(200)">
    <constraints nullable="false"/>
  </column>
  <column name="courses" type="int">
    <constraints nullable="false"/>
  </column>
  <column name="level" type="int">
    <constraints nullable="false"/>
  </column>
  <column name="email" type="varchar(200)">
    <constraints nullable="false"/>
  </column>
</createTable>
```

**Accéder
individuellement
à chaque table**

time-controller

time-court-controller

session-controller

player-controller

participation-controller

disponibility-controller

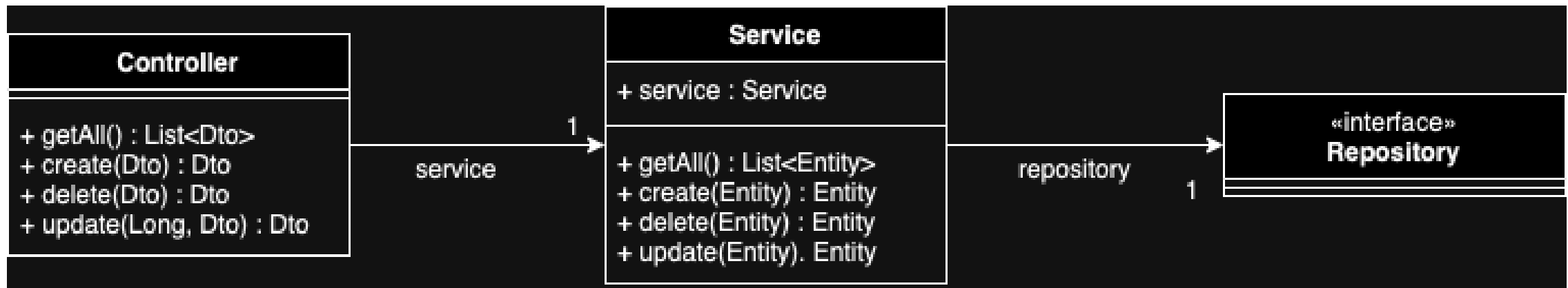
disponibility-player-controller

disponibility-coach-controller

court-controller

coach-controller

Exemple général de fonctionnement pour un objet de l'API



Répartition des différentes classes

```
java
└─ well_tennis_club.projet
   ├── coach
   ├── court
   ├── disponibility
   ├── disponibilityCoach
   ├── disponibilityPlayer
   ├── participation
   ├── player
   ├── session
   ├── time
   ├── timeCourt
   ├── ServletInitializer
   └─ WellTennisClubApplication
```

```
player
├── PlayerController
├── PlayerDto
├── PlayerEntity
├── PlayerMapper
├── PlayerRepository
└── PlayerService
```


Exemple pour la table player

player-controller

GET

/players Get all players

POST

/players Create player

GET

/players/{id} Get one player

DELETE

/players/{id} Delete player


PATCH

/players/{id} Update player

Exemple lorsqu'on récupère le joueur à l'ID 1

```
{ "id": 1,  
  "name": "Micheli", "surname": "Thomas",  
  "birthday": "2013-04-13",  
  "courses": 1,  
  "level": 6,  
  "email": "thomas@gmail.com",  
  "disponibilities": [{  
    "id": 1,  
    "day": "Monday",  
    "open": "8:00",  
    "close": "10:00"  
  }]  
}
```

Plateforme Web – Page Réinitialisation Password



Well Tennis Club

Entrez votre adresse e-mail pour recevoir un lien de réinitialisation de mot de passe.

Envoyer un mail de récupération

[Retour à la page d'accueil](#)

Plateforme Web – Page Administrateur

Données

Contraintes

↑ Importer vos données

Planning - format XLS

Données et contraintes - format CSV

↓ Télécharger vos données

Planning - format XLS

Données et contraintes - format CSV

Nouvelle année

Envoyer le mail de réinscription

Supprimer l'ensemble des joueurs

Terrain 1

Terrain 2

Terrain 3

Filtrer la recherche

Lundi

17:00

18:30

Entraîneur : Billie Sharpe

Âge : 15-18 ans

Niveau : 8-10

Joueurs :

Erica Scott

Alexandra Vance

Anais Ayala

Supprimer

Mardi

19:00

20:30

Entraîneur : Athena Garrett

Âge : 8-12 ans

Niveau : 5-7

Joueurs :

Callan McConnell

Kayla Dotson

Supprimer

Mercredi

10:00

11:30

Entraîneur : Billie Sharpe

Âge : 10-14 ans

Niveau : 6-8

Joueurs :

John Doe

Jane Doe

Kayla Dotson

Jane Doe

Kayla Dotson

Supprimer

10:00

11:30

Entraîneur : Billie Sharpe

Âge : 10-14 ans

Niveau : 6-8

Joueurs :

John Doe

Jane Doe

Supprimer

Plateforme Web – Page Inscription



Inscription

Nom

Prénom

Âge

Nombre de cours (max 3)

Disponibilité

Choisissez un jour



--:--



--:--




Ajouter

Email

S'inscrire

Plateforme Web – Page Inscription



Inscription

Nom

Prénom

Âge

Nombre de cours (max 3)

Disponibilité

Choisissez un jour

--:--

--:--

Ajouter

Lundi: 16:00 - 16:30

Supprimer

Mardi: 17:00 - 18:00

Supprimer

Samedi: 08:00 - 14:00

Supprimer

Email

S'inscrire

Mise en place du lien API -> Frontend

Diagramme de Classes UML

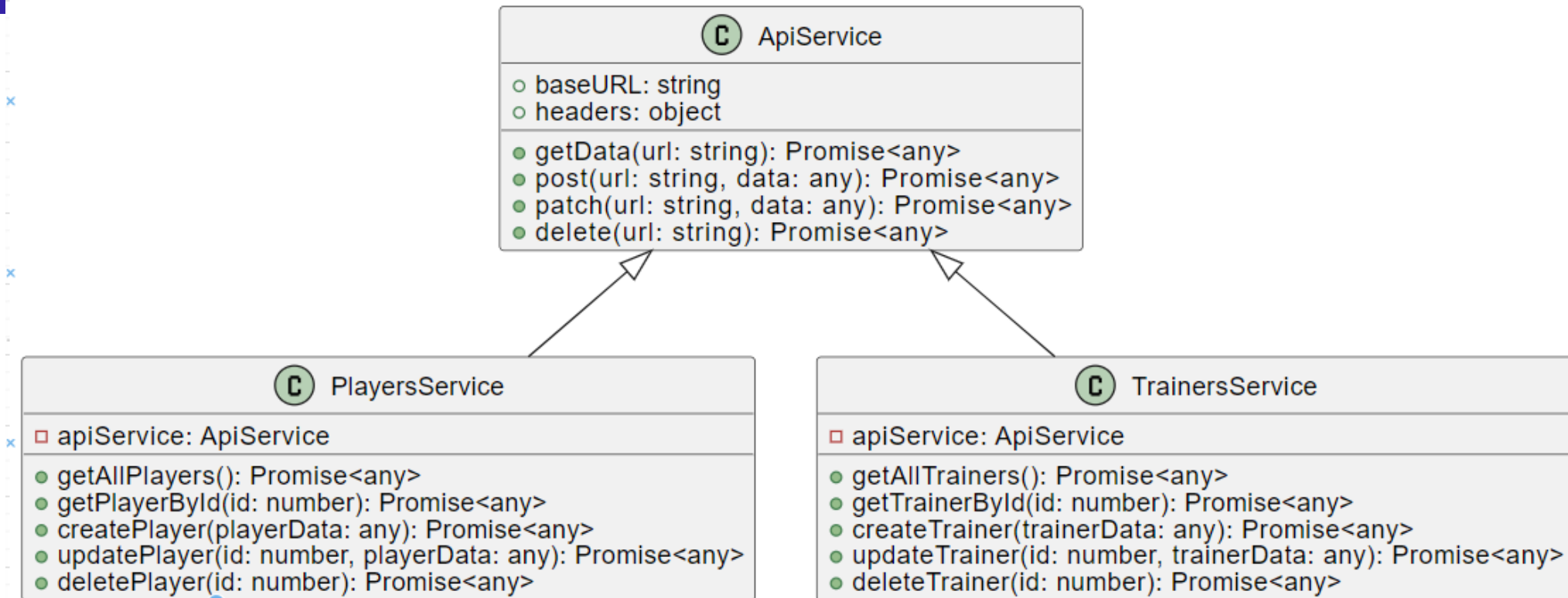


Diagramme de Classes UML

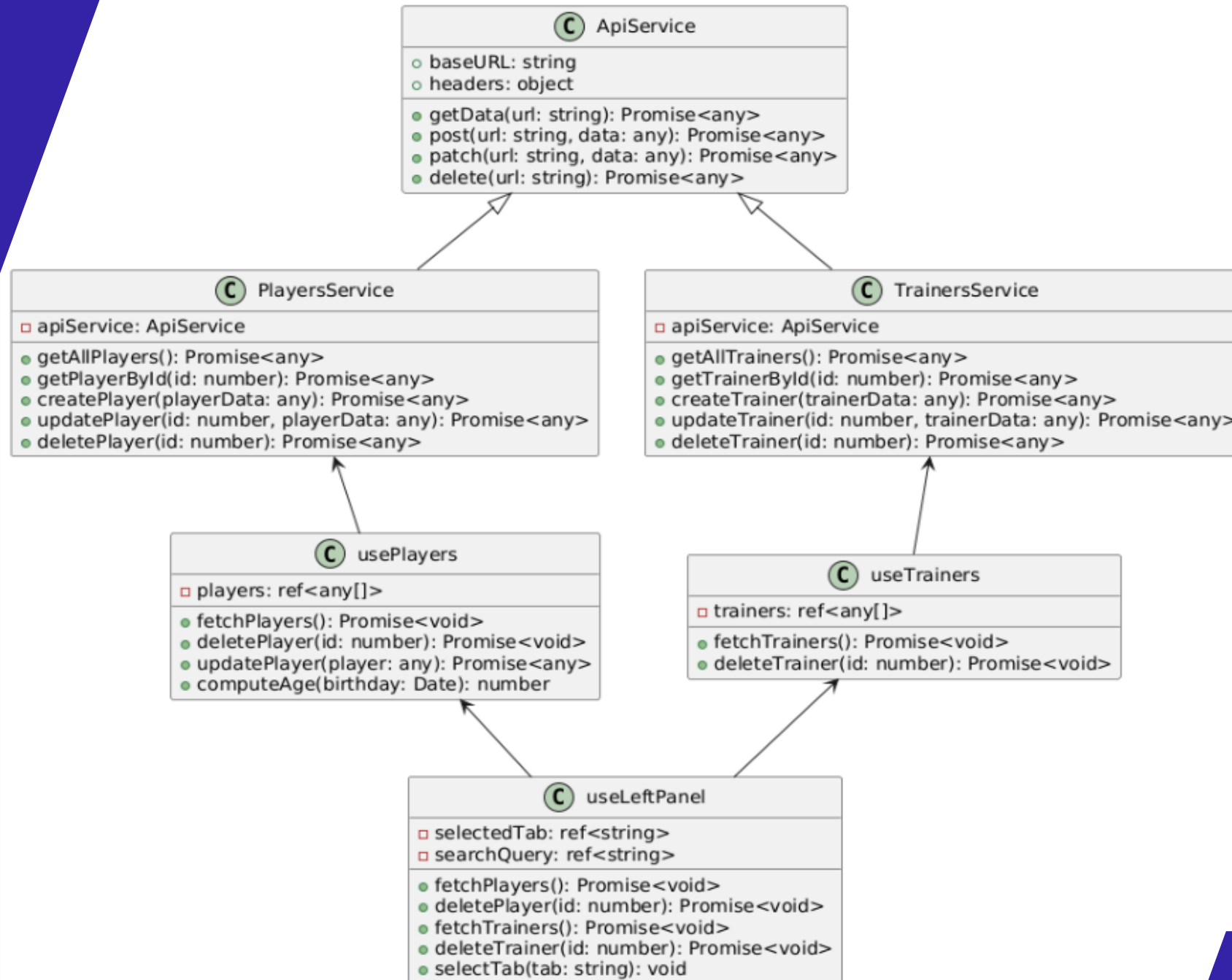
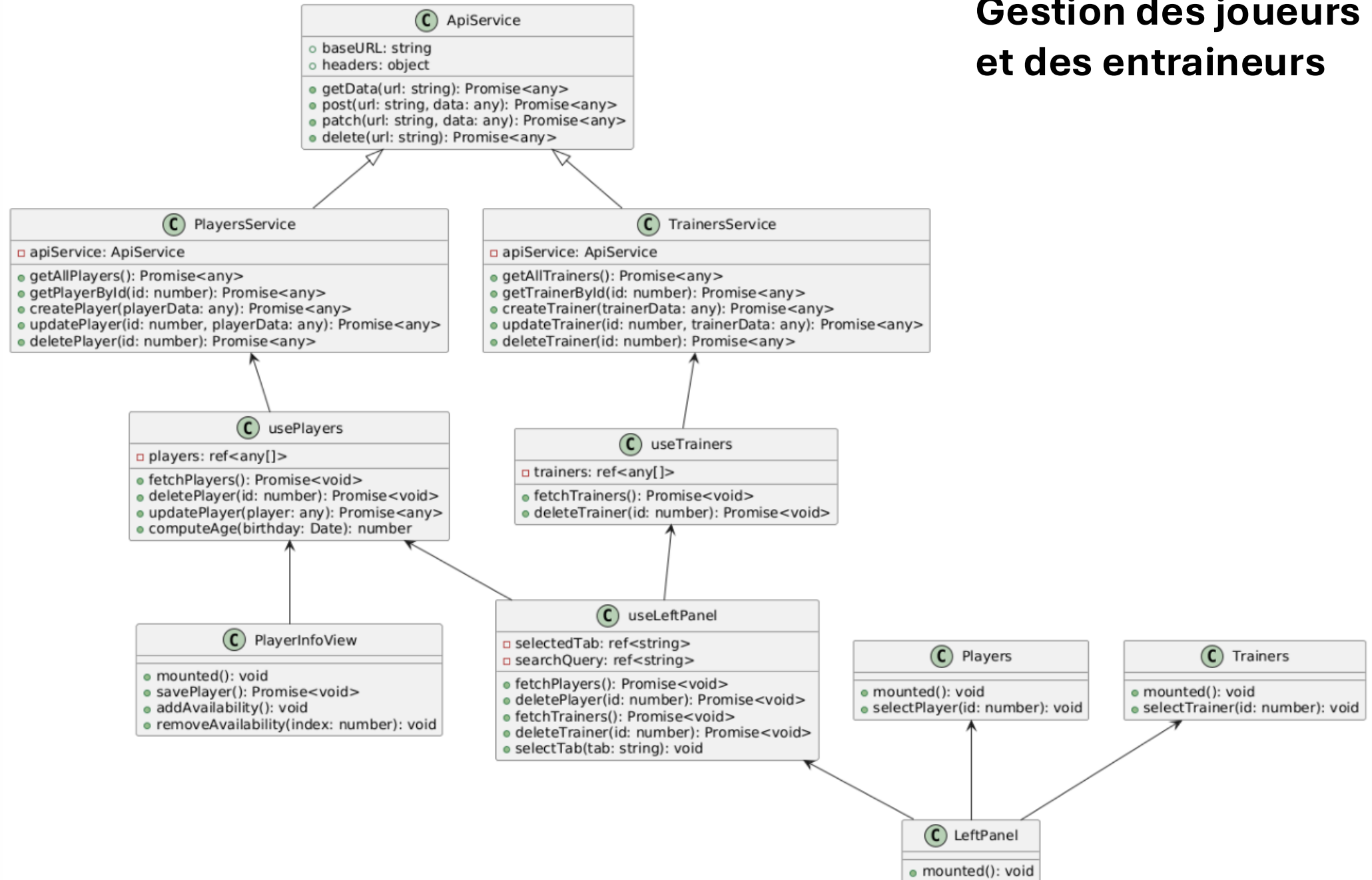
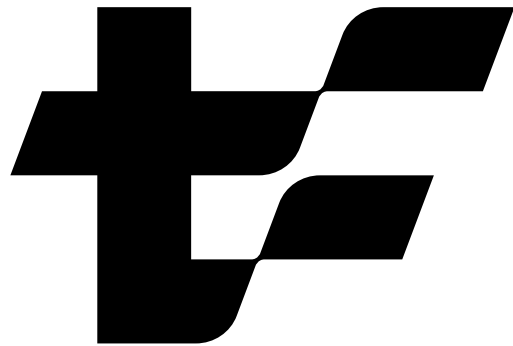


Diagramme de Classes UML

Gestion des joueurs et des entraineurs

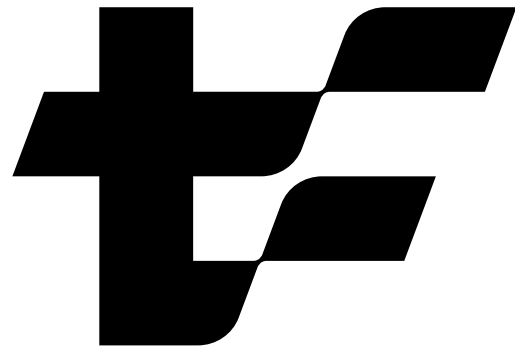


Présentation du fonctionnement



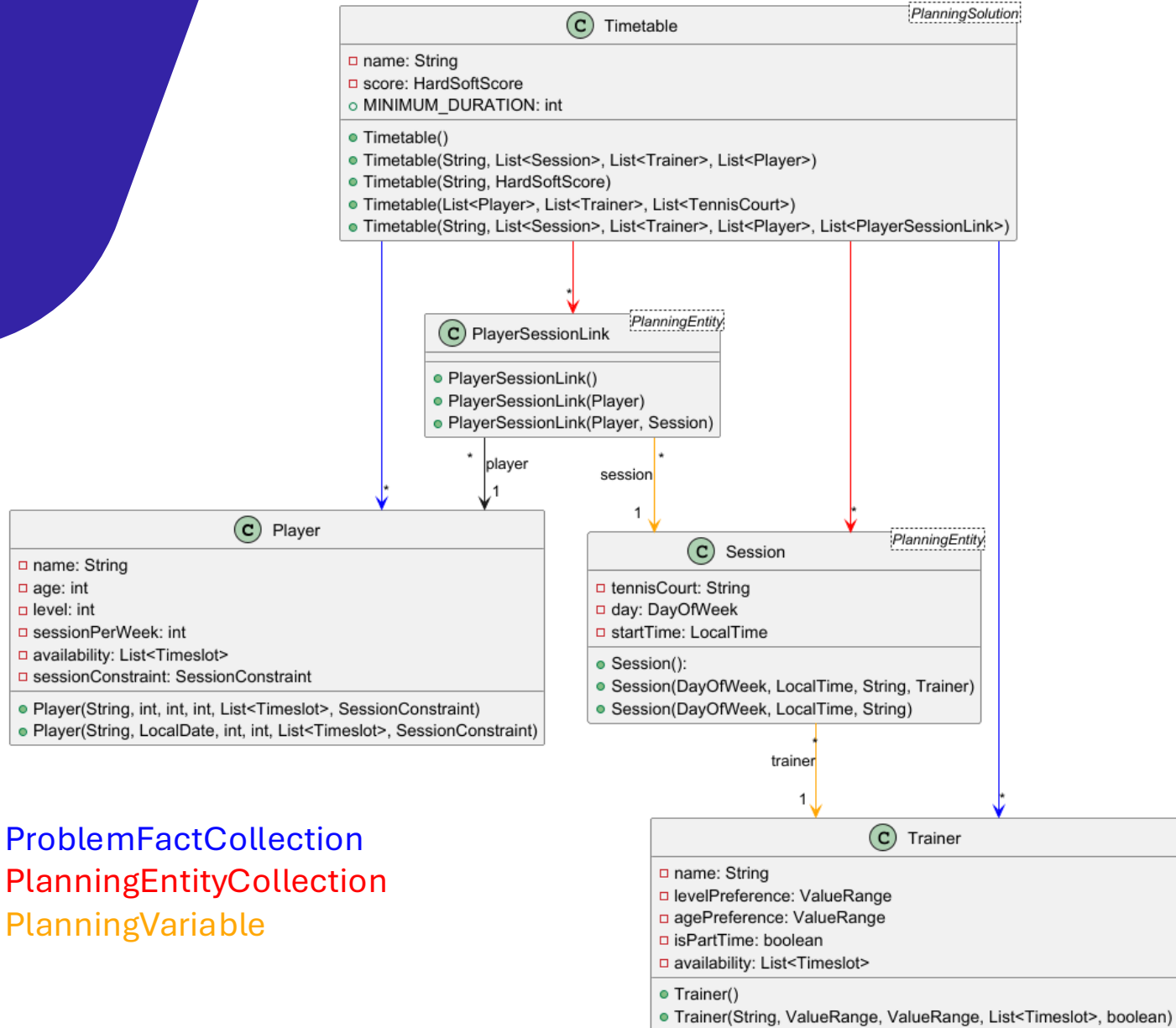
timefold

Présentation du fonctionnement



timefold

1. Domaine du problème
2. Fonctionnement des contraintes
3. Tests des contraintes
4. Exemple d'utilisation



Données d'entrée :

- joueurs ;
- entraîneurs ;
- terrains de tennis.

Données générées :

- sessions ;
- liens joueur-session.

Données de sortie :

- sessions (non vides)
- liens joueur-session.

ProblemFactCollection

PlanningEntityCollection

PlanningVariable

Une contrainte permet de :

- définir le bon et le mauvais ;
- guider l'algorithme ;
- comparer plusieurs solutions.

Le poids d'une contrainte :

- pénalise ou récompense ;
- très bien, bien, mal ou très mal.

Étape 1 : filtrer les données

- choix d'une collection ;
- regroupement ;
- opération spéciale ;
- filtrage.

`forEach, forEachUniquePair, groupBy, join,
filter, ifExist,...`

```
Constraint ageDifferenceBis(ConstraintFactory constraintFactory) {  
    return constraintFactory  
        .forEach(PlayerSessionLink.class)  
        .groupBy(PlayerSessionLink::getSession, ConstraintCollectors.toList())  
        .filter((_, playerSessionLinks) → {  
            int maxAgeDifference = playerSessionLinks.get(0).getPlayer().getSessionConstraint().ageDifference();  
            List<Integer> ages = playerSessionLinks.stream()  
                .map(PlayerSessionLink::getPlayer)  
                .map(Player::getAge)  
                .toList();  
            int ageDifference = new ValueRange(ages).size() - 1;  
            return ageDifference > maxAgeDifference;  
        })  
}
```


Étape 1 : filtrer les données

- choix d'une collection ;
- regroupement ;
- opération spéciale ;
- filtrage.

`forEach`, `forEachUniquePair`, `groupBy`, `join`,
`filter`, `ifExist`,...

Étape 2 : pénaliser ou récompenser

- choix d'un type de poids (léger ou lourd) ;

```
Constraint ageDifferenceBis(ConstraintFactory constraintFactory) {  
    return constraintFactory  
        .forEach(PlayerSessionLink.class)  
        .groupBy(PlayerSessionLink::getSession, ConstraintCollectors.toList())  
        .filter((_, playerSessionLinks) → ageOverflow(playerSessionLinks) > 0)  
        .penalize(HardSoftScore.ONE_SOFT)
```

Étape 1 : filtrer les données

- choix d'une collection ;
- regroupement ;
- opération spéciale ;
- filtrage.

`forEach, forEachUniquePair, groupBy, join, filter, ifExist,...`

Étape 2 : pénaliser ou récompenser

- choix d'un type de poids (léger ou lourd) ;
- formule de calcul du poids.

`penalize, reward, HardSoftScore.ONE_HARD,...`

```
Constraint ageDifferenceBis(ConstraintFactory constraintFactory) {  
    return constraintFactory  
        .forEach(PlayerSessionLink.class)  
        .groupBy(PlayerSessionLink::getSession, ConstraintCollectors.toList())  
        .filter((_, playerSessionLinks) → ageOverflow(playerSessionLinks) > 0)  
        .penalize(HardSoftScore.ONE_SOFT, (_, playerSessionLinks) → ageOverflow(playerSessionLinks))  
}
```

Étape 1 : filtrer les données

- choix d'une collection ;
- regroupement ;
- opération spéciale ;
- filtrage.

`forEach, forEachUniquePair, groupBy, join, filter, ifExist, ...`

Étape 2 : pénaliser ou récompenser

- choix d'un type de poids (léger ou lourd) ;
- formule de calcul du poids.

`penalize, reward, HardSoftScore.ONE_HARD, ...`

Étape 3 : justifier le résultat

- construire la contrainte ;
- nommer la contrainte.

```
Constraint ageDifferenceBis(ConstraintFactory constraintFactory) {  
    return constraintFactory  
        .forEach(PlayerSessionLink.class)  
        .groupBy(PlayerSessionLink::getSession, ConstraintCollectors.toList())  
        .filter((_, playerSessionLinks) → ageOverflow(playerSessionLinks))  
        .penalize(HardSoftScore.ONE_SOFT, (_, playerSessionLinks) → ageOverflow(playerSessionLinks))  
        .asConstraint("La différence d'âge maximum par groupe est dépassée");  
}
```

Étape 4 : créer un collecteur personnalisé

- supplier : initialise le collecteur ;
- accumulator : gère l'ajout d'un élément ;
- finisher : termine et retourne la valeur.

```
Constraint ageDifferenceBis(ConstraintFactory constraintFactory) {  
    return constraintFactory  
        .forEach(PlayerSessionLink.class)  
        .groupBy(PlayerSessionLink::getSession, ConstraintCollectors.toList())  
        .filter((_, playerSessionLinks) → ageOverflow(playerSessionLinks))  
        .penalize(HardSoftScore.ONE_SOFT, (_, playerSessionLinks) → ageOverflow(playerSessionLinks))  
        .asConstraint("La différence d'âge maximum par groupe est dépassée");  
}
```

Étape 4 : créer un collecteur personnalisé

- supplier : initialise le collecteur
- accumulator : gère l'ajout d'un élément
- finisher : termine et retourne la valeur

```
Constraint ageDifferenceBis(ConstraintFactory constraintFactory) {  
    return constraintFactory  
        .forEach(PlayerSessionLink.class)  
        .groupBy(PlayerSessionLink::getSession, CustomCollectors.ageOverflow())  
        .filter((_, ageOverflow) → ageOverflow > 0)  
        .penalize(HardSoftScore.ONE_SOFT, (_, ageOverflow) → ageOverflow)  
        .asConstraint("La différence d'âge maximum par groupe est dépassée");  
}
```

Étape 4 : créer un collecteur personnalisé

- supplier : initialise le collecteur
- accumulator : gère l'ajout d'un élément
- finisher : termine et retourne la valeur

Étape 5 : meilleure justification

- objet utilisant les variables de la contrainte ;
- génère un String personnalisé.

```
Constraint ageDifferenceBis(ConstraintFactory constraintFactory) {  
    return constraintFactory  
        .forEach(PlayerSessionLink.class)  
        .groupBy(PlayerSessionLink::getSession, CustomCollectors.ageOverflow())  
        .filter((_, ageOverflow) → ageOverflow > 0)  
        .penalize(HardSoftScore.ONE_SOFT, (_, ageOverflow) → ageOverflow)  
        .asConstraint("La différence d'âge maximum par groupe est dépassée");  
}
```

Étape 4 : créer un collecteur personnalisé

- supplier : initialise le collecteur
- accumulator : gère l'ajout d'un élément
- finisher : termine et retourne la valeur

Étape 5 : meilleure justification

- objet utilisant les variables de la contrainte ;
- génère un String personnalisé.

```
Constraint ageDifferenceBis(ConstraintFactory constraintFactory) {  
    return constraintFactory  
        .forEach(PlayerSessionLink.class)  
        .groupBy(PlayerSessionLink::getSession, CustomCollectors.ageOverflow())  
        .filter((_, ageOverflow) → ageOverflow > 0)  
        .penalize(HardSoftScore.ONE_SOFT, (_, ageOverflow) → ageOverflow)  
        .justifyWith((session, ageOverflow, _) → new AgeOverflowJustification(session, ageOverflow))  
        .asConstraint("La différence d'âge maximum par groupe est dépassée");  
}
```


Étape 4 : créer un collecteur personnalisé

- supplier : initialise le collecteur
- accumulator : gère l'ajout d'un élément
- finisher : termine et retourne la valeur

[Voir le code de l'exemple](#)

Étape 5 : meilleure justification

- objet utilisant les variables de la contrainte ;
- génère un String personnalisé.

[Voir le code de l'exemple](#)

```
Constraint ageDifferenceBis(ConstraintFactory constraintFactory) {  
    return constraintFactory  
        .forEach(PlayerSessionLink.class)  
        .groupBy(PlayerSessionLink::getSession, CustomCollectors.ageOverflow())  
        .filter((_, ageOverflow) → ageOverflow > 0)  
        .penalize(HardSoftScore.ONE_SOFT, (_, ageOverflow) → ageOverflow)  
        .justifyWith((session, ageOverflow, _) → new AgeOverflowJustification(session, ageOverflow))  
        .asConstraint("La différence d'âge maximum par groupe est dépassée");  
}
```

Contraintes implémentées :

- différence d'âge ;
- différence de niveau ;
- taille du groupe.

PlayerSessionLinks:

player01(8, 5) →
player02(8, 5) →
player03(8, 5) →
player04(8, 5) →
player05(8, 5) →
player06(8, 5) →
player07(8, 5) →
player08(13, 7) →
player09(12, 7) →
player10(11, 6) →
player11(16, 3) →
player12(17, 2) →
player13(17, 3) →

Contraintes implémentées :

- différence d'âge ; → 2 *ans*
- différence de niveau ; → 1 *niveau*
- taille du groupe. → 3 à 6 *joueurs*

PlayerSessionLinks:

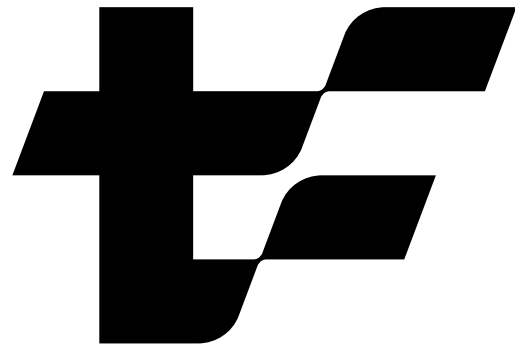
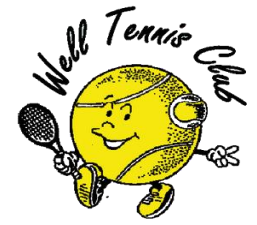
player01(8, 5) → [Terrain 1 le lundi a 08:00]
player02(8, 5) → [Terrain 1 le lundi a 08:00]
player03(8, 5) → [Terrain 1 le lundi a 08:00]
player04(8, 5) → [Terrain 1 le lundi a 08:00]
player05(8, 5) → [Terrain 1 le lundi a 08:30]
player06(8, 5) → [Terrain 1 le lundi a 08:30]
player07(8, 5) → [Terrain 1 le lundi a 08:30]
player08(13, 7) → [Terrain 1 le lundi a 09:00]
player09(12, 7) → [Terrain 1 le lundi a 09:00]
player10(11, 6) → [Terrain 1 le lundi a 09:00]
player11(16, 3) → [Terrain 1 le lundi a 09:30]
player12(17, 2) → [Terrain 1 le lundi a 09:30]
player13(17, 3) → [Terrain 1 le lundi a 09:30]

Bugs et Améliorations

- Page administrateur non encore responsive.
- Problème d'auto-incrémentation en présence de données initiales dans la table.
- Problème de jointures de tables lors de l'ajout de disponibilités via "Players".

Objectif : itération 3

- Lier l'emploi du temps, les données et les contraintes du serveur avec l'interface web.
- Finir l'implémentation de TimeFold.
- Ajouter la fonctionnalité de modification des données, avec mise à jour dans la base de données.
- Générer de nouveaux plannings après prise en compte des ajustements.
- Joindre TimeFold et le serveur Spring.
- Vérifier la bonne transmission et sauvegarde des données.
- Ajout de Spring Security et lien avec la page de connexion de l'interface web.



timefold