Projet Hackathon - UML2 - ESGI

Diagramme de classes métier

Basé sur l'analyse du cahier des charges fourni, les principales entités métier identifiées pour le projet Hackat'Orga sont les suivantes :

- **Hackathon** : Représente un événement hackathon. Il a un nom, une date, un lieu, un thème, un nombre de places limité et une date limite d'inscription.
- **Participant** : Représente une personne qui s'inscrit à un hackathon. Il a un nom, un prénom, un email, un téléphone, une date de naissance et un lien vers son portfolio. Un participant peut s'inscrire à plusieurs hackathons.
- **Inscription** : Représente l'acte d'un participant s'inscrivant à un hackathon. Elle contient la date d'inscription et un numéro unique d'inscription. Elle lie un Participant à un Hackathon.
- **Équipe** : Représente une équipe formée pour un hackathon. Elle a un nom et est associée à un projet. Une équipe est composée de plusieurs participants, dont un chef de projet.
- **Projet** : Représente un projet sur lequel les équipes travaillent pendant le hackathon. Il a un nom et un lien vers le prototype.
- **MembreJury** : Représente un membre du jury. Il peut attribuer des notes aux équipes. Un membre du jury peut aussi être un participant pour d'autres hackathons.
- **NoteJury** : Représente la note attribuée par un membre du jury à une équipe pour un hackathon donné.
- **PhaseHackathon** : Représente une phase spécifique du déroulement d'un hackathon (accueil, travail, présentation, etc.). Elle a un nom, une heure de début et une heure de fin.
- **Coach** : Représente un coach affecté à une équipe.

Relations entre les classes :

- Un **Hackathon** peut avoir plusieurs **Inscriptions**.
- Un **Participant** peut avoir plusieurs **Inscriptions**.
- Une **Inscription** lie un **Participant** à un **Hackathon**.
- Un **Hackathon** peut avoir plusieurs **Équipes**.
- Une **Équipe** est composée de plusieurs **Participants** (dont un chef de projet).

- Un **Participant** peut faire partie de plusieurs **Équipes** (mais pas sur le même hackathon).
- Une **Équipe** travaille sur un seul **Projet**.
- Un **Projet** peut être travaillé par plusieurs **Équipes**.
- Un **Hackathon** a plusieurs **PhaseHackathon**.
- Un **MembreJury** attribue plusieurs **NoteJury**.
- Une **NoteJury** est attribuée par un **MembreJury** à une **Équipe**.
- Un **Coach** est affecté à une **Équipe**.

Voici une représentation textuelle simplifiée du diagramme de classes. Pour une représentation graphique, un outil de modélisation UML serait nécessaire.

```
classDiagram
    class Hackathon {
        +String nom
        +Date date
        +String lieu
        +String theme
        +int placesLimitees
        +Date dateLimiteInscription
    }

    class Participant {
        +String nom
        +String prenom
        +String email
        +String telephone
        +Date dateNaissance
        +String portfolioUrl
    }

    class Inscription {
        +Date dateInscription
        +int numeroInscription
    }

    class Equipe {
        +String nomEquipe
        +int classement
        +int noteFinale
    }

    class Projet {
        +String nomProjet
        +String lienPrototype
    }

    class MembreJury {
        +String nom
```

```
        +String prenom
        +String email
    }

    class NoteJury {
        +int note
    }

    class PhaseHackathon {
        +String nomPhase
        +Time heureDebut
        +Time heureFin
    }

    class Coach {
        +String nom
        +String prenom
    }

    Hackathon "1" -- "*" Inscription : contient
    Participant "1" -- "*" Inscription : s'inscrit
    Hackathon "1" -- "*" Equipe : forme
    Equipe "1" -- "*" Participant : comprend
    Equipe "1" -- "1" Projet : travaille_sur
    Projet "1" -- "*" Equipe : est_travaillé_par
    Hackathon "1" -- "*" PhaseHackathon : a_pour_phases
    MembreJury "1" -- "*" NoteJury : attribue
    NoteJury "*" -- "1" Equipe : est_attribuée_à
    Coach "1" -- "*" Equipe : est_affecté_à
    MembreJury <|-- Participant : peut_être
```

Base de données correspondante

La base de données sera relationnelle, avec des tables correspondant aux classes métier identifiées. Voici une ébauche des tables et de leurs colonnes principales, ainsi que les clés primaires (PK) et étrangères (FK).

Table `Hackathons`

Colonne	Type	Description	Clé
`id_hackathon`	INT	Clé primaire auto-incrémentée	PK
`nom`	VARCHAR(255)	Nom du hackathon	
`date`	DATE	Date du hackathon	

Colonne	Type	Description	Clé
`lieu`	VARCHAR(255)	Lieu du hackathon	
`theme`	VARCHAR(255)	Thème du hackathon	
`places_limitees`	INT	Nombre maximum de participants	
`date_limite_inscription`	DATE	Date limite pour s'inscrire	

Table `Participants`

Colonne	Type	Description	Clé
`id_participant`	INT	Clé primaire auto-incrémentée	PK
`nom`	VARCHAR(255)	Nom du participant	
`prenom`	VARCHAR(255)	Prénom du participant	
`email`	VARCHAR(255)	Adresse email (unique)	
`telephone`	VARCHAR(20)	Numéro de téléphone	
`date_naissance`	DATE	Date de naissance	
`portfolio_url`	VARCHAR(255)	Lien vers le portfolio	

Table `Inscriptions`

Colonne	Type	Description	Clé
`id_inscription`	INT	Clé primaire auto-incrémentée	PK
`id_hackathon`	INT	Référence au hackathon	FK (`Hackathons.id_hackathon`)
`id_participant`	INT	Référence au participant	FK (`Participants.id_participant`)
`date_inscription`	DATE		

Colonne	Type	Description	Clé
		Date de l'inscription	
`numero_inscription`	INT	Numéro unique d'inscription pour le hackathon	
`competence_outil`	TEXT	Texte libre sur compétence/outil	

Table `Equipes`

Colonne	Type	Description	Clé
`id_equipe`	INT	Clé primaire auto-incrémentée	PK
`id_hackathon`	INT	Référence au hackathon	FK (`Hackathons.id_hackathon`)
`nom_equipe`	VARCHAR(255)	Nom de l'équipe	
`id_projet`	INT	Référence au projet sur lequel l'équipe travaille	FK (`Projets.id_projet`)
`id_chef_projet`	INT	Référence au participant chef de projet	FK (`Participants.id_participant`)

Colonne	Type	Description	Clé
`classement`	INT	Classement final de l'équipe (1, 2, 3)	
`note_finale`	INT	Note finale de l'équipe	

Table `Projets`

Colonne	Type	Description	Clé
`id_projet`	INT	Clé primaire auto-incrémentée	PK
`nom_projet`	VARCHAR(255)	Nom du projet	
`lien_prototype`	VARCHAR(255)	Lien vers le prototype	

Table `MembresJury`

Colonne	Type	Description	Clé
`id_membre_jury`	INT	Clé primaire auto-incrémentée	PK
`nom`	VARCHAR(255)	Nom du membre du jury	
`prenom`	VARCHAR(255)	Prénom du membre du jury	
`email`	VARCHAR(255)	Adresse email (unique)	

Table `NotesJury`

Colonne	Type	Description	Clé
`id_note_jury`	INT	Clé primaire auto-incrémentée	PK
`id_membre_jury`	INT	Référence au membre du jury	FK (`MembresJury.id_membre_jury`)

Colonne	Type	Description	Clé
`id_equipe`	INT	Référence à l'équipe notée	FK (`Equipes.id_equipe`)
`note`	INT	Note attribuée (0-5)	

Table `PhasesHackathon`

Colonne	Type	Description	Clé
`id_phase`	INT	Clé primaire auto-incrémentée	PK
`id_hackathon`	INT	Référence au hackathon	FK (`Hackathons.id_hackathon`)
`nom_phase`	VARCHAR(255)	Nom de la phase (Accueil, Travail, etc.)	
`heure_debut`	TIME	Heure de début de la phase	
`heure_fin`	TIME	Heure de fin de la phase	

Table `Coachs`

Colonne	Type	Description	Clé
`id_coach`	INT	Clé primaire auto-incrémentée	PK
`nom`	VARCHAR(255)	Nom du coach	
`prenom`	VARCHAR(255)	Prénom du coach	

Table `Equipe_Participant` (Table de liaison pour la relation plusieurs-à-plusieurs entre Equipe et Participant)

Colonne	Type	Description	Clé
`id_equipe`	INT	Référence à l'équipe	PK, FK (`Equipes.id_equipe`)
`id_participant`	INT	Référence au participant	PK, FK (`Participants.id_participant`)

Table `Equipe_Coach` (Table de liaison pour la relation plusieurs-à-plusieurs entre Equipe et Coach)

Colonne	Type	Description	Clé
`id_equipe`	INT	Référence à l'équipe	PK, FK (`Equipes.id_equipe`)
`id_coach`	INT	Référence au coach	PK, FK (`Coachs.id_coach`)

Jeu de données de test

Voici un exemple de jeu de données de test pour les tables principales. Ces données sont fictives et servent à illustrer le fonctionnement de la base de données.

`Hackathons`

id_hackathon	nom	date	lieu	theme	places_limitees	date_limite
1	Hackathon InnovTech 2025	2025-06-20	Paris	Technologies Vertes	100	2025-06-10
2	Hackathon Santé Connectée	2025-07-15	Lyon	Santé et IA	80	2025-07-05

Participants

id_participant	nom	prenom	email	telephone	date_naiss
1	Dupont	Jean	jean.dupont@example.com	0612345678	1990-01-15
2	Martin	Sophie	sophie.martin@example.com	0798765432	1992-03-20
3	Bernard	Pierre	pierre.bernard@example.com	0655443322	1988-11-01
4	Dubois	Marie	marie.dubois@example.com	0711223344	1995-07-25

Inscriptions

id_inscription	id_hackathon	id_participant	date_inscription	numero_inscription	co
1	1	1	2025-06-01	1	Dé We
2	1	2	2025-06-02	2	UI Fig
3	1	3	2025-06-03	3	Ma Py
4	2	4	2025-07-01	1	Da
5	2	1	2025-07-02	2	Ba De No

Projets

id_projet	nom_projet	lien_prototype
1	Smart Recycler	http://prototype-recycler.com
2	Eco-Dashboard	http://prototype-ecodash.com

id_projet	nom_projet	lien_prototype
3	Health Monitor App	http://prototype-health.com

`Equipes`

id_equipe	id_hackathon	nom_equipe	id_projet	id_chef_projet	classement	note_f
1	1	Green Coders	1	1	1	20
2	1	Eco Innovators	2	3	2	18
3	2	Health Techies	3	4	NULL	NULL

`Equipe_Participant`

id_equipe	id_participant
1	1
1	2
2	3
3	4
3	1

`MembresJury`

id_membre_jury	nom	prenom	email
1	Durand	Claire	claire.durand@example.com
2	Lefevre	Marc	marc.lefevre@example.com

`NotesJury`

id_note_jury	id_membre_jury	id_equipe	note
1	1	1	5
2	1	2	4
3	2	1	4
4	2	2	4

`PhasesHackathon`

id_phase	id_hackathon	nom_phase	heure_debut	heure_fin
1	1	Accueil des participants	17:00:00	19:00:00
2	1	Travail en autonomie	19:00:00	23:59:59
3	1	Travail en autonomie (Jour 2)	00:00:00	12:00:00
4	1	Repas festif	12:00:00	14:00:00
5	1	Présentation des projets	14:00:00	16:00:00
6	1	Délibération et résultats	16:00:00	17:00:00

`Coachs`

id_coach	nom	prenom
1	Dubois	Thomas
2	Petit	Laura

`Equipe_Coach`

id_equipe	id_coach
1	1
2	2

id_equipe	id_coach
3	1

Diagramme des Cas d’ Utilisation

Le diagramme des cas d'utilisation permet de visualiser les interactions entre les acteurs du système et les fonctionnalités offertes par l'application Hackat'Orga. Il met en évidence les services que le système rend à ses utilisateurs.

Acteurs identifiés :

- **Participant** : Utilisateur souhaitant s'inscrire à un hackathon, créer ou gérer son profil, et potentiellement devenir chef de projet au sein d'une équipe.
- **Organisateur (Hackat'Innov)** : Responsable de la gestion globale des hackathons, incluant l'initialisation, la publication, la gestion des inscriptions, la constitution des équipes, la gestion des résultats et l'édition du planning.
- **Membre du Jury** : Personne chargée d'évaluer les prototypes des équipes et d'attribuer des notes.
- **Coach** : Personne accompagnant les équipes durant le hackathon.

Cas d'utilisation principaux (pour la partie Hackat'Orga) :

- **Gérer Profil Participant** : Permet au participant de créer, consulter et modifier ses informations personnelles (nom, prénom, email, téléphone, date de naissance, portfolio).
- **Consulter Hackathons Disponibles** : Permet à un participant ou à un visiteur de consulter la liste des hackathons, leurs caractéristiques (thème, dates, lieu, places disponibles, date limite d'inscription) et le planning.
- **S'inscrire à un Hackathon** : Permet à un participant connecté de s'inscrire à un hackathon spécifique, en fournissant une compétence ou un outil.
- **Gérer Inscriptions** : Permet à l'organisateur de visualiser et de gérer les inscriptions aux hackathons (clôture des inscriptions, etc.).
- **Constituer Équipes** : Permet à l'organisateur de former les équipes à partir des participants inscrits, d'affecter des projets et de désigner un chef de projet par équipe.
- **Affecter Coachs aux Équipes** : Permet à l'organisateur d'affecter des coachs aux équipes.
- **Gérer Projets** : Permet à l'organisateur de gérer les projets des hackathons (création, modification, lien prototype).

- **Gérer Votes Jury** : Permet au membre du jury d'attribuer des notes aux équipes et à l'organisateur de gérer la délibération et les résultats.
- **Éditer Planning Hackathon** : Permet à l'organisateur de définir et de modifier les phases et les horaires d'un hackathon.
- **Consulter Résultats** : Permet aux participants et organisateurs de consulter les classements et les notes finales des équipes.

```
graph TD
    A[Participant] --> UC1(Gérer Profil Participant)
    A --> UC2(Consulter Hackathons Disponibles)
    A --> UC3(S'inscrire à un Hackathon)

    B[Organisateur] --> UC4(Gérer Inscriptions)
    B --> UC5(Constituer Équipes)
    B --> UC6(Affecter Coachs aux Équipes)
    B --> UC7(Gérer Projets)
    B --> UC8(Gérer Votes Jury)
    B --> UC9(Éditer Planning Hackathon)
    B --> UC10(Consulter Résultats)

    C[Membre du Jury] --> UC8

    D[Coach] --> UC6

    UC3 -- extends --> UC1
    UC8 -- include --> UC10

    subgraph Système Hackat'Orga
        UC1
        UC2
        UC3
        UC4
        UC5
        UC6
        UC7
        UC8
        UC9
        UC10
    end
```

Diagrammes de séquence système et diagramme d’activités des cas d’utilisation

Pour cette section, nous allons détailler deux cas d'utilisation clés afin de démontrer la compréhension des principes de conception de SI en UML :

1. **S'inscrire à un Hackathon** (Cas d'utilisation centré sur le Participant)
2. **Constituer Équipes** (Cas d'utilisation centré sur l'Organisateur)

Cas d'utilisation : S'inscrire à un Hackathon

Description : Ce cas d'utilisation permet à un participant de s'inscrire à un hackathon spécifique après s'être connecté et, si nécessaire, avoir créé son profil. Il doit sélectionner le hackathon et fournir une compétence ou un outil pertinent pour le thème.

Flux d'événements principal : 1. Le Participant consulte la liste des hackathons disponibles. 2. Le Système affiche les hackathons avec leurs détails (nom, date, lieu, thème, places restantes, date limite d'inscription). 3. Le Participant sélectionne un hackathon et indique son intention de s'inscrire. 4. Le Système vérifie si le Participant est connecté. Si non, il lui demande de se connecter ou de créer un profil. 5. Le Participant se connecte ou crée un profil (voir cas d'utilisation "Gérer Profil Participant"). 6. Le Système affiche le formulaire d'inscription pour le hackathon sélectionné. 7. Le Participant saisit une compétence ou un outil pertinent pour le thème du hackathon. 8. Le Participant soumet le formulaire d'inscription. 9. Le Système valide l'inscription (vérifie les places disponibles, la date limite). 10. Le Système enregistre l'inscription, mémorise la date de saisie et attribue un numéro unique d'inscription. 11. Le Système confirme l'inscription au Participant.

Flux alternatifs : * **A1 : Places limitées atteintes ou date limite dépassée :** * 9a. Si les places sont toutes prises ou la date limite est dépassée, le Système informe le Participant que l'inscription n'est pas possible. * 9b. Le cas d'utilisation se termine.

Diagramme de Séquence Système (SSD) : S'inscrire à un Hackathon

```
sequenceDiagram
    actor Participant
    participant Système

    Participant->>Système: consulterHackathonsDisponibles()
    Système-->>Participant: afficherHackathons(listeHackathons)
    Participant->>Système: selectionnerHackathon(idHackathon)
    alt Participant non connecté
```

```
        Système-->>Participant:
demanderConnexionOuCreationProfil()
        Participant->>Système: seConnecterOuCreerProfil()
    end
    Système-->>Participant:
afficherFormulaireInscription(idHackathon)
    Participant->>Système: soumettreInscription(idHackathon,
competenceOutil)
    alt Inscription valide
        Système->>Système: validerInscription(idHackathon,
idParticipant)
        Système->>Système: enregistrerInscription(idHackathon,
idParticipant, competenceOutil)
        Système-->>Participant:
confirmerInscription(numeroInscription)
    else Inscription invalide
        Système-->>Participant: informerEchecInscription(raison)
    end
```

Diagramme d'Activités : S'inscrire à un Hackathon

```
@startuml
start
:Consulter Hackathons Disponibles;
:Sélectionner Hackathon;
if (Participant connecté?) then (non)
  :Demander Connexion ou Création Profil;
  :Se Connecter ou Créer Profil;
else (oui)
endif
:Afficher Formulaire Inscription;
:Saisir Compétence/Outil;
:Soumettre Inscription;
if (Inscription valide?) then (oui)
  :Enregistrer Inscription;
  :Confirmer Inscription;
else (non)
  :Informer Échec Inscription;
endif
stop
@enduml
```

Cas d'utilisation : Constituer Équipes

Description : Ce cas d'utilisation permet à l'organisateur de former les équipes pour un hackathon donné une fois les inscriptions closes. Il implique l'affectation des participants aux équipes, la désignation d'un chef de projet et l'association d'un projet à chaque équipe.

Flux d'événements principal : 1. L'Organisateur accède à la fonctionnalité de constitution des équipes pour un hackathon. 2. Le Système affiche la liste des participants inscrits et non encore affectés pour le hackathon. 3. L'Organisateur crée une nouvelle équipe, lui donne un nom et sélectionne un projet parmi les projets retenus. 4. Le Système enregistre la nouvelle équipe. 5. L'Organisateur sélectionne des participants inscrits pour les ajouter à l'équipe. 6. Le Système ajoute les participants à l'équipe et vérifie qu'un participant n'est affecté qu'à une seule équipe pour ce hackathon. 7. L'Organisateur désigne un des participants de l'équipe comme chef de projet. 8. Le Système enregistre le chef de projet pour l'équipe. 9. L'Organisateur peut répéter les étapes 3 à 8 pour créer d'autres équipes. 10. L'Organisateur valide la constitution des équipes. 11. Le Système finalise la constitution des équipes pour le hackathon.

Flux alternatifs : * **A1 : Participant déjà affecté :** * 6a. Si l'Organisateur tente d'ajouter un participant déjà affecté à une autre équipe pour le même hackathon, le Système refuse l'ajout et informe l'Organisateur. * 6b. L'Organisateur peut choisir un autre participant.

Diagramme de Séquence Système (SSD) : Constituer Équipes

```
sequenceDiagram
    actor Organisateur
    participant Système

    Organisateur->>Système:
accederConstitutionEquipes(idHackathon)
    Système-->>Organisateur:
afficherParticipantsNonAffectes(listeParticipants)

    loop Création d'équipes
        Organisateur->>Système: creerEquipe(nomEquipe, idProjet)
        Système->>Système: enregistrerEquipe(nomEquipe,
idProjet)
        Système-->>Organisateur:
confirmerCreationEquipe(idEquipe)

        loop Ajout de participants à l'équipe
            Organisateur->>Système:
ajouterParticipantAEquipe(idEquipe, idParticipant)
            alt Participant non affecté
                Système->>Système:
lierParticipantEquipe(idEquipe, idParticipant)
                Système-->>Organisateur:
confirmerAjoutParticipant()
            else Participant déjà affecté
                Système-->>Organisateur:
informerParticipantDejaAffecte()
            end
```

```
        end

        Organisateur->>Système: designerChefProjet(idEquipe,
idParticipantChefProjet)
        Système->>Système: enregistrerChefProjet(idEquipe,
idParticipantChefProjet)
        Système-->>Organisateur: confirmerChefProjet()
    end

    Organisateur->>Système:
validerConstitutionEquipes(idHackathon)
    Système->>Système: finaliserConstitutionEquipes(idHackathon)
    Système-->>Organisateur: confirmerFinalisationEquipes()
```

Diagramme d'Activités : Constituer Équipes

```
@startuml
start
:Accéder Constitution Équipes;
:Afficher Participants Non Affectés;
repeat
  :Créer Nouvelle Équipe (Nom, Projet);
  :Enregistrer Équipe;
  repeat
    :Sélectionner Participant à Ajouter;
    if (Participant déjà affecté?) then (oui)
      :Informer Participant Déjà Affecté;
    else (non)
      :Ajouter Participant à Équipe;
    endif
  while (Ajouter d'autres participants?) is (oui)
  :Désigner Chef de Projet;
  :Enregistrer Chef de Projet;
repeat while (Créer d'autres équipes?) is (oui)
:Valider Constitution Équipes;
:Finaliser Constitution Équipes;
:Confirmer Finalisation;
stop
@enduml
```

Maquettage des cas d’ utilisation

Pour illustrer les cas d'utilisation "Consulter Hackathons Disponibles" et "S'inscrire à un Hackathon", voici des maquettes visuelles de l'interface utilisateur.

Maquette 1 : Liste des Hackathons Disponibles

Cette maquette représente la page d'accueil ou une section dédiée où les utilisateurs peuvent voir les hackathons ouverts aux inscriptions. Chaque hackathon est présenté sous forme de carte avec les informations essentielles et un bouton d'action.

Maquette 2 : Formulaire d'Inscription à un Hackathon

Cette maquette illustre le formulaire que les participants rempliraient pour s'inscrire à un hackathon. Il inclut les champs nécessaires pour les informations personnelles et les détails spécifiques à l'inscription au hackathon.

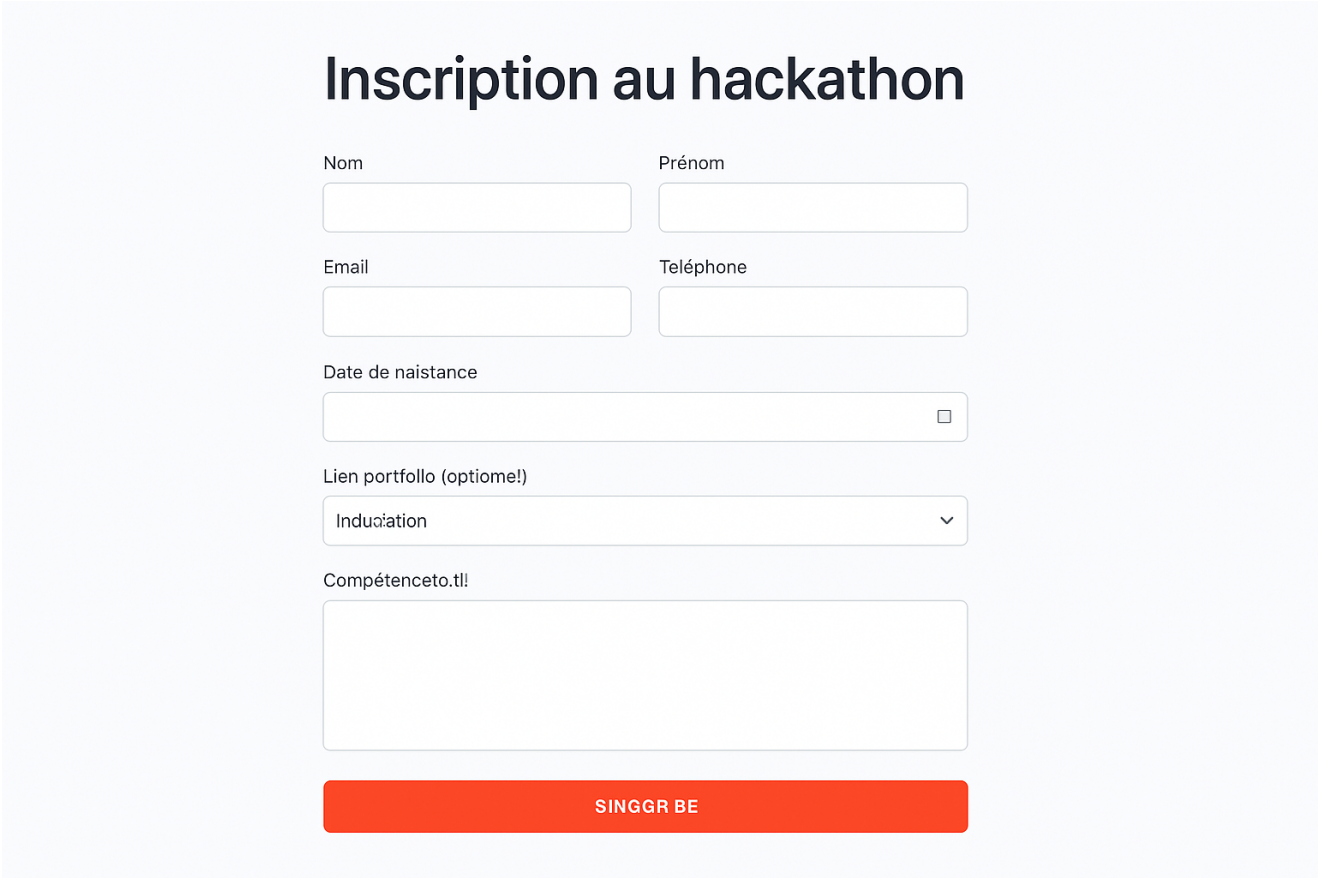

Diagramme de séquence MVC des cas d’ utilisation

Nous allons détailler le diagramme de séquence MVC pour le cas d'utilisation "S'inscrire à un Hackathon", qui est un processus clé impliquant l'interaction entre l'interface utilisateur, la logique métier et la persistance des données.

Cas d'utilisation : S'inscrire à un Hackathon

Description : Ce diagramme illustre comment les composants Model-View-Controller (MVC) interagissent pour permettre à un participant de s'inscrire à un hackathon. Il montre le flux des requêtes et des réponses entre la vue (interface utilisateur), le contrôleur (logique de traitement) et le modèle (gestion des données).

Diagramme de Séquence MVC : S'inscrire à un Hackathon

```
sequenceDiagram
    actor Participant
    participant Vue as "Vue: Formulaire Inscription"
    participant Controleur as "Contrôleur:
InscriptionController"
    participant Modele as "Modèle: InscriptionService /
ParticipantService / HackathonService"
    database BaseDeDonnees as "Base de Données"
```

```
    Participant->>Vue: clique sur "S'inscrire"
    Vue->>Controleur: soumettreInscription(donneesFormulaire)

    Controleur->>Modele: validerDonnees(donneesFormulaire)
    Modele->>Modele: vérifier validité (champs, format)
    Modele-->>Controleur: données valides/invalides

    alt Données invalides
        Controleur-->>Vue: afficherErreur(messagesErreur)
        Vue-->>Participant: affiche les erreurs
    else Données valides
        Controleur->>Modele: getHackathon(idHackathon)
        Modele->>BaseDeDonnees: SELECT * FROM Hackathons WHERE
id = idHackathon
        BaseDeDonnees-->>Modele: données hackathon
        Modele-->>Controleur: objet Hackathon

        Controleur->>Modele: verifierDisponibilite(idHackathon)
        Modele->>BaseDeDonnees: SELECT COUNT(*) FROM
Inscriptions WHERE id_hackathon = idHackathon
        BaseDeDonnees-->>Modele: nombreInscriptions
        Modele-->>Controleur: estDisponible (true/false)

        alt Hackathon non disponible
            Controleur-->>Vue: afficherMessage(Hackathon
complet ou date limite dépassée)
            Vue-->>Participant: affiche message
        else Hackathon disponible
            Controleur->>Modele:
creerInscription(donneesInscription)
            Modele->>BaseDeDonnees: INSERT INTO Inscriptions
(...)
            BaseDeDonnees-->>Modele: inscription enregistrée
            Modele-->>Controleur: confirmation inscription

            Controleur-->>Vue:
afficherConfirmation(numeroInscription)
            Vue-->>Participant: affiche confirmation
        end
    end
```