Année universitaire 2022-2023

SAÉ Semestre 3 :

Partie
Analyse et Conception

Wittmann Grégory

Rouillon Tom

UNIVERSITÉ DE LORRAINE

IUT Saint-Dié

Sommaire :

Présentation de l'application

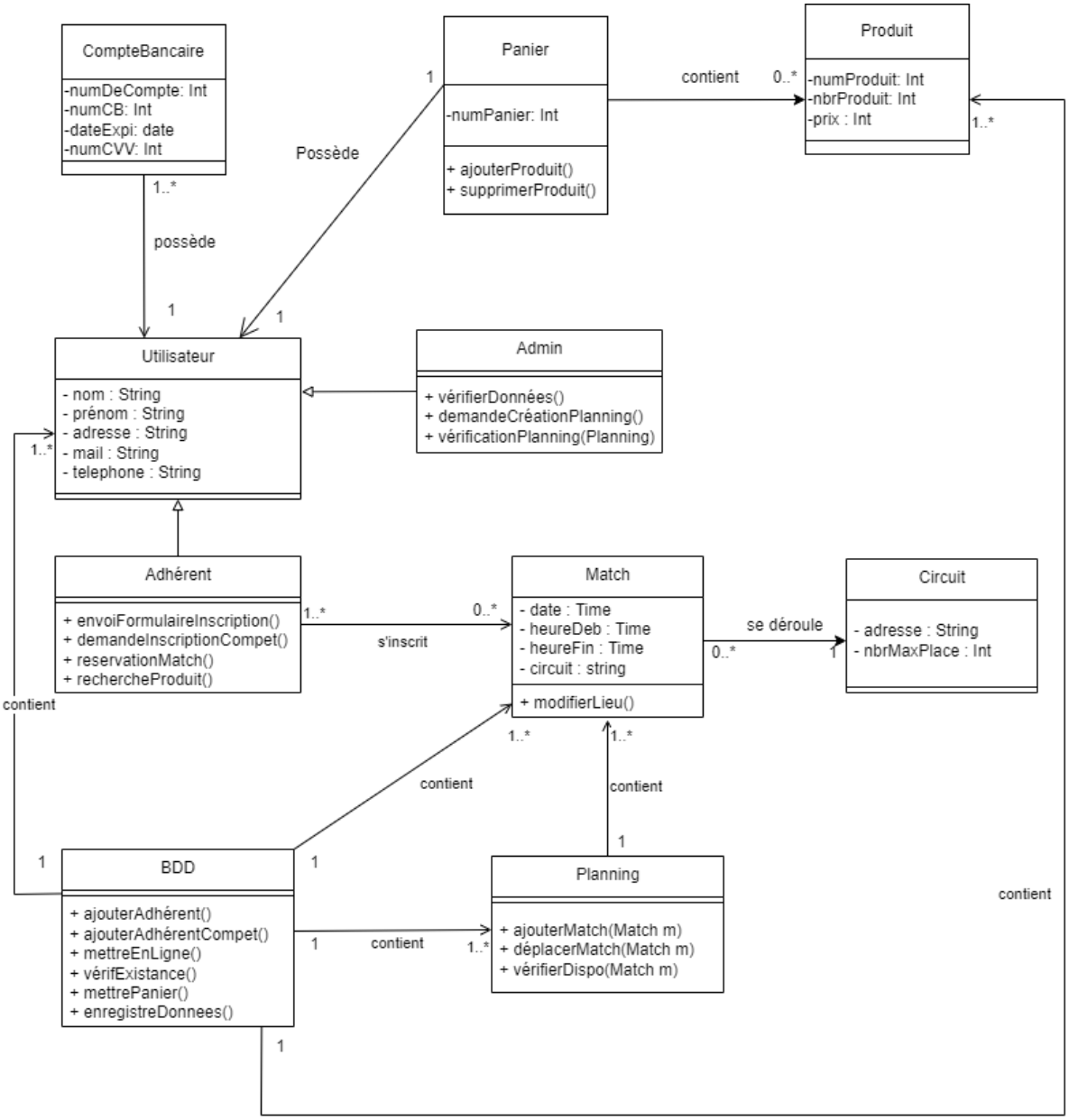
Notre projet est de développer une application web pour un club de karting, comportant 3 modules : un pour la gestion des adhérents, un autre pour la gestion des compétitions et des résultats, et un dernier pour la planification des matchs.

Cela permettra donc à l'utilisateur de s'inscrire en tant qu'adhérent, ce qui lui offrira l'accès à d'autres fonctionnalités, comme l'inscription à des compétitions ou la réservation d'un circuit, dans notre cas une piste.

D'un autre côté, les administrateurs devront être capable de saisir les compétitions et de valider les inscriptions des adhérents à ces compétitions. Ils devront aussi pouvoir créer les plannings.

Diagramme de packages

Cette partie et ce diagramme aide à comprendre quelles sont les dépendances entre chaque classe, c'est-à-dire quelle classe à besoin de quelle classe.



Nous retrouvons ainsi toutes les classes qui vont être développées dans les prochaines parties. Ainsi, nous avons une vue d'ensemble, ce qui nous permet de voir que la classe BDD est au cœur de tous les packages, ce qui est logique puisque l'appli entière reposera sur des données communes.

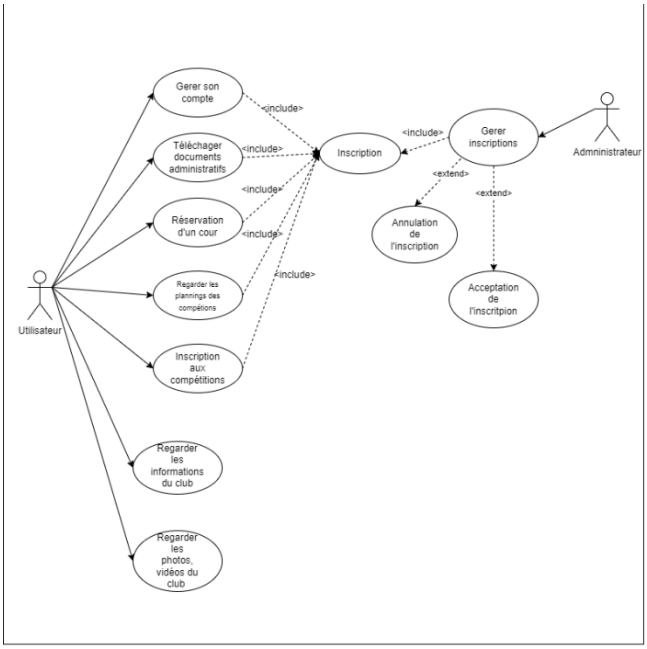
La classe Utilisateur sera elle aussi importante, puisque l'on voit qu'elle est reliée, directement ou indirectement, à presque toutes les autres classes. C'est aussi quelque chose de logique, puisque les utilisateurs sont censés pouvoir avoir à toutes les fonctionnalités présentes sur l'appli.

Diagrammes pour chaque package

Dans cette partie, nous analyserons chaque package de l'application grâce aux différents diagrammes UML. Cela permettra d'avoir une vision plus précise de ce que sera l'application finale.

Inscription :

Cas d'utilisation :

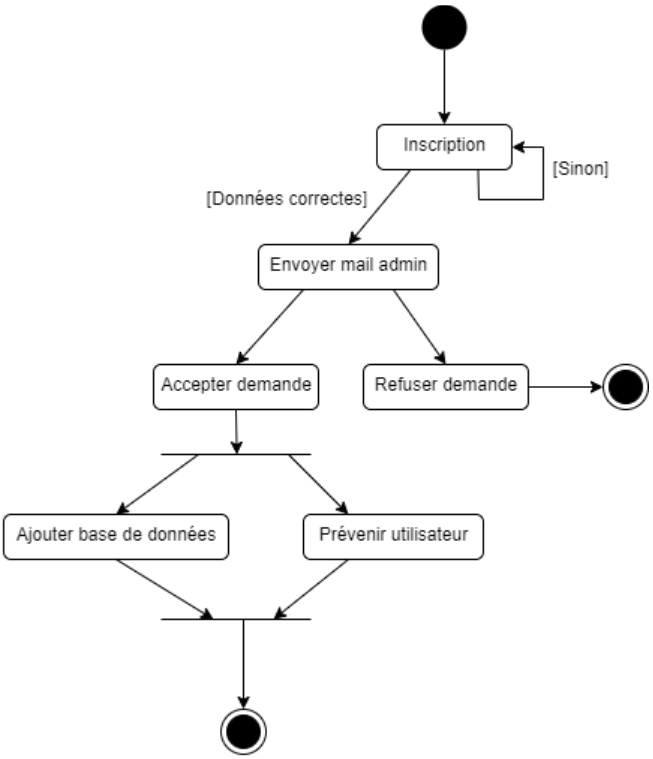


Ce diagramme nous aide à comprendre comment fonctionnera l'inscription. Par exemple, il faudra être obligatoirement connecté pour s'inscrire aux compétitions ou pour réserver un circuit.

Nous voyons aussi que ce sont les administrateurs qui géreront la validation ou non des inscriptions. Une autre version sera proposée pour enlever cette étape.

Inscription pour devenir adhérent

Diagramme d'activité



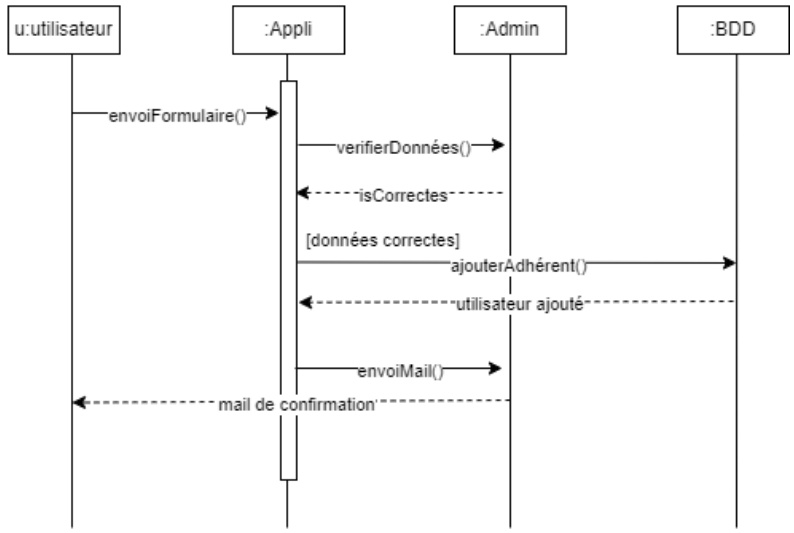
Scénarios

1. L'utilisateur se rend sur la page d'inscription
2. L'utilisateur tape son nom, son prénom, son mot de passe et son mail
3. Le site indique à l'utilisateur que les données saisies sont bonnes
4. Un mail est envoyé à un administrateur pour confirmer l'inscription
5. L'administrateur accepte la demande
6. Un mail est envoyé à l'utilisateur pour confirmer son inscription
7. L'utilisateur est ajouté dans la base de données en tant qu'adhérent

---

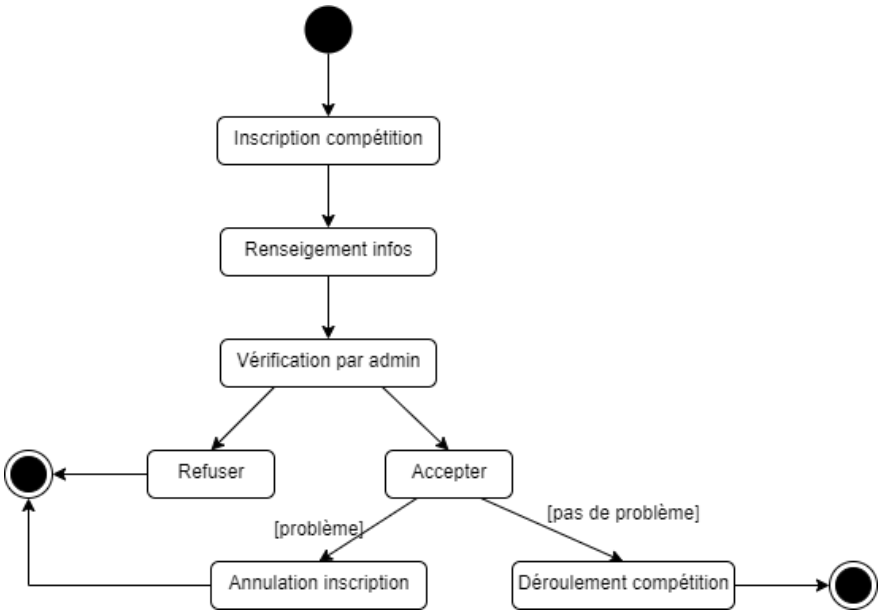
1. L'utilisateur se rend sur la page d'inscription
2. L'utilisateur tape son nom, son prénom, son mot de passe et son mail
3. Le site indique à l'utilisateur que les données saisies sont bonnes
4. Un mail est envoyé à un administrateur pour confirmer l'inscription
5. L'administrateur refuse la demande
6. Un mail est envoyé à l'utilisateur expliquant la raison du refus

Diagramme de séquence



Inscription aux compétitions

Diagramme d'activité



Ici, ce diagramme nous montre qu'il est possible pour l'admin d'annuler l'inscription si un problème se présente, comme des conditions météos dangereuses ou une indisponibilité de la part de l'adhérent.

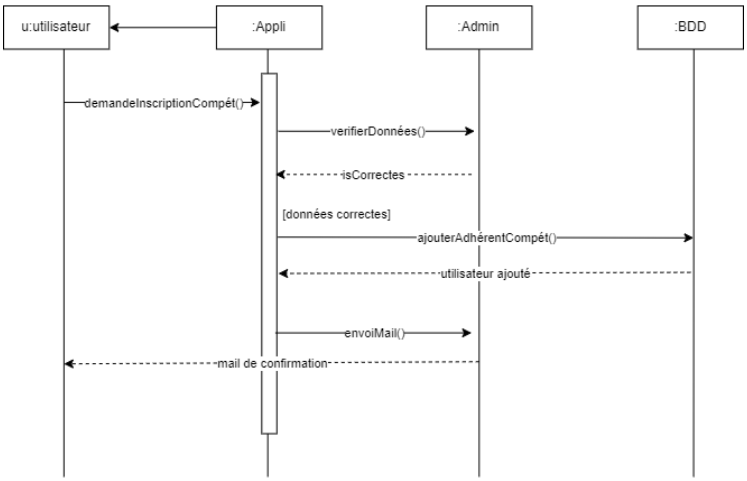
Scénarios

1. L’adhérent s’inscrit à une compétition en renseignant ses infos
2. Sa demande est envoyée à un administrateur
3. L’administrateur accepte la demande
4. Un mail est envoyé à l’adhérent récapitulant les détails
5. L’adhérent est ajouté à la base de données en tant que participant
6. La compétition se déroule

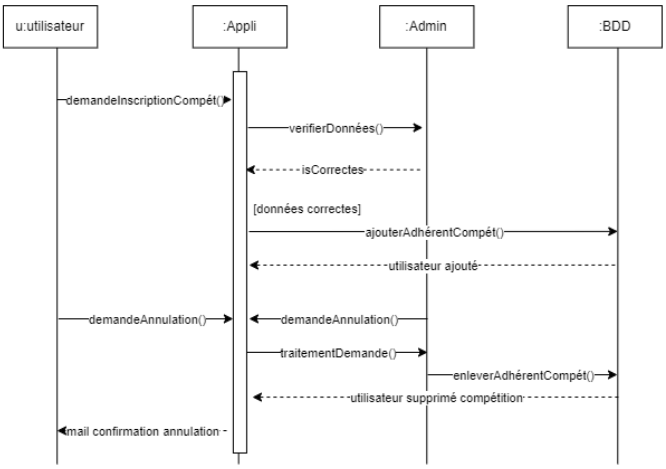
---

1. L’adhérent s’inscrit à une compétition
2. Sa demande est envoyée à un admin
3. L’admin accepte la demande
4. Un mail est envoyé à l’adhérent récapitulant les détails
5. Un problème survient
6. L’admin prend contact avec l’adhérent pour annuler la compétition
7. La compétition est annulée et supprimée de la base de données

Diagramme de séquence



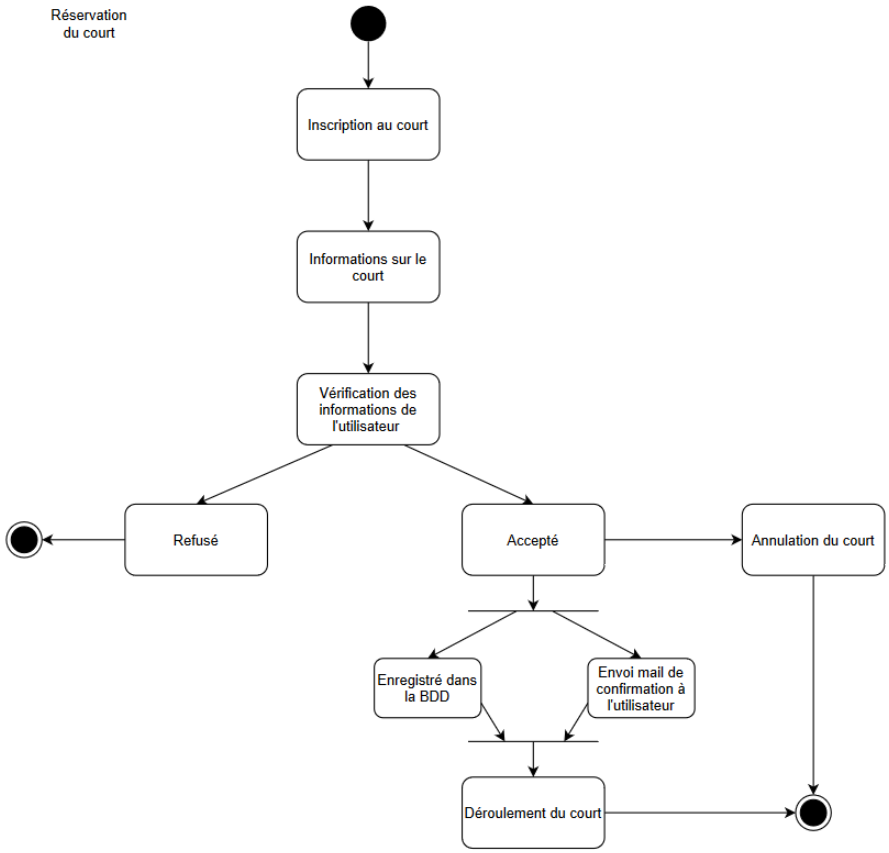
Ce diagramme représente le 1er scénario où tout se passe bien, avec la compétition qui se déroule normalement.



Ce 2ème diagramme représente le 2ème scénario, où un problème survient et où la compétition est annulée ou où l'adhérent est supprimé de cet compétition.

Réservation d'un circuit

Diagramme d'activité



Ici le diagramme d'activité nous montre comment se déroule en interne la réservation d'un circuit par un utilisateur. Il peut se voir refuser sa réservation comme accepter mais il peut tout de même l'annuler une fois réservée.

Scénarios

1. L'adhérent réserve un circuit
2. La demande est envoyée à un administrateur
3. L'administrateur accepte la réservation
4. L'adhérent reçoit un mail de confirmation avec les infos
5. L'adhérent est ajouté à la base de données
6. L'adhérent se présente sur le circuit et l'occupe le temps défini

---

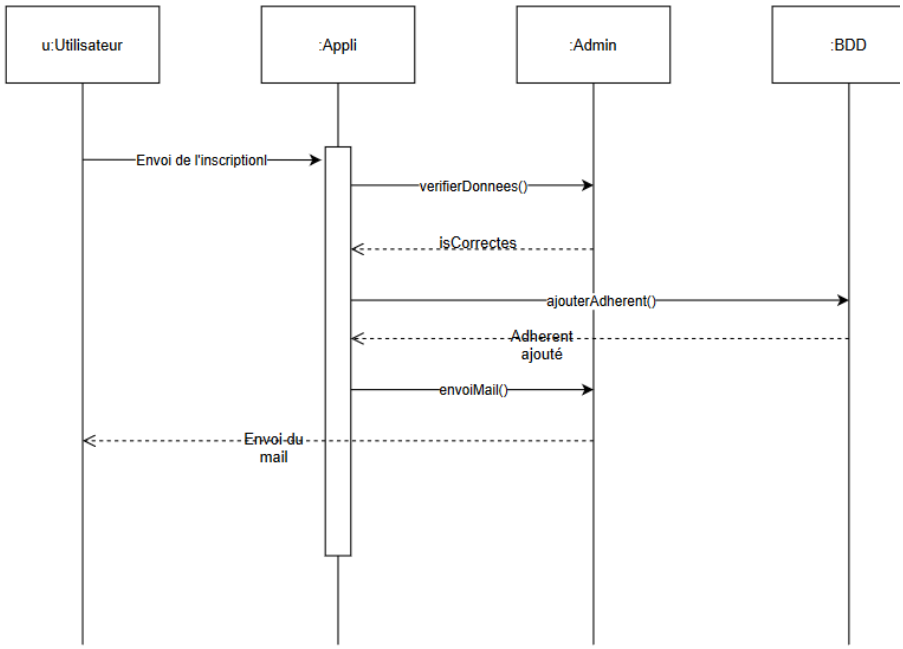
1. L'adhérent réserve un circuit
2. La demande est envoyée à un administrateur
3. L'administrateur accepte la réservation
4. L'adhérent reçoit un mail de confirmation avec les infos
5. L'adhérent est ajouté à la base de données
6. L'adhérent envoi l'annulation de sa réservation
7. L'adhérent est supprimé de la base de données
8. Mail d'annulation envoyé à l'adhérent

---

1. L'adhérent réserve un circuit
2. La demande est envoyée à un administrateur
3. L'administrateur refuse la réservation
4. Un mail d'information est envoyé à l'adhérent

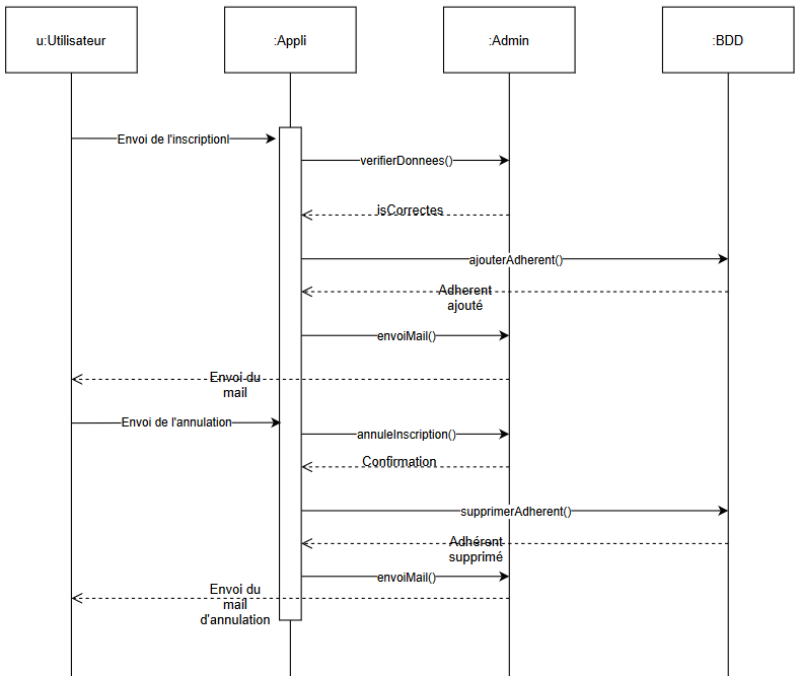
Diagramme de séquence

Scénario 1



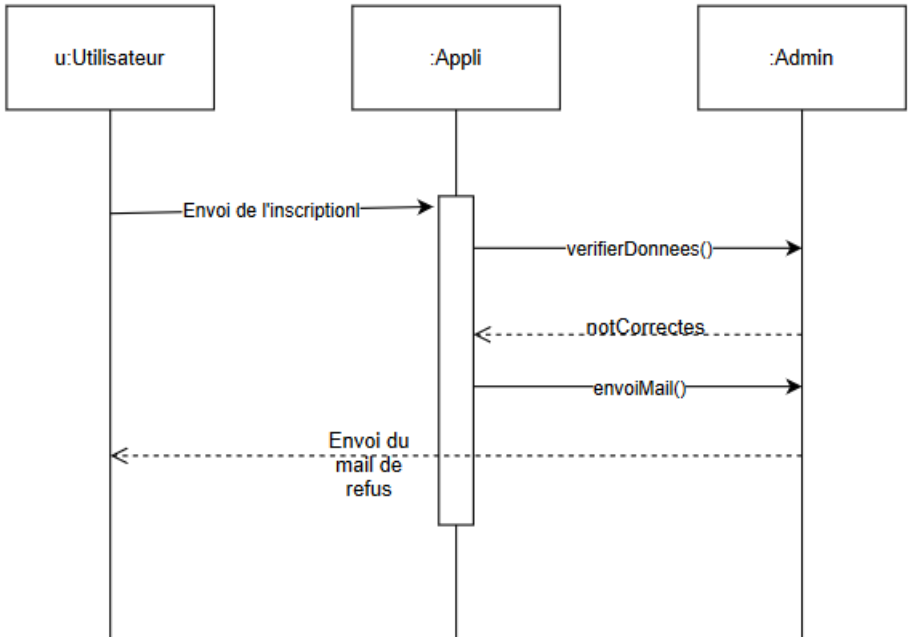
Dans ce premier scénario nous avons le cas où tout se passe correctement avec la réservation et la validation de celle-ci.

Scénario 2



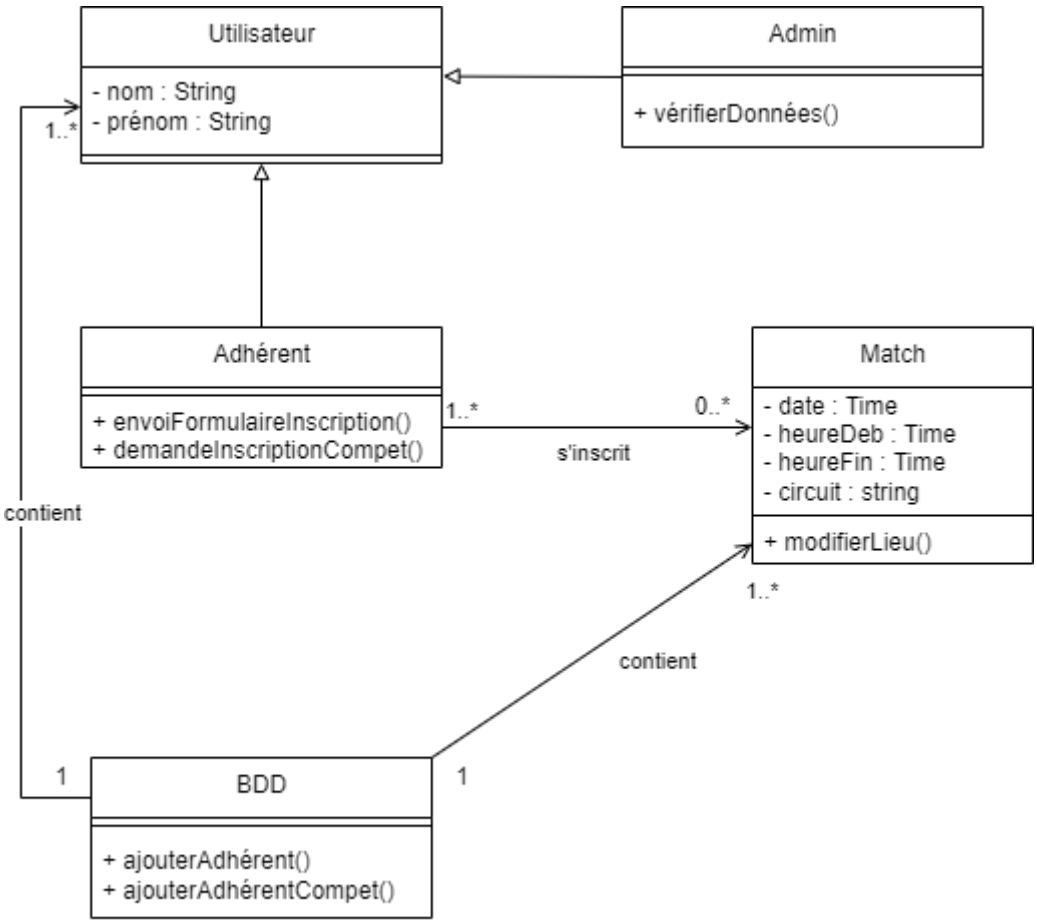
Dans le deuxième scénario nous avons le cas où l'adhérent, après avoir réservé le circuit, annule la réservation. L'application envoi alors une demande de suppression de l'adhérent dans la base de données et un mail de confirmation de l'annulation est envoyé à l'adhérent.

Scénario 3



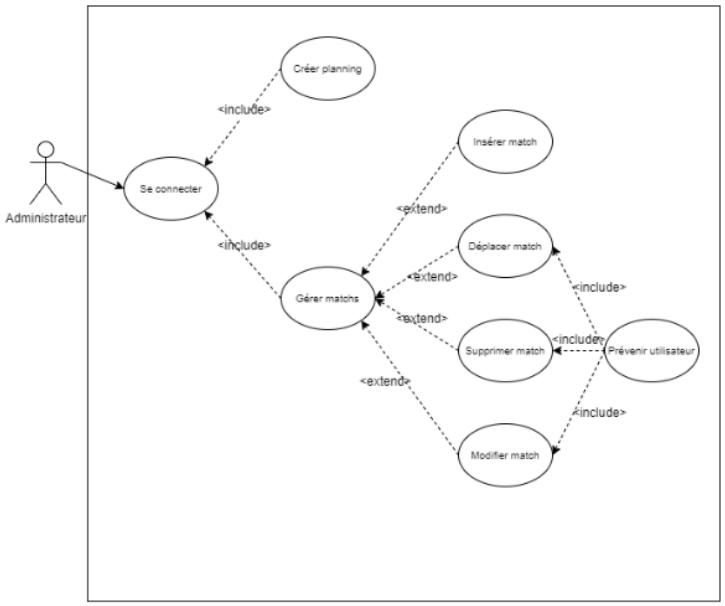
Dans le dernier scénario, nous avons le cas où l'administrateur refuse la réservation d'un utilisateur suite à une erreur dans les données inscrites lors de la réservation.

Diagramme de classe :



Planning :

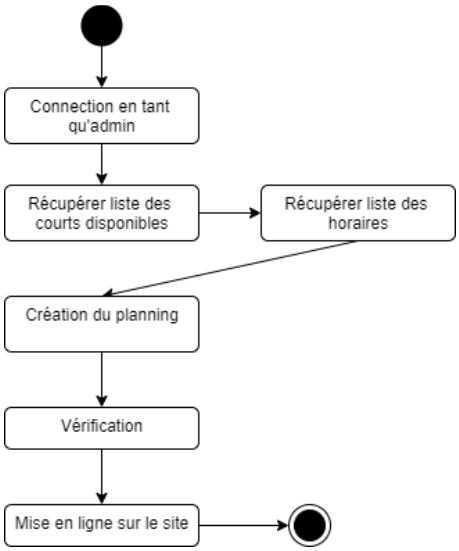
Cas d'utilisation :



Ce diagramme nous aide à comprendre ce qu'il sera possible de faire avec le planning. Déjà, seuls les administrateurs seront capables de gérer le planning. Ils pourront le créer à partir d'une liste de circuit et d'une liste d'horaires. Ils pourront aussi le gérer, c'est-à-dire soit insérer un match manuellement, en déplacer un existant sur une plage horaire de libre, en supprimer un ou en modifier un (changer le lieu, le nombre de participants…). On remarque aussi que dans ces 3 derniers cas, un mail devra être envoyé pour prévenir tous les participants du match.

Créer planning

Diagramme d'activité



Ici, on voit que la création du planning requiert uniquement d'être un admin et d'avoir la liste des circuits disponibles et la liste des horaires. Le programme Java s'occupera de tout mettre en ordre.

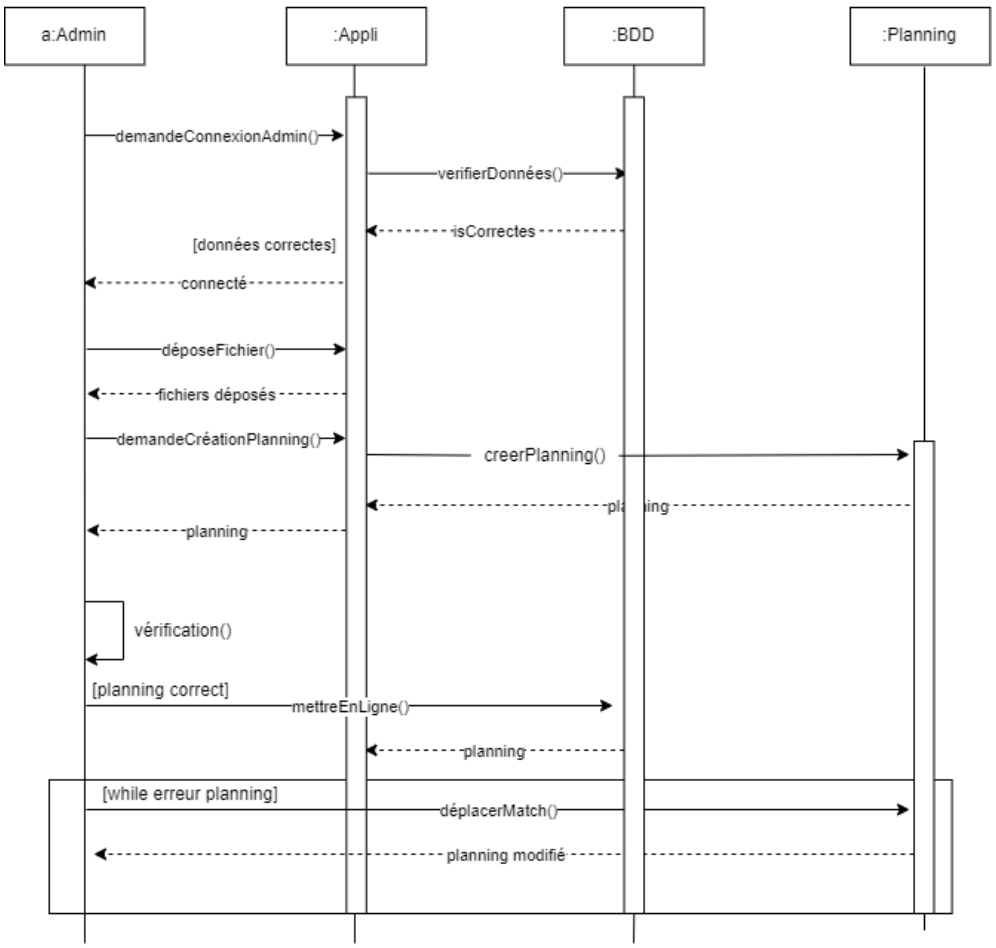
Scénarios

1. L'admin se connecte à son compte
2. Il se rend sur la page consacrée au planning
3. Il dépose les fichiers contenant les circuits et les horaires
4. Le site lui renvoie le planning créé
5. L'admin ne remarque aucune erreur
6. L'admin met en ligne le planning

---

1. L'admin se connecte à son compte
2. Il se rend sur la page consacrée au planning
3. Il dépose les fichiers contenant les circuits et les horaires
4. Le site lui renvoie le planning créé
5. L'admin remarque une erreur
6. Il change manuellement les matchs
7. L'admin met en ligne le planning

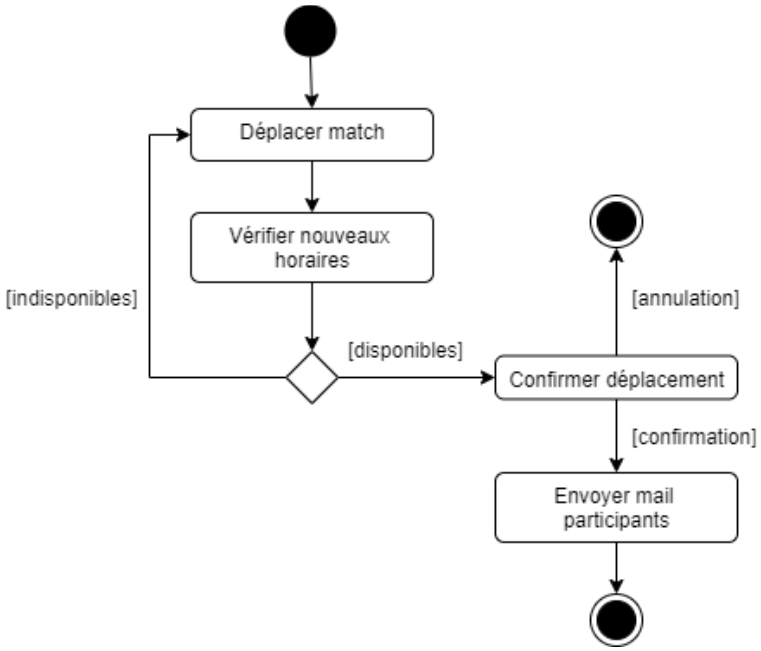
Diagramme de séquence :



Ce diagramme regroupe les deux scénarios ci-dessus, avec en premier lieu le cas où aucune erreur n'est détectée par l'admin, donc le planning est directement mis en ligne, et le deuxième cas où une ou des erreurs sont détectées, et où l'admin déplace les matchs manuellement jusqu'à ce qu'il ne voie plus d'erreurs.

Déplacer match

Diagramme d'activité



Ce diagramme nous permet de comprendre qu'un match ne peut être déplacé que si aucun autre match ou réservation ne se déroule en même temps aux nouveaux horaires sur ce circuit.

Scénarios

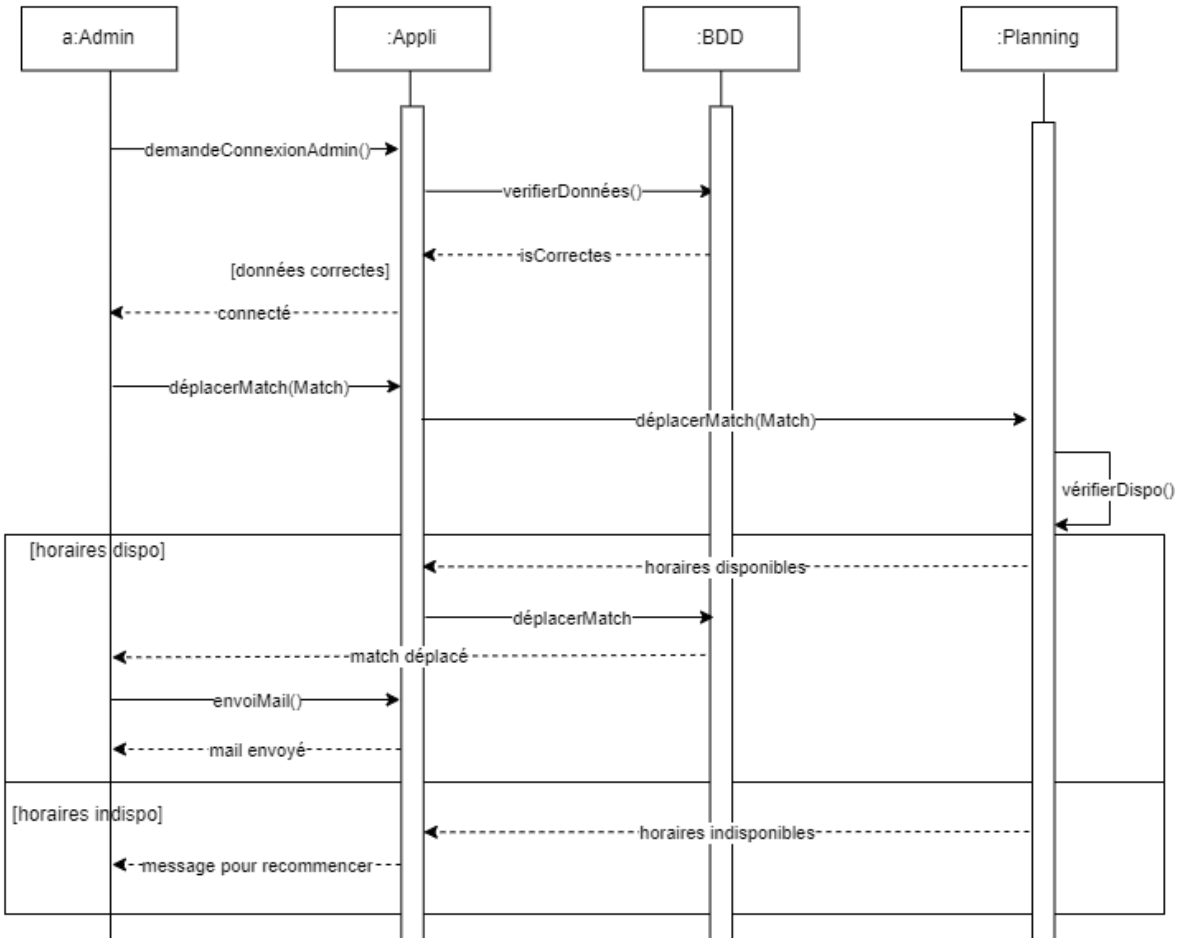
1. L'admin est connecté
2. L'admin se rend sur le planning
3. Il déplace un match
4. L'appli vérifie qu'aucun match ou réservation ne se déroule en même temps sur ce circuit
5. L'appli ne trouve aucune erreur
6. Le match est déplacé
7. Un mail est envoyé à chaque adhérent s'étant inscrit

---

1. L'admin est connecté
2. L'admin se rend sur le planning
3. Il déplace un match

4. L’appli vérifié qu’aucun match ou réservation ne se déroule en même temps sur ce circuit
5. L’appli trouve une erreur
6. Un message s’affiche à l’admin et lui demande de recommencer

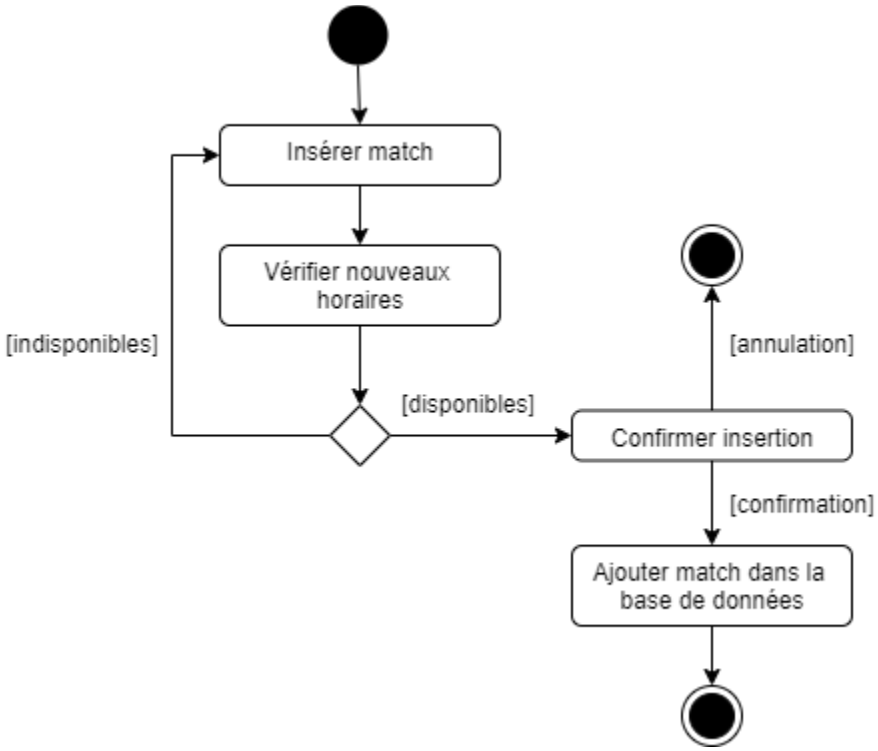
Diagramme de séquence



Ici aussi, le diagramme nous expose les deux cas présentés ci-dessus, avec les nouveaux horaires disponibles et avec les horaires indisponibles. Quand l’admin se retrouve dans ce 2ème cas, l’appli va juste lui demander de recommencer, alors que dans le 1er cas, le match va être déplacé et la base de données va être actualisé.

Insérer match

Diagramme d'activité



Scénarios

1. L'admin est connecté
2. L'admin se rend sur le planning
3. Il insère un match
4. L'appli vérifie qu'aucun match ou réservation ne se déroule en même temps sur ce circuit
5. L'appli ne trouve aucune erreur
6. Le match est inséré
7. Le nouveau match est ajouté à la base de données

---

1. L'admin est connecté
2. L'admin se rend sur le planning
3. Il insère un match

4. L’appli vérifie qu’aucun match ou réservation ne se déroule en même temps sur ce circuit
5. L’appli détecte une erreur
6. Un message s’afficher et l’appli demande à l’admin de recommencer

Diagramme de séquence

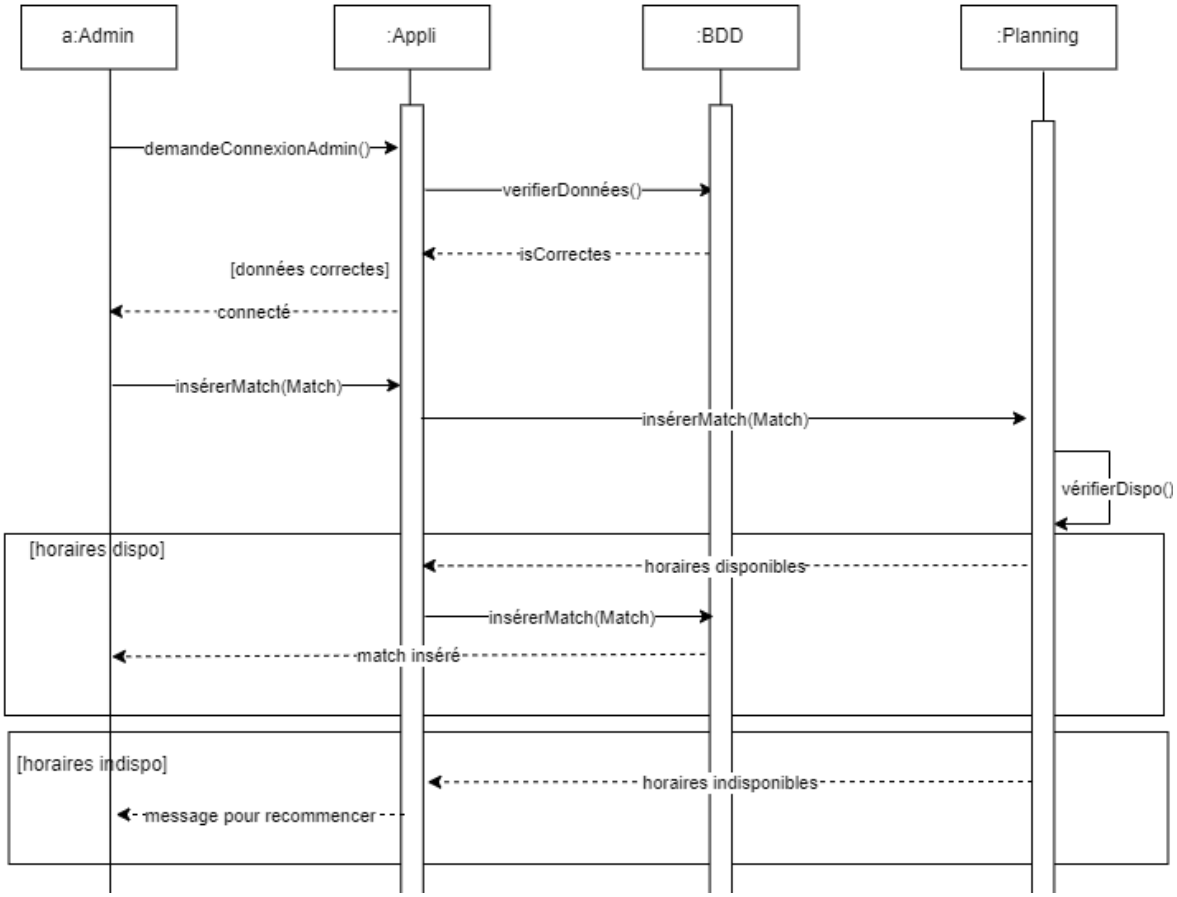
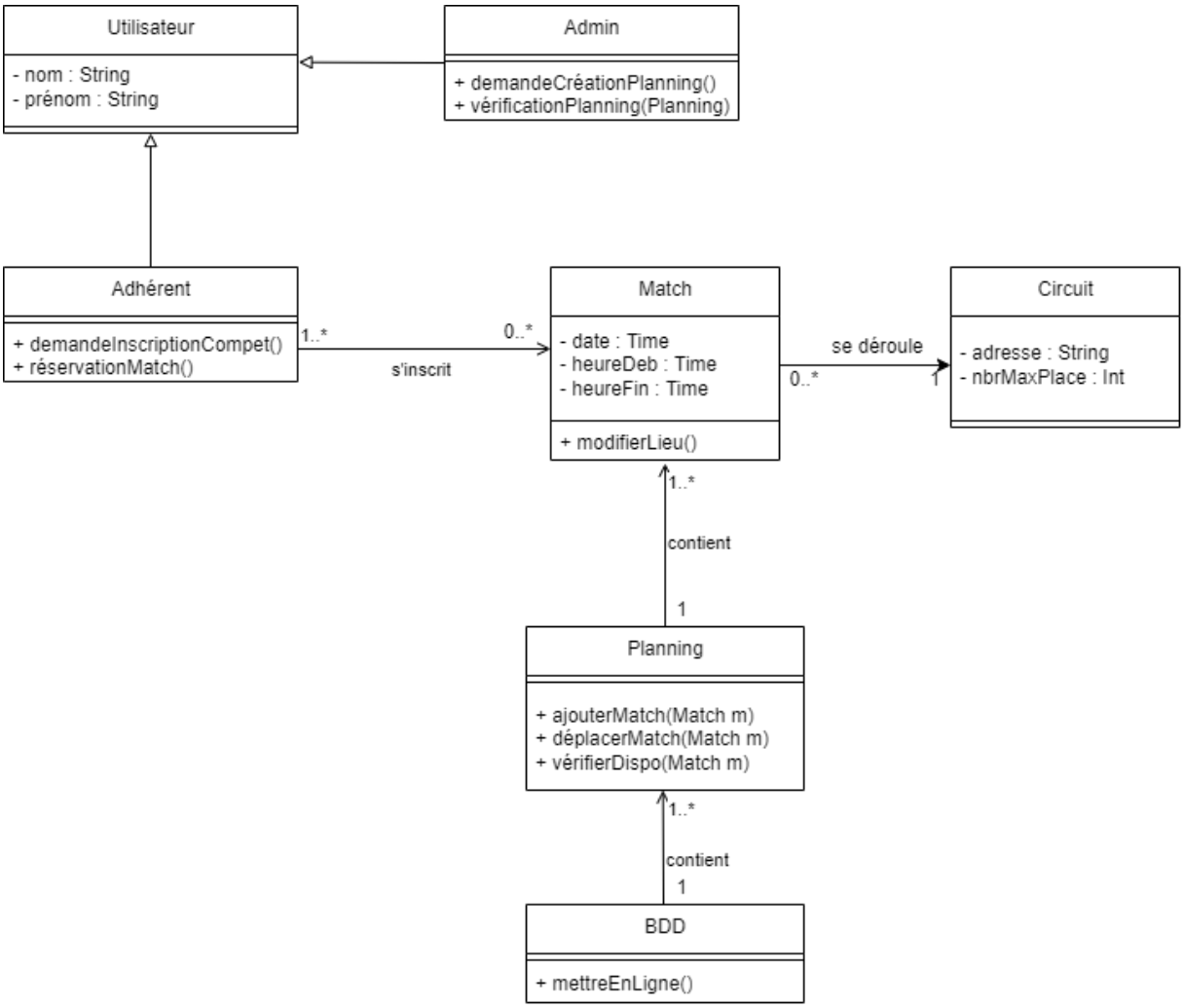


Diagramme de classe :

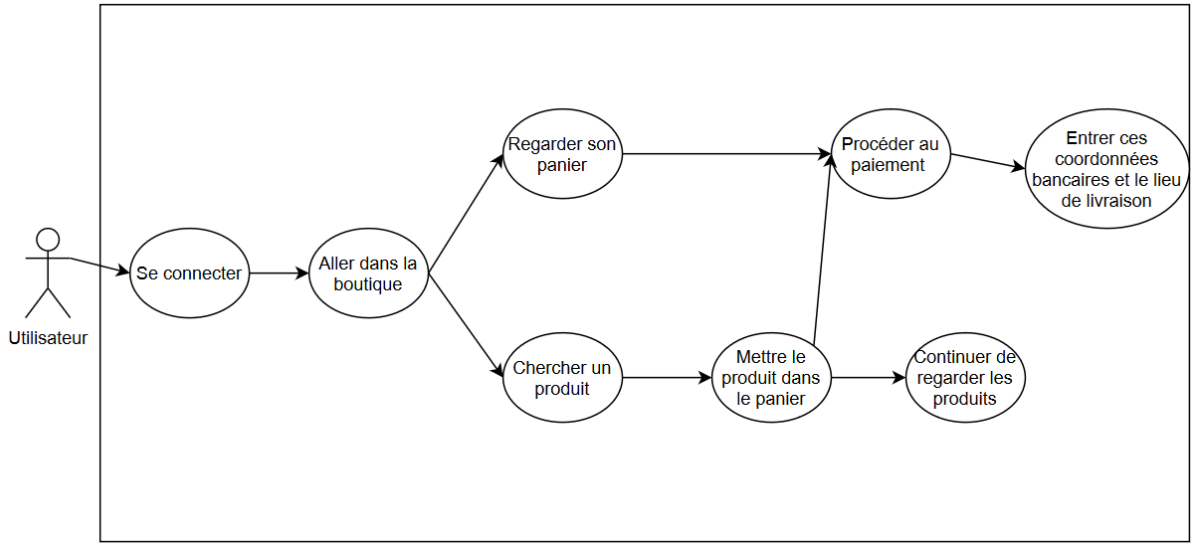


Ce diagramme nous montre toutes les classes reliées au fonctionnement du planning. Par exemple, on peut voir qu'un planning est fait d'une multitude de matchs, que l'on peut ajouter un match dedans ou que l'on peut en déplacer. On remarque aussi que les adhérents peuvent s'inscrire dans autant de matchs qu'ils le veulent et qu'ils peuvent aussi réserver des courts en illimité.

De plus, dans chaque match, il y aura un tableau d'adhérents représentant les participants.

Boutique :

Cas d'utilisation :

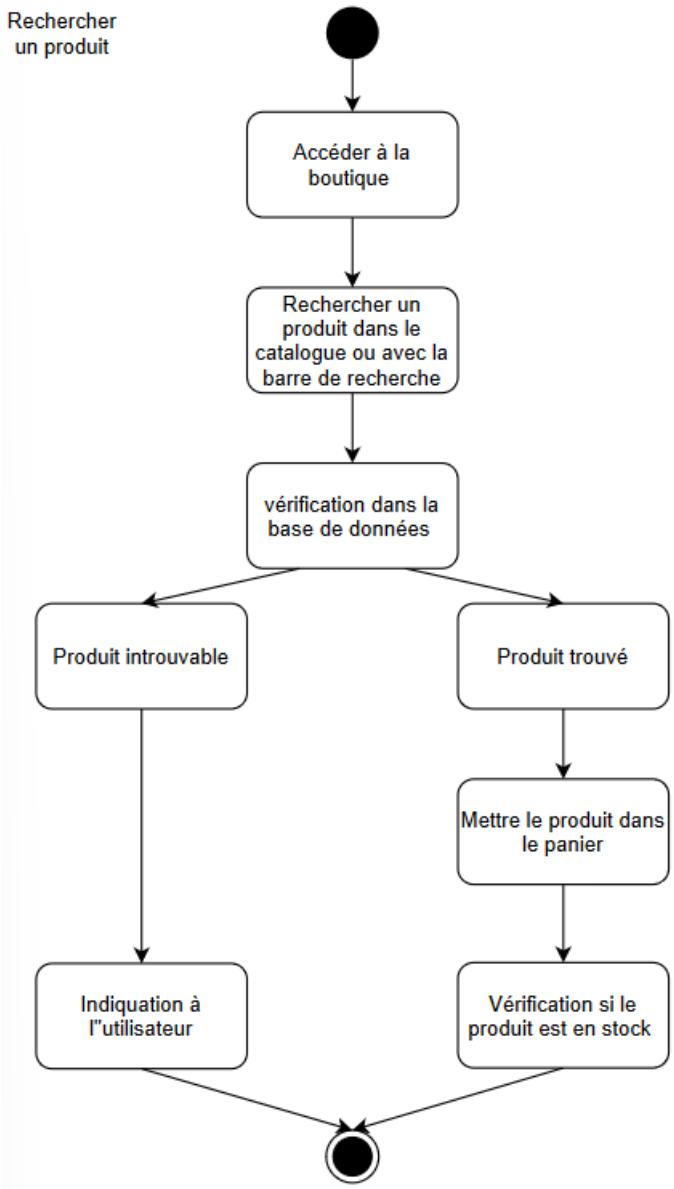


Ce diagramme nous aide à comprendre comment le système d'achat ou de recherche de produit de la boutique fonctionnera. Comme nous pouvons le constater, il faudra être obligatoirement connecté pour pouvoir accéder aux fonctionnalités de paiements. Un utilisateur pourra chercher un produit, ou plusieurs produits, qu'il pourra ensuite mettre dans son panier et s'il le souhaite, procéder au paiement en rentrant ses coordonnées bancaires ainsi que le lieu où il souhaite se faire livrer, comme son domicile ou un point relai.

Il faudra faire très attention à la protection des données sensibles de l'utilisateur.

Recherche d’un produit

Diagramme d’activité



Ici, nous voyons qu’à chaque recherche et ou le fait d’accéder à un produit, une demande est faite à la base de données pour vérifier que ce produit est bien accessible. Un message est envoyé à l’utilisateur si le produit est inexistant ou manquant dans les stocks.

Scénarios

1. L’utilisateur accède à la boutique
2. L’utilisateur cherche un article dans le catalogue ou avec la barre de recherche
3. La base de données envoie à l’utilisateur que le produit existe
4. L’utilisateur accède au produit
5. La base de données vérifie si le produit est en stock

---

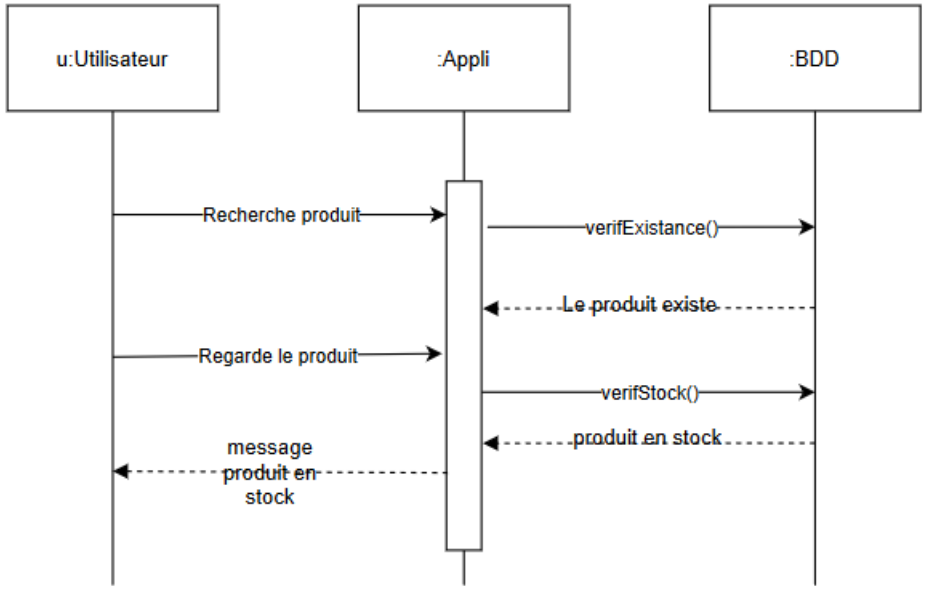
1. L’utilisateur accède à la boutique
2. L’utilisateur cherche un article dans le catalogue ou avec la barre de recherche
3. La base de données envoie à l’utilisateur que le produit n’existe pas
4. L’application affiche un message pour prévenir que le produit n’existe pas

---

1. L’utilisateur accède à la boutique
2. L’utilisateur cherche un article dans le catalogue ou avec la barre de recherche
3. La base de données envoie à l’utilisateur que le produit existe
4. L’utilisateur accède au produit
5. La base de données vérifie les stocks et le produit n’est pas en stock
6. L’application envoie à l’administrateur qu’un produit manque
7. L’administrateur envoie à l’utilisateur que le produit arrive bientôt

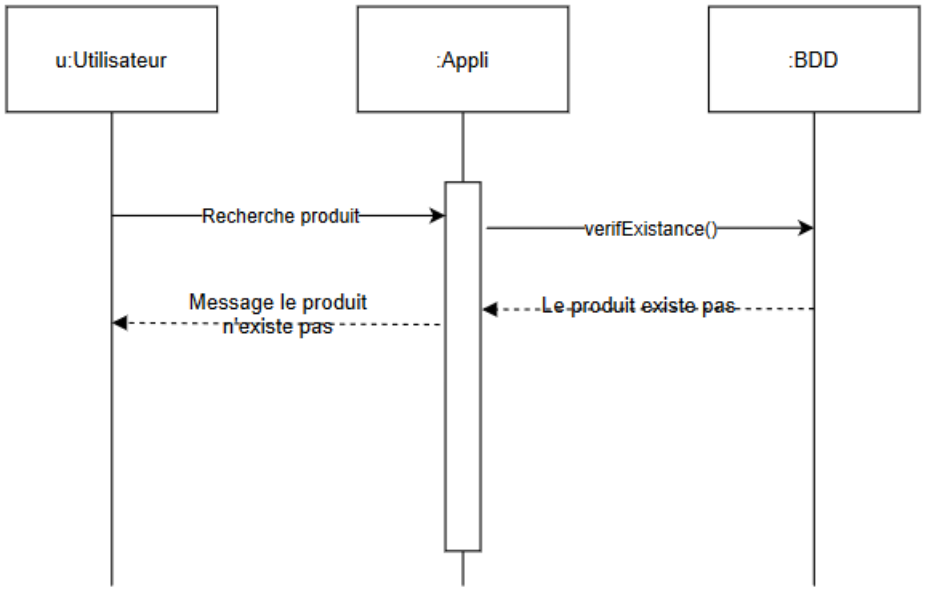
Diagramme de séquence

Scénario 1



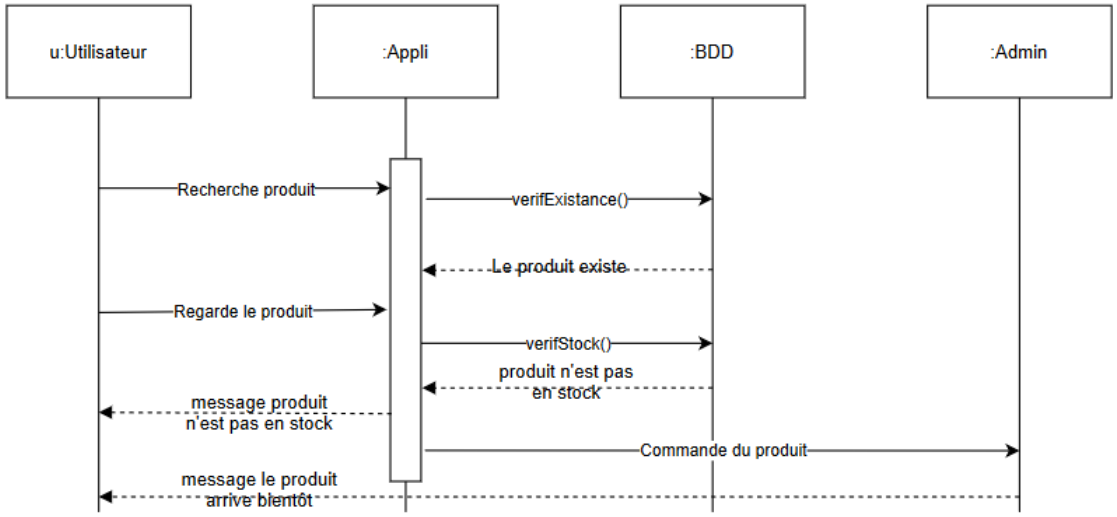
Ce scénario représente le cas parfait où l'utilisateur recherche un produit et celui-ci est trouvable et est en stock.

Scénario 2



Dans ce scénario, le produit n’existe pas et l’application prévient donc l’utilisateur avec un message pour dire que le produit n’existe pas.

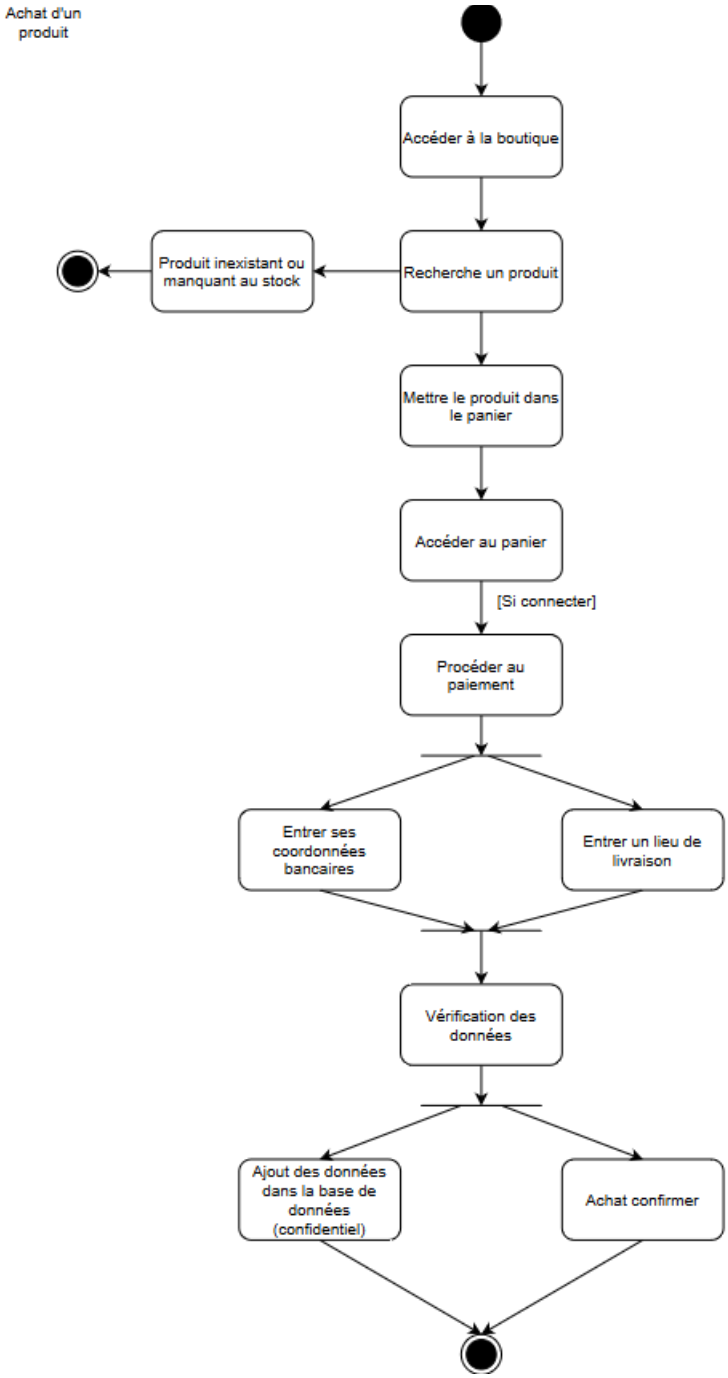
Scénario 3



Dans ce scénario, le produit existe mais son stock est vide, une demande est envoyée à un administrateur pour faire une commande et prévenir l’utilisateur que le produit arrive bientôt.

Achat d’un produit

Diagramme d’activité



Dans ce diagramme, nous pouvons voir comment le processus de paiement est fait.

L’utilisateur va rechercher un produit puis la base de données va vérifier si le produit existe ou s’il est disponible. L’utilisateur va le mettre dans son panier et procéder au paiement en entrant ces coordonnées bancaires et le lieu de livraison. Ces coordonnées seront cryptées et enregistrée dans la base de données.

Scénarios

1. L’utilisateur cherche un produit
2. La base de données vérifie l’existence de celui-ci
3. L’utilisateur met le produit dans son panier
4. L’utilisateur accède au panier
5. L’utilisateur procède au paiement
6. L’utilisateur entre ses coordonnées bancaires et le lieu de livraison
7. Vérification des données bancaires
8. La base de données enregistre et crypte ces données
9. L’utilisateur reçoit une confirmation de son achat avec les informations

---

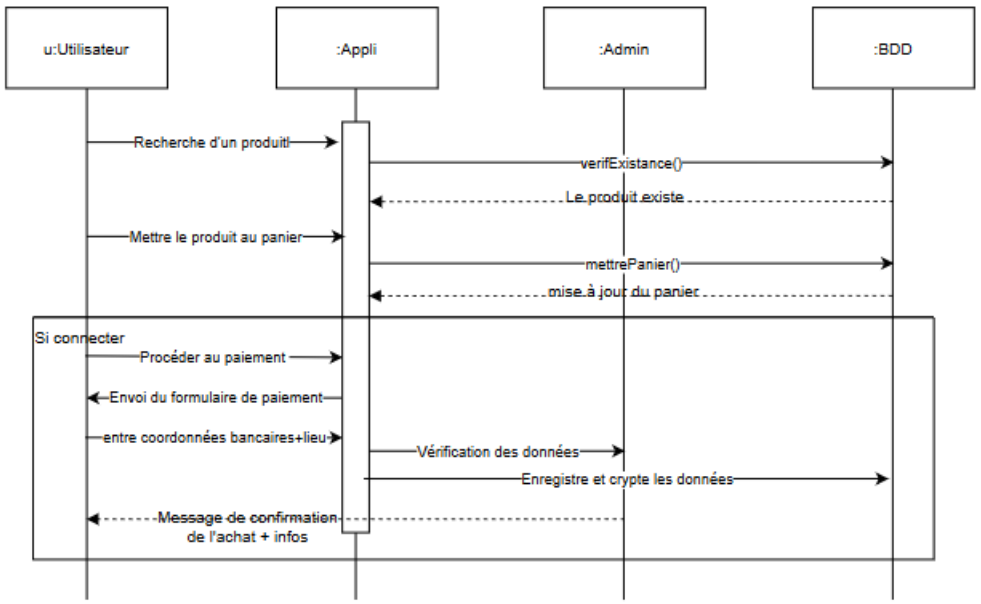
1. L’utilisateur cherche un produit
2. La base de données vérifie l’existence de celui-ci
3. Le produit n’existe pas ou est manquant
4. Un message est envoyé à l’utilisateur

---

1. L’utilisateur cherche un produit
2. La base de données vérifie l’existence de celui-ci
3. L’utilisateur met le produit dans son panier
4. L’utilisateur accède au panier
5. L’utilisateur procède au paiement
6. L’utilisateur entre ses coordonnées bancaires et le lieu de livraison
7. Les coordonnées bancaires sont erronées
8. Un message d’erreur est envoyé à l’utilisateur

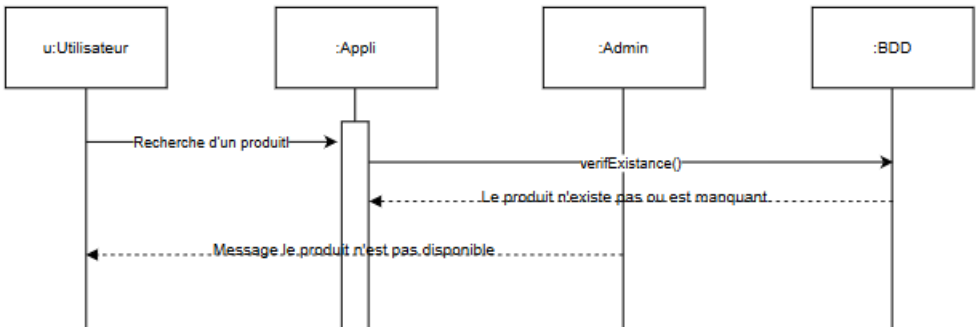
Diagramme de séquence

Scénario 1



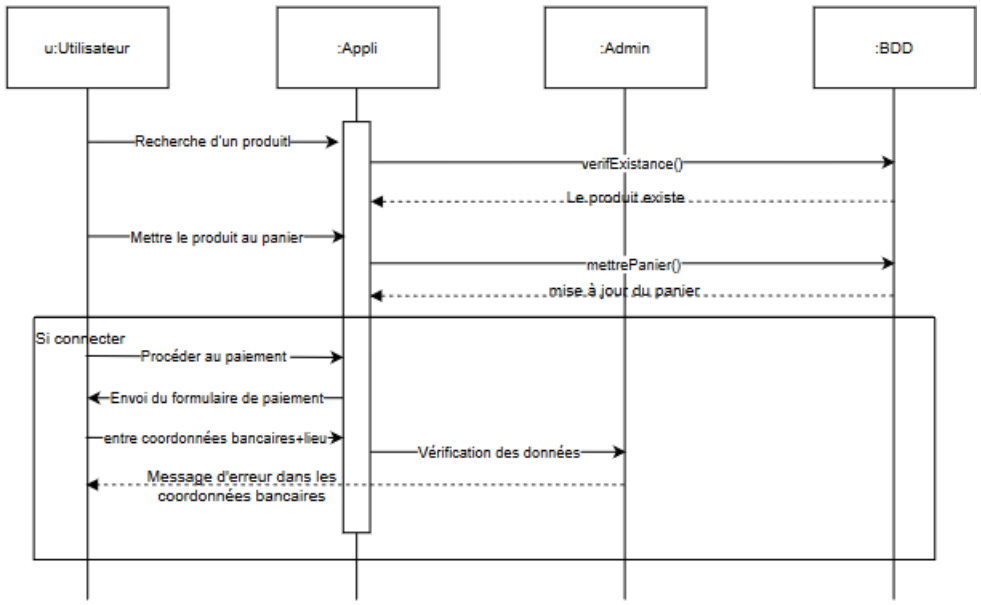
Dans ce cas-là, l'utilisateur trouve un produit qui existe et est disponible. Celui-ci le met dans son panier et procède au paiement, où tout se passe bien et reçoit bien le message de confirmation d'achat.

Scénario 2



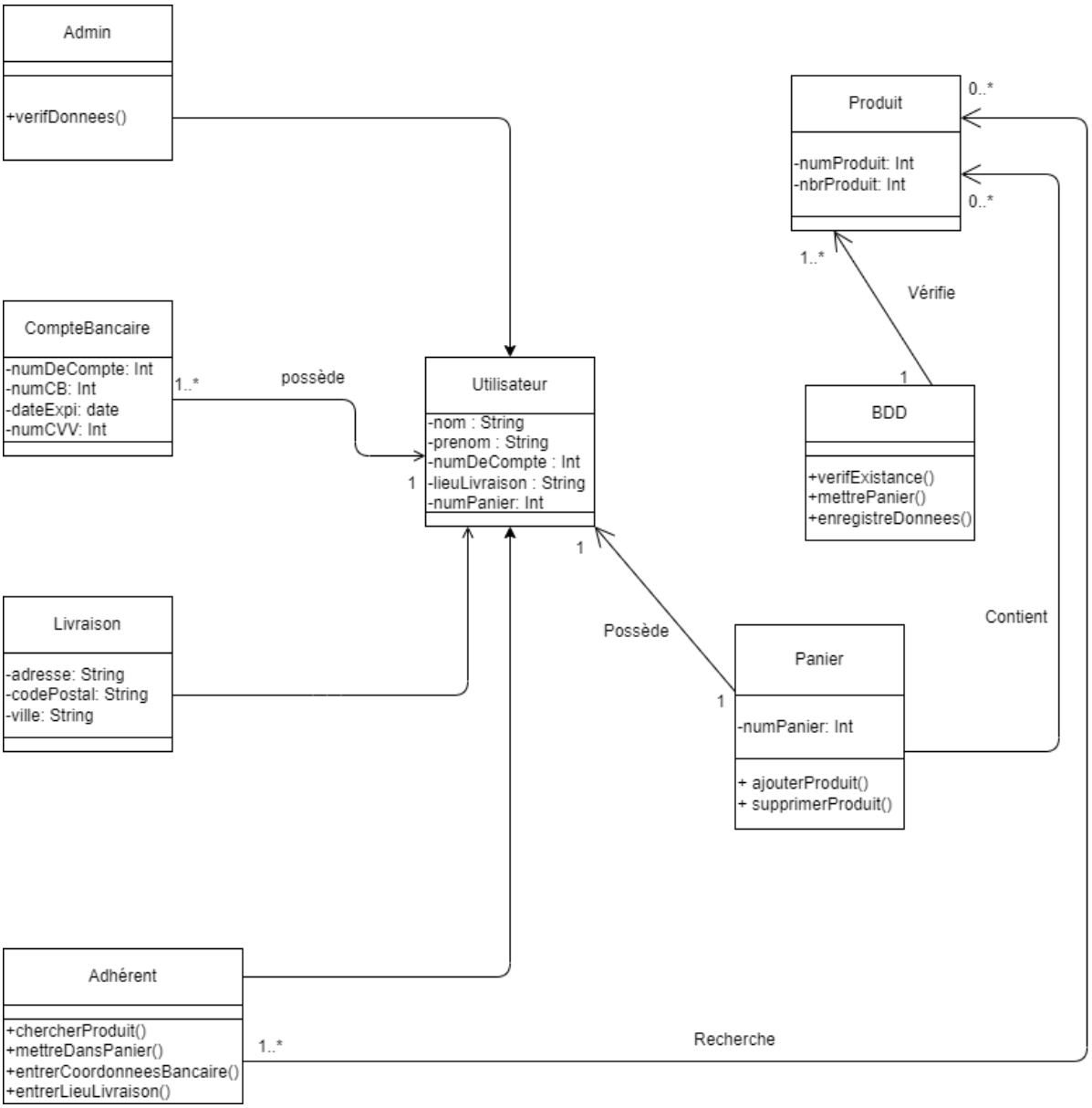
Le scénario 2 représente le cas où le produit n'existe pas où n'est pas disponible, un message lui est alors envoyé pour le prévenir.

Scénario 3



Ici, tout le procédé de recherche et de mise au panier se passe bien mais c'est au paiement qu'il y a une erreur dans les coordonnées bancaires. Un administrateur va donc vérifier et envoyer un message d'erreur à l'utilisateur.

Diagramme de classe

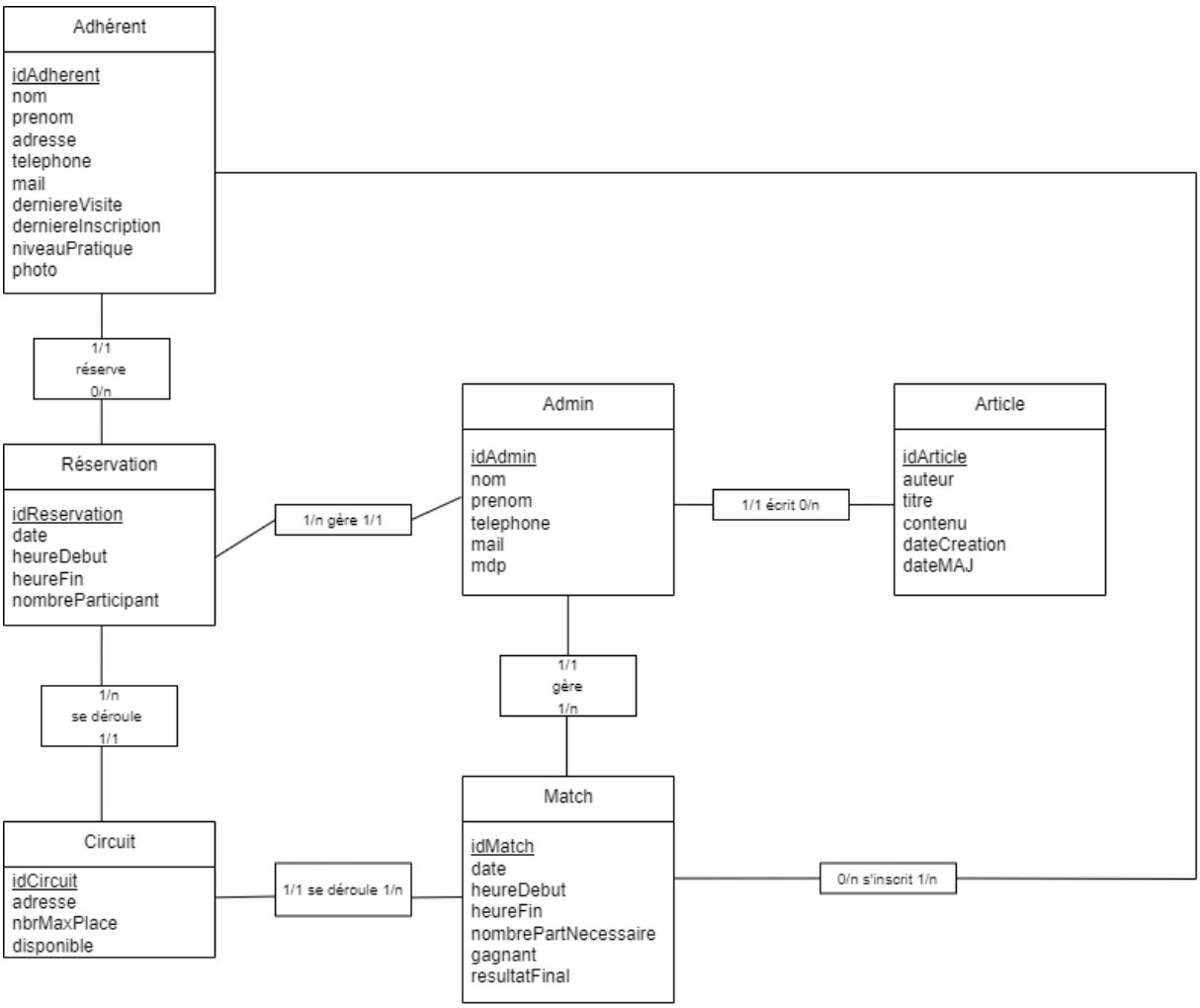


Partie IHM

Partie Base de données

Dans cette partie, nous analyserons la conception de la base de données ainsi que quelques scripts SQL permettant la création de tables relationnelles. Cela aidera à comprendre quelles sont les données persistantes qui seront utilisées par tous les modules.

Modèle Entités/Associations



Ce modèle décrit comment fonctionnera la base de données. Quelques informations ont déjà été donné avant, comme le fait qu'un adhérent puisse réserver et s'inscrire à plusieurs matchs en même temps, à condition qu'ils se déroulent à des horaires différents.

Mais nous avons des informations en plus, comme le fait qu'un administrateur puisse rédiger des articles, pour annoncer les résultats ou pour annoncer d'autres informations.

Nous avons aussi la table circuit, permettant de gérer plus facilement les circuits disponibles. Par exemple, si un circuit est momentanément indisponible (pour cause de travaux ou autres), il suffit de changer le statut du circuit et ainsi il ne sera plus possible d'ajouter des matchs sur celui-ci.

Script SQL

Voici les scripts SQL pour la création de la table Match et Adhérent :

Adhérent →

```
CREATE TABLE IF NOT EXISTS `mydb`.`Adhérent` (
  `idAdhérent` INT NOT NULL AUTO_INCREMENT,
  `nom` VARCHAR(45) NOT NULL,
  `prenom` VARCHAR(45) NOT NULL,
  `adresse` VARCHAR(120) NOT NULL,
  `telephone` VARCHAR(12) NOT NULL,
  `mail` VARCHAR(45) NOT NULL,
  `dernièreVisite` DATE NULL,
  `dernièreInscription` DATE NULL,
  `niveauPratique` VARCHAR(45) NULL,
  `photo` BLOB NULL,
  PRIMARY KEY (`idAdhérent`))
ENGINE = InnoDB;
```

Match →

```
CREATE TABLE IF NOT EXISTS `mydb`.`Match` (
  `idMatch` INT NOT NULL AUTO_INCREMENT,
  `date` DATE NOT NULL,
  `heureDebut` TIME NOT NULL,
  `heureFin` TIME NOT NULL,
```

```
  `nbrPartNecessaire` INT NOT NULL,
  `gagnant` INT NULL,
  `resultatFinal` VARCHAR(45) NULL,
  `Circuit_idCircuit` INT NOT NULL,
  `Admin_idAdmin` INT NOT NULL,
  `Adhérent_idAdhérent` INT NOT NULL,
  PRIMARY KEY (`idMatch`),
  INDEX `gagnant_idx` (`gagnant` ASC) VISIBLE,
  INDEX `fk_Match_Circuit1_idx` (`Circuit_idCircuit` ASC) VISIBLE,
  INDEX `fk_Match_Admin1_idx` (`Admin_idAdmin` ASC) VISIBLE,
  INDEX `fk_Match_Adhérent1_idx` (`Adhérent_idAdhérent` ASC) VISIBLE,
  CONSTRAINT `gagnant`
    FOREIGN KEY (`gagnant`)
    REFERENCES `mydb`.`Adhérent` (`idAdhérent`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Match_Circuit1`
    FOREIGN KEY (`Circuit_idCircuit`)
    REFERENCES `mydb`.`Circuit` (`idCircuit`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Match_Admin1`
    FOREIGN KEY (`Admin_idAdmin`)
    REFERENCES `mydb`.`Admin` (`idAdmin`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Match_Adhérent1`
    FOREIGN KEY (`Adhérent_idAdhérent`)
    REFERENCES `mydb`.`Adhérent` (`idAdhérent`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Dans cette table, nous voyons qu'elle possède plusieurs clés étrangères. En effet, pour chaque match, il faut plusieurs participants, un circuit et un gagnant.