Glossaire

- Cybersquattage : Pratique consistant à occuper des noms de domaine correspondant à un domaine déjà existant pour plusieurs raisons : profiter de sa notoriété
- Framework : Ensemble de composant logiciels ou outils formant une bibliothèque.
- JPA : Java Persistence API. C'est une interface permettant d'organisée des données relationnelles.
- Responsive : Technique de conception de pages qui fait en sorte que l'affichage s'adapte automatiquement aux différents appareils (mobile / tablette / site web).
- SPA : Single Pages Application en anglais. Cela signifie qu'une seule page web est accessible et la moindre action utilisateur est récupérée et affichée dynamiquement.
- URL : Uniform Resource Locator en anglais. Cela correspond à la localisation d'une page d'un site web.

Sommaire

Introduction

Dans le cadre de l'unité d'enseignement GLG204 - Architectures Logicielles Java niveau 2, il y a une réalisation d'un projet, ainsi que sa soutenance. Ce dernier a pour objectifs de mettre à l'épreuve mes compétences acquises lors de cette formation d'ingénieur, ainsi que mes capacités d'analyses et de prise de recul.

Ce projet s'inscrit pour ma part sur une fin d'études du diplôme d'ingénieur en informatique : Architecture et intégration des systèmes et des logiciels (CYC9101A). La société décrite dans ce projet est une société fictive dont les besoins ont été imaginés par mes soins.

Dans un premier temps, nous allons définir l'expression des besoins de la société "Temple Félidé". Sur la base de cette expression, nous effectuerons l'analyse de ses besoins, puis nous déterminerons l'architecture et les technologies à mettre en place.

Expression des besoins

Les objectifs de cette section sont de décrire les besoins du Temple Félidé, en établissant les critères de sélection, en favorisant l'aspect concurrentiel et en facilitant la prise de décision. Ces objectifs sont essentiels pour aider l'entreprise à trouver les solutions les mieux adaptées à ses besoins et à ses contraintes, afin de toucher le plus de clients potentiels.

Pour ce faire, nous allons présenter le projet, la société, ainsi que ses acteurs. Nous décrirons les besoins au travers de diagrammes de cas d'utilisation.

2. Présentation

2.1 Présentation du projet

Le Temple félidé souhaite mettre en place un site responsive de garde de chats. Ce site permettrait de mettre en relation des particuliers, souhaitant faire garder leurs animaux soit par des particuliers amoureux d'animaux ou par des professionnels formés.

2.2 Situation actuelle

La société ne possède actuellement aucun site et fonctionne par le bouche-à-oreille, ainsi que sur papier.

2.3 Les contraintes

- Le site doit également être consultable en mobile au vu des usages actuels
- Le site doit être plaisant et centrer autour de la réservation

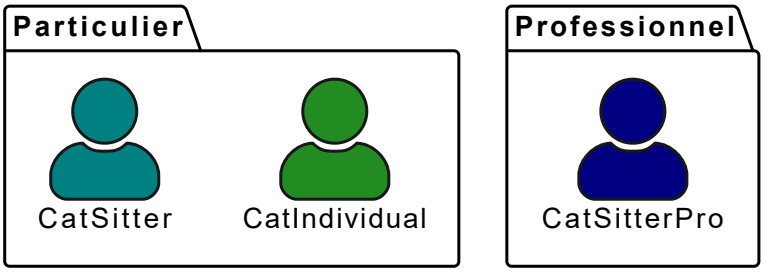
2.4 Présentation de la société

La société le Temple félidé est une garderie professionnelle dédié aux chats. Le personnel de l'entreprise ont tous une formation d'assistant vétérinaire et sont passionnés par les félins. Les espaces intérieurs et extérieurs sont aménagés et pensés pour le plaisir et le confort de nos amis les chats. Les chambres individuelles et familiales sont imaginés pour satisfaire les félins les plus exigeant. Le gérant souhaitait mettre en place un site pour faciliter les réservations et passer sur un planning en ligne.

Il a commencé à élaborer une réflexion autour de son besoin avec l'entreprise (Nous) acceptant de réaliser son site. Dans sa réflexion, il a pris en compte que leur garderie est souvent complète à certaines périodes. De nombreux particuliers se retrouvant sans solution pour faire garder leurs chats. Après réflexion ils se sont dit que créer un site qui mettrait en relation des particulier et des professionnels permettrait de résoudre cette problématique ainsi que celle de la réservation et du planning. Cela permet également de générer des partenariats privilégiés. De plus, il souhaite mettre en avant l'intérêt de passer par un professionnel formé et possédant des espaces adaptés pour accueillir des chats. Le gérant imaginait aussi ajouter de nombreuses fonctionnalités par la suite et ainsi lotir son besoin.

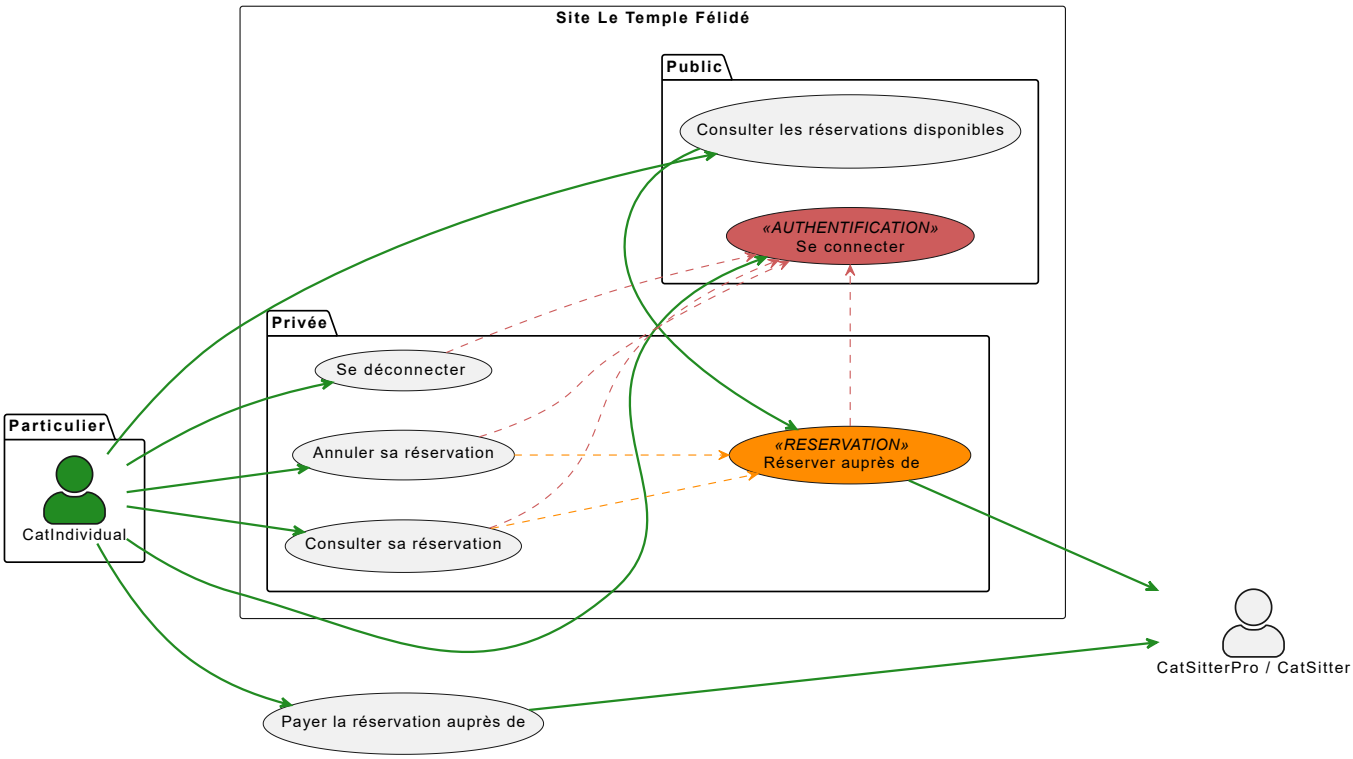
3. Les acteurs

- Le particulier souhaitant faire garder son/ses chat(s) sera nommé : CatIndividual
- Le particulier proposant de garder un ou plusieurs chats sera nommé : CatSitter
- Le professionnel sera nommé : CatSitterPro



4. Cas d'utilisations

4.1 Réservation d'un particulier



4.1.1 Réserver une garde de mon chat auprès d'un particulier ou d'un professionnel

Résumé

Le particulier souhaite faire garder son animal et pour ce faire sélectionne un particulier ou un professionnel sur le créneau de son choix. Puis il s'authentifie, afin de finaliser la réservation du créneau préalablement sélectionné.

Acteurs

- Le particulier (CatIndividual)

- Le particulier proposant du gardiennage (CatSitter)
- Le professionnel (CatSitterPro)

Pré-conditions

- Le particulier doit avoir un compte.
- Le particulier et le professionnel proposant leurs services de garde doivent obligatoirement avoir créé un compte et donner leurs disponibilités au préalable. Dans les deux cas, se référer au cas d'utilisation « Gestion d'un compte ».

Description

1. Le particulier consulte les réservations
2. Le particulier sélectionne un créneaux disponible correspondant à un particulier ou à un professionnel
3. Il s'authentifie avec son login et son mot de passe
4. Il est redirigé vers sa réservation et la confirme
5. La réservation est effective

Exceptions

L'utilisateur ne s'authentifie pas : Les données renseignées sont incorrectes et ne permettent pas de s'authentifier.

La réservation est de 1 jour minimum : La plage de date choisie doit s'étaler sur au minimum 1 jour.

La date choisie est antérieur à la date du jour: La réservation doit explicitement être faite dans le futur.

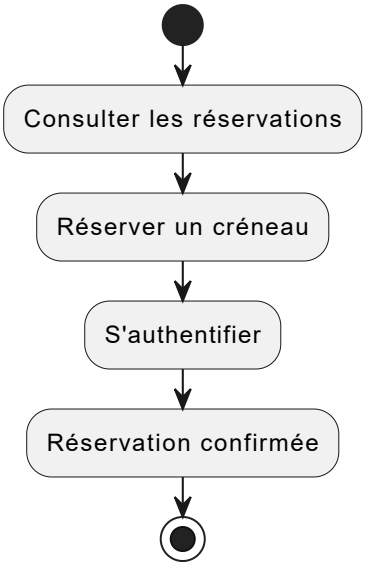
La date choisie ne peut être supérieur à 1 an: La réservation ne peut se faire au dela d'une année.

Le créneau n'existe plus : Une autre personne a reservé le créneau avant l'utilisateur courant.

Post-conditions

La réservation est confirmée.

Diagramme d'activité



4.1.2 Consulter ma réservation

Résumé

Le particulier souhaite consulter sa réservation afin de vérifier des informations ou les dates choisies.

Acteurs

- Le particulier (CatIndividual)

Pré-conditions

- Le particulier doit être authentifié.
- Le particulier doit avoir un compte.
- Le particulier doit avoir effectué une réservation au préalable.

Description

1. Le particulier clique sur son icône de profil
2. Il clique sur le menu "Mes réservations"
3. Il accède à ses créneaux réservés

Exceptions

Le particulier ne s'est pas authentifié : Le menu lui permettant de consulter les réservations n'est pas visible en tant qu'anonyme

Le particulier n'a effectué aucune réservation : La page lui indique qu'il n'a aucune réservation en cours.

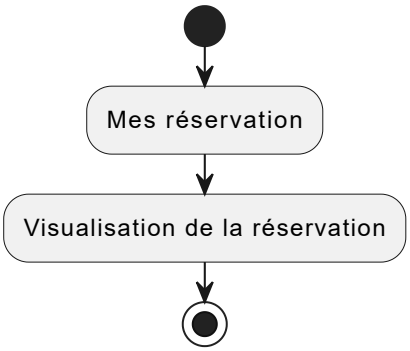
Post-conditions

Le particulier a pu voir les informations concernant sa réservation.

Remarques

Le particulier peut voir en dessous de sa réservation en cours ou s'il n'a aucune réservation, l'historique de ses réservations passées.

Diagramme d'activité



4.1.3 Annuler ma réservation

Résumé

Le particulier souhaite pour des raisons personnelles annuler sa réservation. En rebond de cette annulation, une notification doit être envoyé à l'utilisateur qui effectuait le gardiennage de l'animal.

Acteurs

- Le particulier (CatIndividual)
- Le particulier proposant du gardiennage (CatSitter)
- Le professionnel (CatSitterPro)

Pré-conditions

- Le particulier doit être authentifié.
- Le particulier doit avoir un compte.
- Le particulier doit avoir effectué une réservation au préalable.

Description

1. Le particulier clique sur son icône de profil
2. Il clique sur le menu "Mes réservations"
3. Il accède à ses créneaux réservés
4. Il sélectionne le créneau à annuler
5. Il annule la réservation
6. La réservation n'est plus visible dans son espace
7. Une notification d'annulation est envoyée au professionnel ou au particulier devant garder l'animal

Exceptions

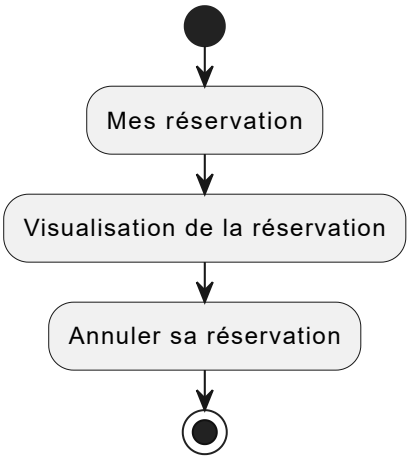
Le particulier ne s'est pas authentifié : Le menu lui permettant de consulter les réservations n'est pas visible en tant qu'anonyme

Le particulier n'a effectué aucune réservation : La page lui indique qu'il n'a aucune réservation en cours.

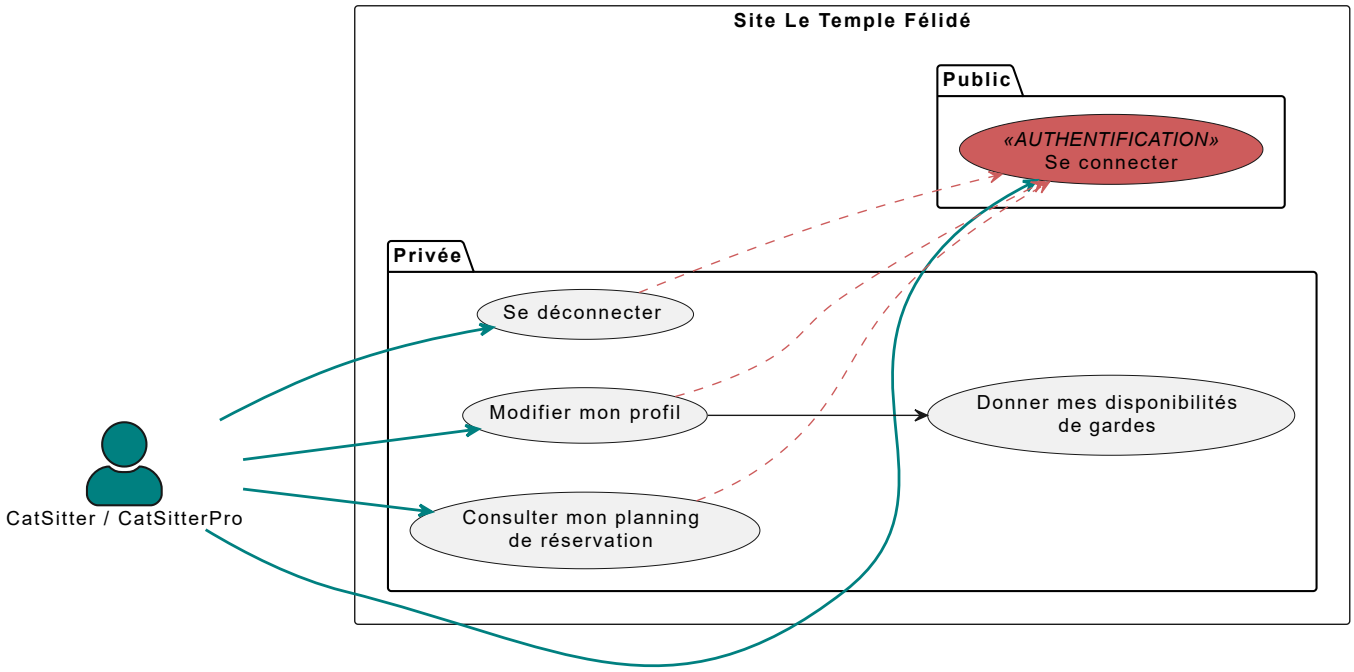
Post-conditions

Le particulier a pu annuler sa réservation.

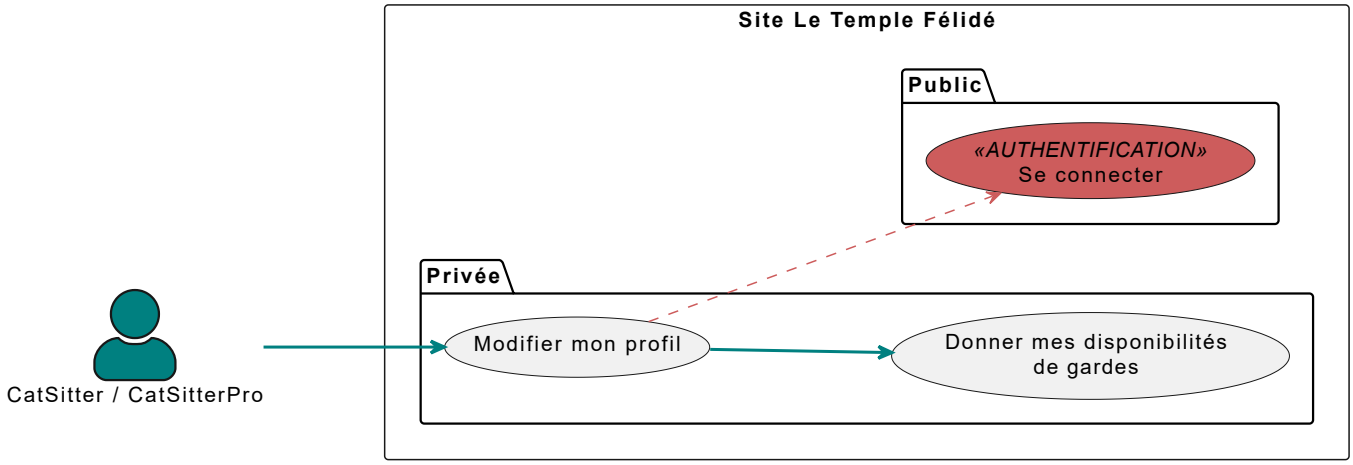
Diagramme d'activité



4.2 Gestion d'un CatSitter



4.2.1 Donner mes disponibilités de gardes



Résumé

Un utilisateur connecté peut indiquer les plages de jours où il est disponible pour garder des félins.

Acteurs

- Le particulier proposant du gardiennage (CatSitter)
- Le professionnel (CatSitterPro)

Pré-conditions

- L'utilisateur doit avoir un compte
- L'utilisateur doit être connecté

Description

1. L'utilisateur clique sur le menu "Modifier mon profil"
2. il se rend sur la page d'authentification
3. Il fournit son identifiant et son mot de passe
4. Le système vérifie la concordance des données rentrées et attendues
5. L'utilisateur est redirigé vers la page "Mon profil" en étant connecté
6. Il a un encart dédié à l'ajout de ses disponibilités avec un planning
7. Il remplit ses disponibilités

Exceptions

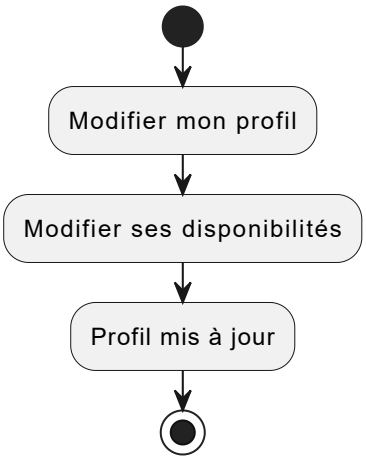
Donner une disponibilité en doublon : Si l'utilisateur a déjà donné cette disponibilité, alors un message d'alerte s'affiche pour doublon et l'oblige à changer sa saisie

Se rendre indisponible sur un créneau déjà réservé : L'utilisateur avait renseigné au préalable sa disponibilité et un particulier a réservé ce créneau. Une notification lui indique une réservation en cours sur cette plage et lui demande confirmation quant à son annulation. L'utilisateur peut soit confirmer la suppression de cette réservation et le particulier sera notifié par mail, soit ne rien faire et garder le créneau disponible.

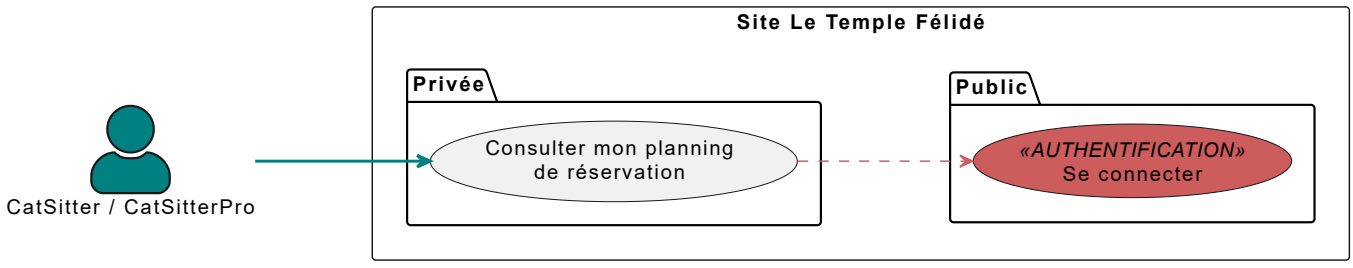
Post-conditions

Son planning est mis à jour et permet la réservation d'un particulier

Diagramme d'activité



4.2.2 Consulter mon planning de réservation



Résumé

Un utilisateur connecté peut consulter son planning de réservation et voir les jours où il effectuera des gardes et pour quel animal ainsi que le propriétaire.

Acteurs

- Le particulier proposant du gardiennage (CatSitter)
- Le professionnel (CatSitterPro)

Pré-conditions

- L'utilisateur doit avoir un compte
- L'utilisateur doit être connecté
- L'utilisateur doit avoir rempli ses disponibilités de gardes

Description

1. L'utilisateur clique sur le menu "Mon planning"
2. il se rend sur la page d'authentification
3. Il fournit son identifiant et son mot de passe
4. Le système vérifie la concordance des données rentrées et attendues
5. L'utilisateur est redirigé vers la page "Mon planning" en étant connecté
6. Il consulte son planning de réservation

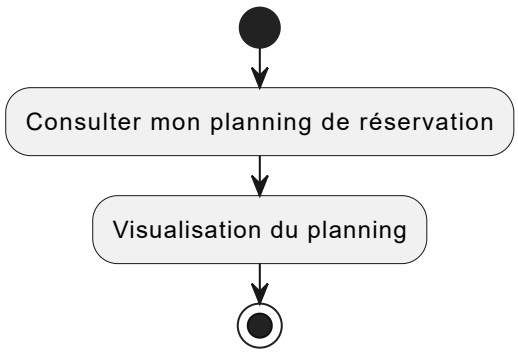
Exceptions

L'utilisateur n'a pas rempli ses disponibilités : Un texte lui indique qu'il ne propose pas ses services de gardiennage et lui propose s'il le souhaite d'être redirigé vers la page dédié à l'ajout de ses disponibilités.

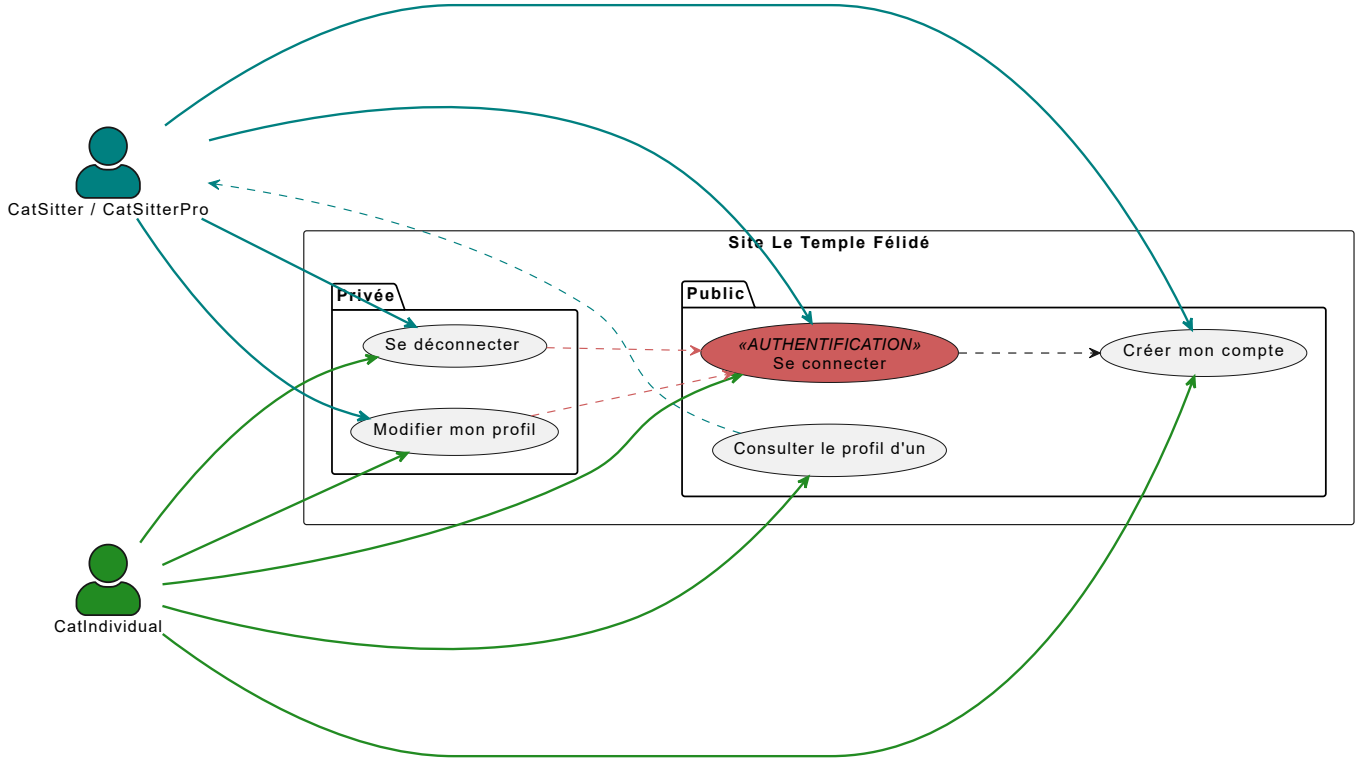
Post-conditions

Il consulte son planning de réservation

Diagramme d'activité



4.3 Gestion d'un compte



4.2.3 Créer mon compte

Résumé

Permet à un utilisateur de créer son compte sur le site.

Acteurs

Tous les acteurs du site.

Pré-conditions ❌

Description

1. L'utilisateur se rend sur la page d'authentification
2. L'utilisateur clique sur le bouton création de compte
3. Il est redirigé vers la page de création de compte
4. Si c'est un professionnel il clique sur la case à cocher correspondante
5. Il rentre son numéro de siret
6. Il rentre toutes les informations obligatoires et facultatives s'il le souhaite
7. Le système vérifie le format des données et sauvegarde le compte
8. L'utilisateur est redirigé vers la page d'authentification en lui indiquant que son compte a bien été créer et l'invite à se connecter

Exceptions

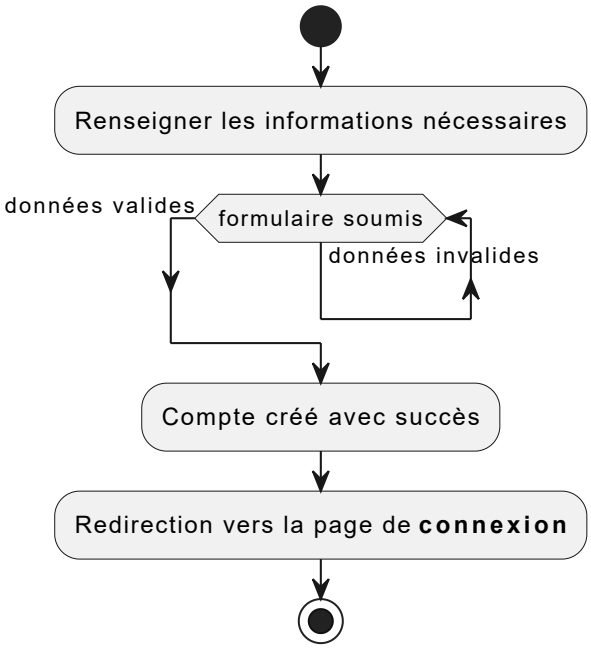
Utilisateur déjà connu : L'utilisateur renseigne toutes ses données et le système constate que l'identifiant est déjà connu. L'utilisateur est prévenu qu'un email lui a été envoyé et il s'agit du même mail que celui de la réinitialisation du mot de passe, afin d'éviter de donner une information à un potentiel attaquant.

Les données ne sont pas au bon format : Les informations renseignés par l'utilisateur sont incorrecte, il peut modifier ses informations et relancer la création de son compte.

Post-conditions

L'utilisateur a désormais un compte.

Diagramme d'activité



4.2.4 Se connecter

Résumé

Permet à un utilisateur de s'authentifier sur le site.

Acteurs

Tous les acteurs du site.

Pré-conditions

L'utilisateur doit avoir un compte sur le site

Description

1. L'utilisateur se rend sur la page d'authentification
2. L'utilisateur fournit son identifiant et son mot de passe
3. Le système vérifie la concordance des données rentrées et attendues

4. L'utilisateur est redirigé vers la page qu'il souhaitait accéder en étant connecté

Exceptions

L'identifiant ou le mot de passe sont mal renseignés : Les informations renseignés par l'utilisateur sont incorrectes, il peut modifier ses informations et relancer l'authentification

L'utilisateur a effectué trois tentatives de connexions infructueuses : Le nombre de tentatives de connexion a verrouillé l'utilisateur. Le système lui indique de procéder à la récupération de son compte en passant par mot de passe oublié ou attendre que son compte se déverrouille.

Utilisateur non connu : Soit l'utilisateur constate qu'il ne peut se connecter, car il n'a pas créé de compte, il peut alors effectuer la création. Sinon l'utilisateur effectue les trois tentatives infructueuses et on lui indique le même message malgré qu'il n'existe pas.

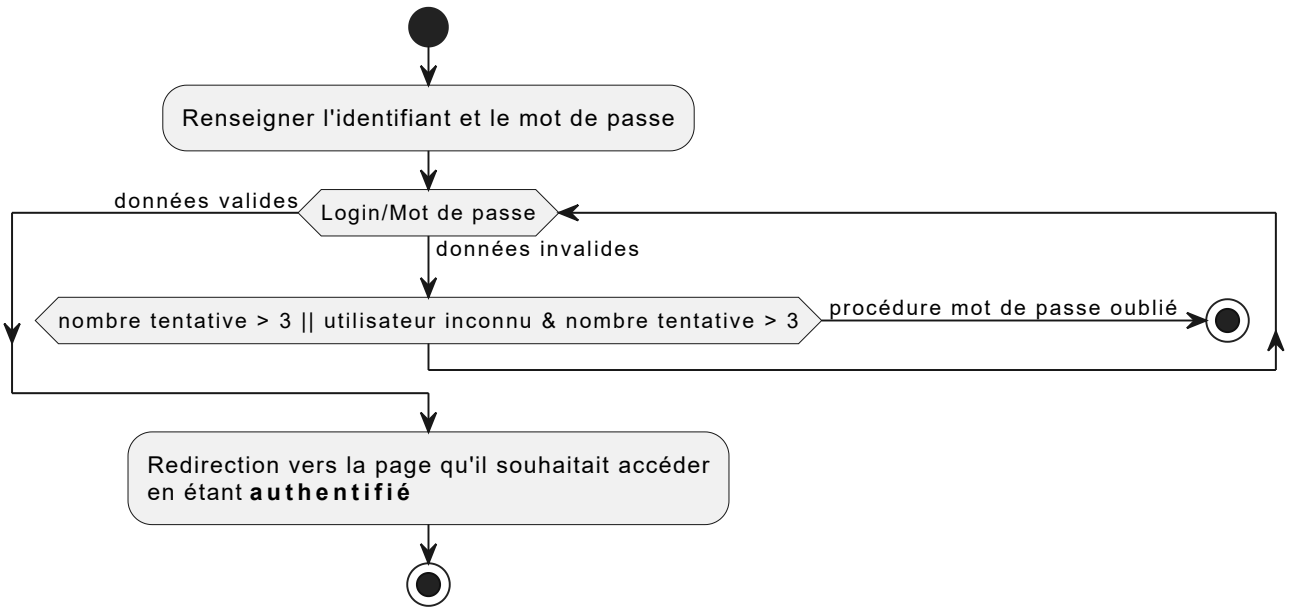
Post-conditions

L'utilisateur est authentifié.

Remarques

Si l'utilisateur n'existe pas, pour éviter de donner une information à une personne malintentionnée on indique à l'utilisateur d'effectuer la démarche "mot de passe oublié".

Diagramme d'activité



4.2.5 Se déconnecter

Résumé

Permet à un utilisateur de se déconnecter du site.

Acteurs

Tous les acteurs du site.

Pré-conditions

L'utilisateur doit s'être authentifié auparavant.

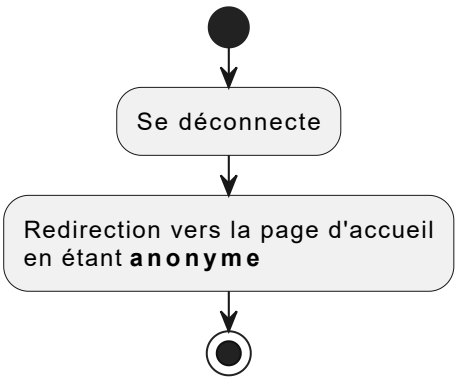
Description

1. L'utilisateur clique sur son profil
2. L'utilisateur clique sur le menu "Se déconnecter"
3. L'utilisateur est redirigé vers la page d'accueil sans être authentifié

Post-conditions

L'utilisateur est déconnecté.

Diagramme d'activité



4.2.6 Modifier mon profil

Résumé

Permet à un utilisateur de modifier son profil sur le site

Acteurs

Tous les acteurs du site.

Pré-conditions

L'utilisateur doit s'être authentifié auparavant.

Description

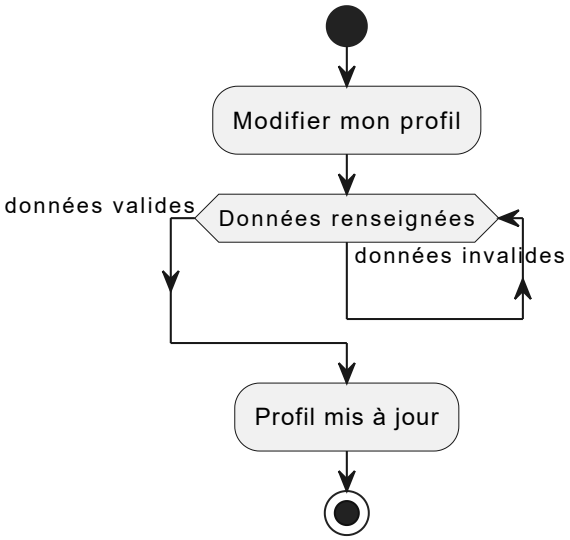
1. L'utilisateur clique sur son profil
2. Il clique sur le menu "Modifier son profil"
3. Il est redirigé vers la page de son profil en modification
4. Il modifie les données qu'il souhaite changer

5. Il valide les données changer

Post-conditions

Le profil de l'utilisateur a changé.

Diagramme d'activité



5. Lots non réalisés dans ce besoin

Numéro de lot	Description	Remarques
Lot 2	Intégration du profil "Mon(Mes) chat(s)"	Permet au CatSitter/CatSitter d'avoir le profil des animaux et mieux appréhender leurs besoins
Lot 2	Intégration des paiements	Sécuriser les paiments en ligne
Lot 2	Intégration des connexions type Google/Facebook	
Lot 3	Choix de la chambre / Pièce pour les professionnels	
Lot 3	Tarifs en fonction de la date de réservation	Les dates de vacances scolaires sont généralements plus chères qu'à d'autres périodes
Lot 3	Tarifs en fonction du choix de la pièce	Choisir une pièce plus grande type famille impacte le prix
Lot 4	Mise en place des notations entre utilisateurs	
Lot 5	Mise à niveau du site en marque blanche	Vendre le site à d'autres professionnels pour cibler les besoins par localité

Analyse des besoins

Dans la section précédente nous avons établi les exigences du Temple Félidé. Nous allons désormais nous concentrer sur l'analyse de ces besoins exprimés. Dans ce cadre, nous allons choisir les cas d'utilisations qui nous paraissent les plus pertinents et en déterminer les diagrammes de classes. En effet, nous obtiendrons ainsi nos principaux objets. Puis par la suite, nous allons modéliser les diagrammes de séquences et obtenir les relations entre chaque élément, ainsi que les comportements des objets.

Au sein de chaque cas d'utilisation, nous allons déterminer les objets candidats, les interactions entre ces objets, puis décrire les classes correspondantes.

Au travers de la méthode UML nous allons utiliser les catégories d'objets suivants :

- `<<Entity>>` Objet métier : informations durables et persistantes
- `<<Boundary>>` Objet à la frontière entre le système et un acteur
- `<<Control>>` Objet assurant une coordination d'autres objets (Façade entre objet `<<Boundary>>` et objet `<<Entity>>`)
- `<<Life cycle>>` Objet responsable de trouver les objets `<<Entity>>`

2. Cas d'utilisations

Les cas d'utilisation du Temple Félidé sont rappelés dans le sommaire suivant en précisant ceux qui sont analysés ou non.

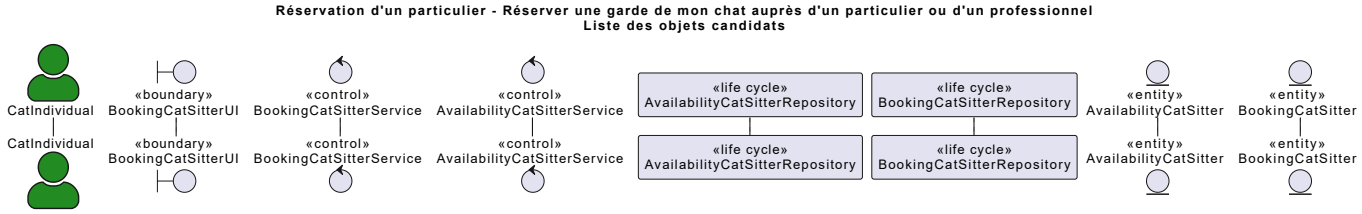
- Réservation d'un particulier
  - Réserver une garde de mon chat auprès d'un particulier ou d'un professionnel ☑
  - Consulter ma réservation ❌
  - Annuler ma réservation ❌
- Gestion d'un CatSitter
  - Donner mes disponibilités de gardes ☑
  - Consulter mon planning de réservation ❌
- Gestion d'un compte
  - Créer mon compte ☑
  - Se connecter ❌
  - Se déconnecter ❌
  - Modifier mon profil ❌

Analysé oui : ☑ / Analysé non : ❌

2.1 Réservation d'un particulier - Réserver une garde de mon chat auprès d'un particulier ou d'un professionnel

2.1.1 Liste des objets candidats

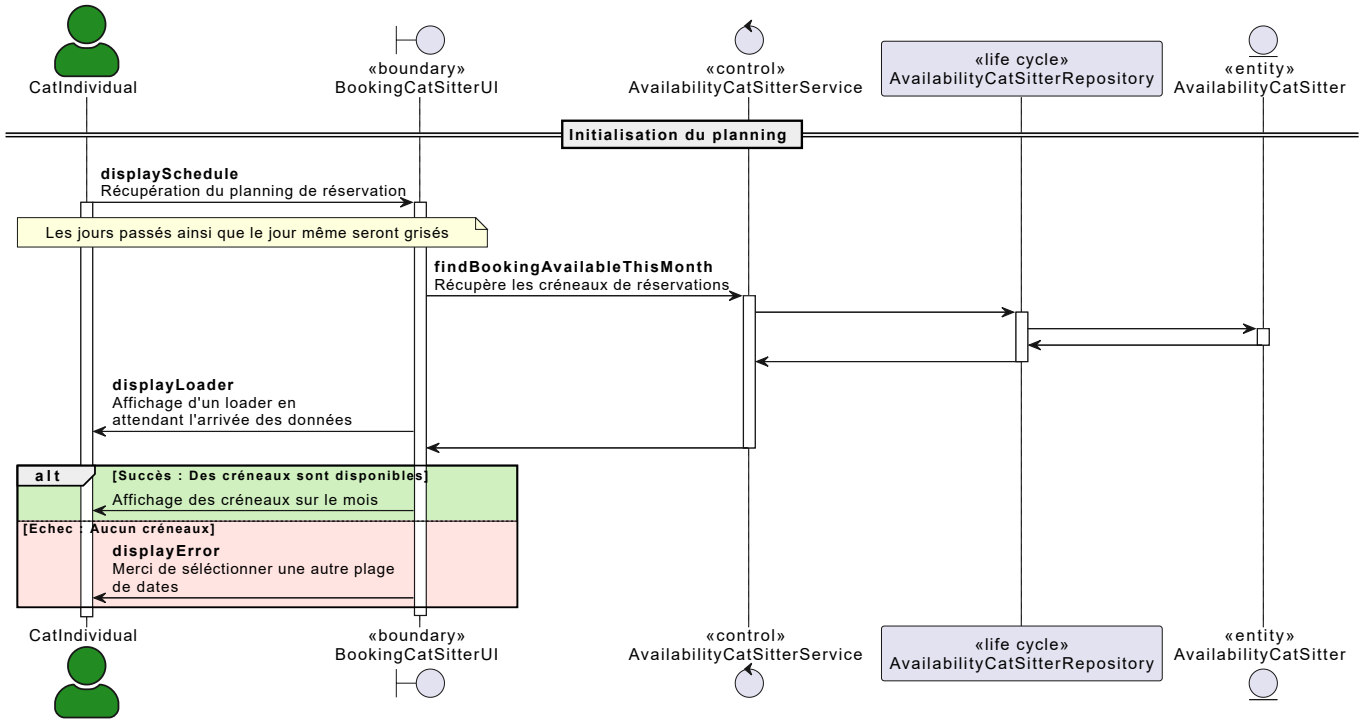
- `<<boundary>>` BookingCatSitterUI
- `<<control>>` BookingCatSitterService
- `<<control>>` AvailabilityCatSitterService
- `<<entity>>` BookingCatSitter
- `<<entity>>` AvailabilityCatSitter
- `<<life cycle>>` BookingCatSitterRepository
- `<<life cycle>>` AvailabilityCatSitterRepository



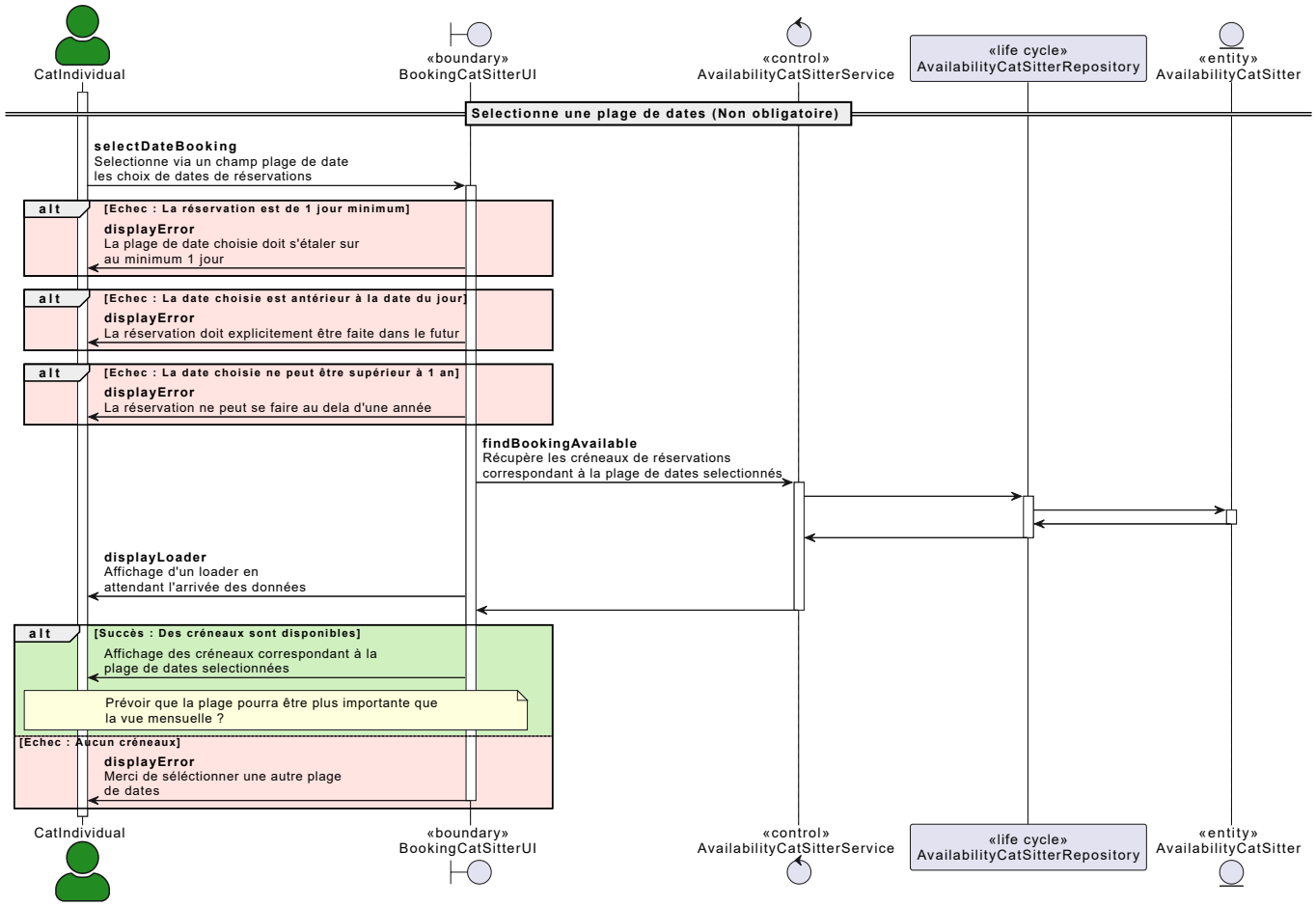
2.1.2 Description des interactions entre objets

Pour gagner en lisibilité, les diagrammes ont été séparés en 4 étapes. Les trois premières sont ordonnancées par ordre chronologique. Pour la dernière étape, il s'agit d'une sous-partie de l'étape 3.

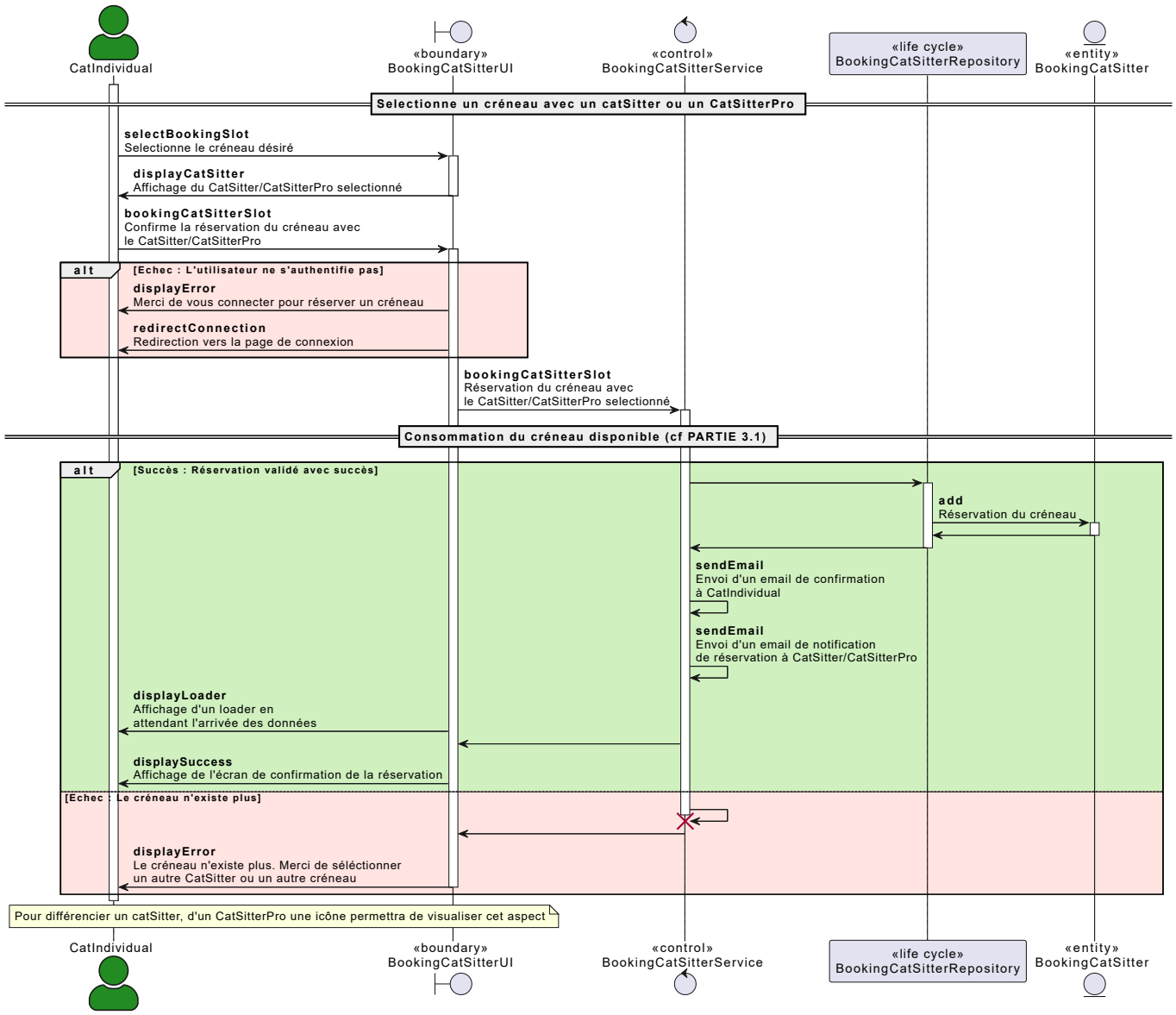
Réservation d'un particulier - Réserver une garde de mon chat auprès d'un particulier ou d'un professionnel
Diagramme de séquence - PARTIE 1



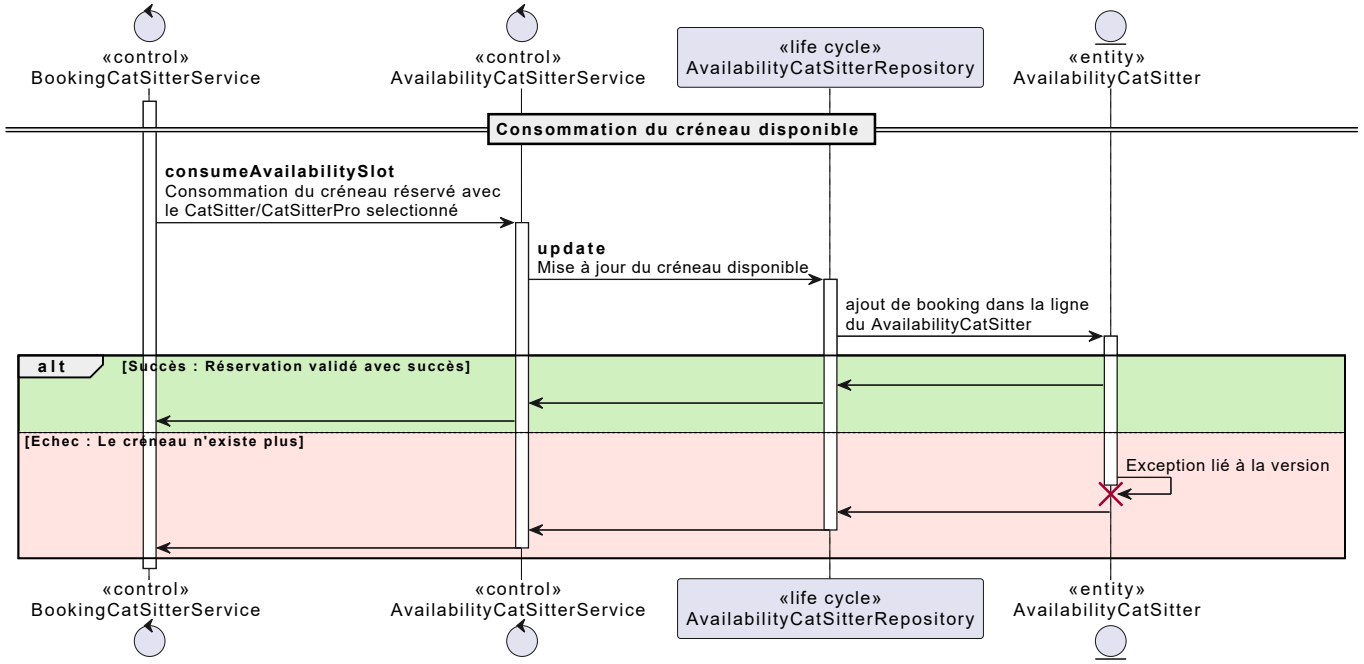
Réservation d'un particulier - Réserver une garde de mon chat auprès d'un particulier ou d'un professionnel
Diagramme de séquence - PARTIE 2



Réservation d'un particulier - Réserver une garde de mon chat auprès d'un particulier ou d'un professionnel
Diagramme de séquence - PARTIE 3



Réservation d'un particulier - Réserver une garde de mon chat auprès d'un particulier ou d'un professionnel
Diagramme de séquence - PARTIE 3.1



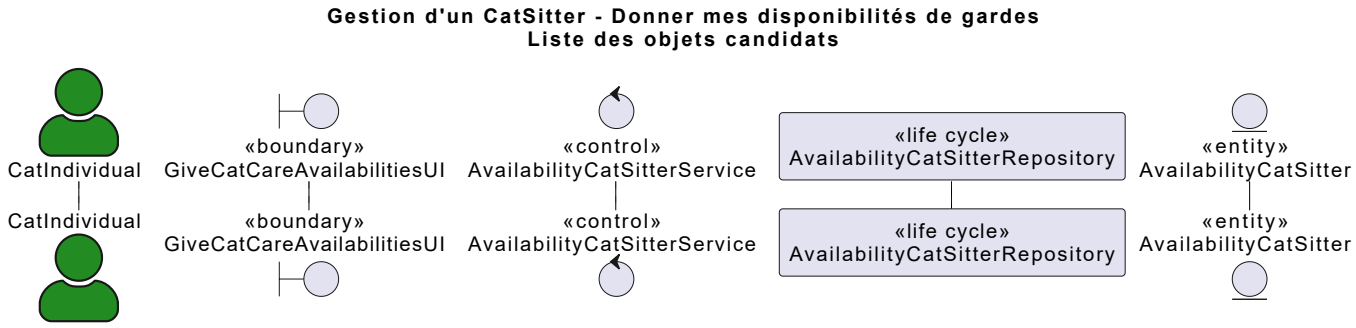
2.1.3 Description des classes

```
PlantUML 1.2023.11beta2
[From string (line 55) ]

@startuml
title Réservation d'un particulier - Réserver une garde de mon chat auprès d'un particulier ou d'un professionnel\nDiagr ...

class "<<boundary>>\nBookingCatSitterUI" as BookingCatSitterUI
class "<<control>>\nBookingCatSitterService" as BookingCatSitterService
...
... ( skipping 30 lines )
...
BookingCatSitter : User catIndividual
BookingCatSitter : Date booking
BookingCatSitter : [...]

AvailabilityCatSitter : User catSitter
AvailabilityCatSitter : Date start
AvailabilityCatSitter : Date end
AvailabilityCatSitter : Integer maxCat
AvailabilityCatSitter : [...]

User : [...]

BookingCatSitterUI ..> BookingCatSitterService
BookingCatSitterUI ..> AvailabilityCatSitterService
BookingCatSitterService ..> BookingCatSitterRepository
BookingCatSitterService .> AvailabilityCatSitterService
AvailabilityCatSitterService..> AvailabilityCatSitterRepository
BookingCatSitterRepository ..> BookingCatSitter
AvailabilityCatSitterRepository..> AvailabilityCatSitter
BookingCatSitter _- User : catSitter
 Syntax Error?
```

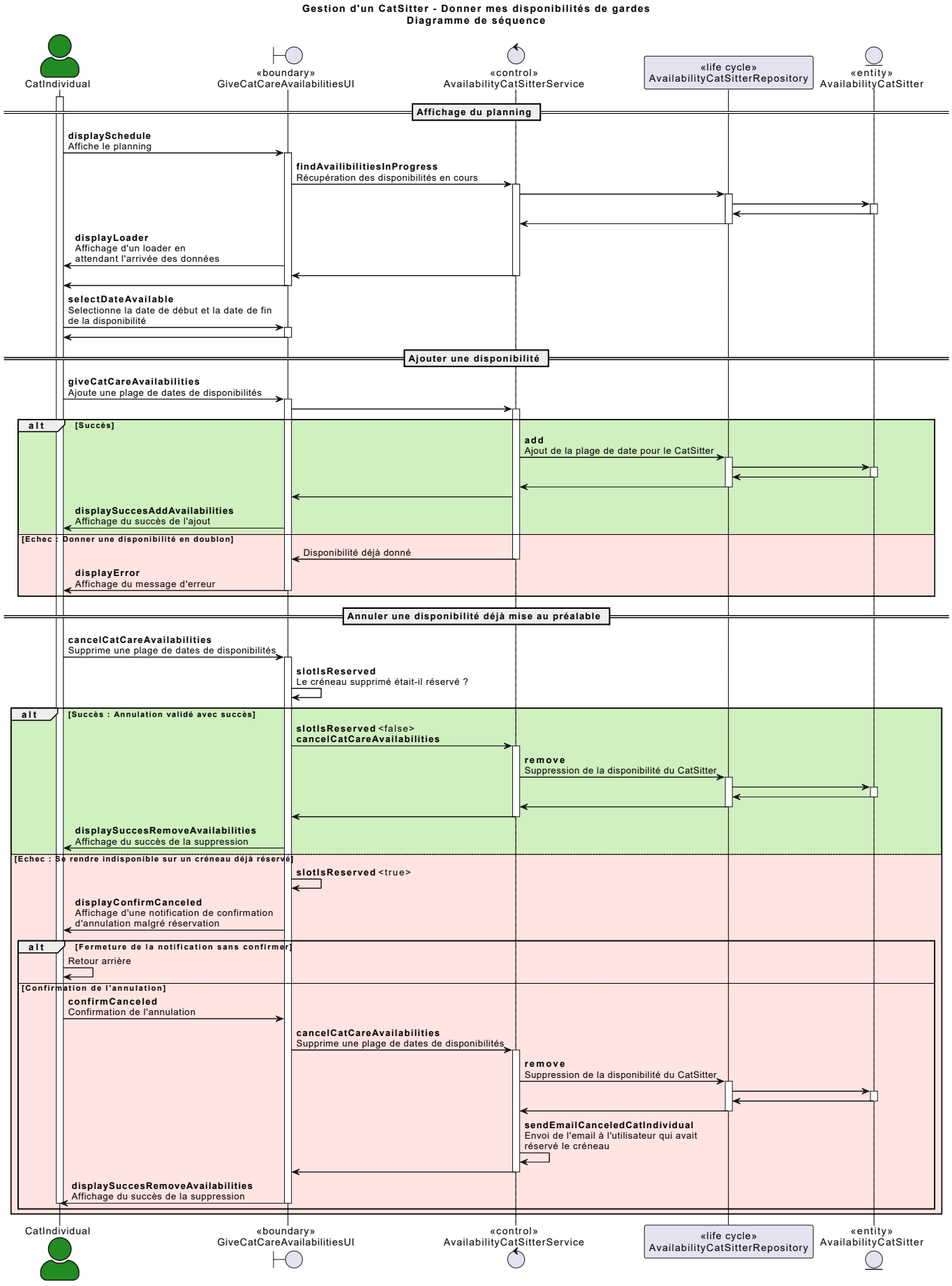
2.2 Gestion d'un CatSitter - Donner mes disponibilités de gardes

2.2.1 Liste des objets candidats

- `<<boundary>>` GiveCatCareAvailabilitiesUI
- `<<control>>` AvailabilityCatSitterService
- `<<entity>>` AvailabilityCatSitter
- `<<life cycle>>` AvailabilityCatSitterRepository

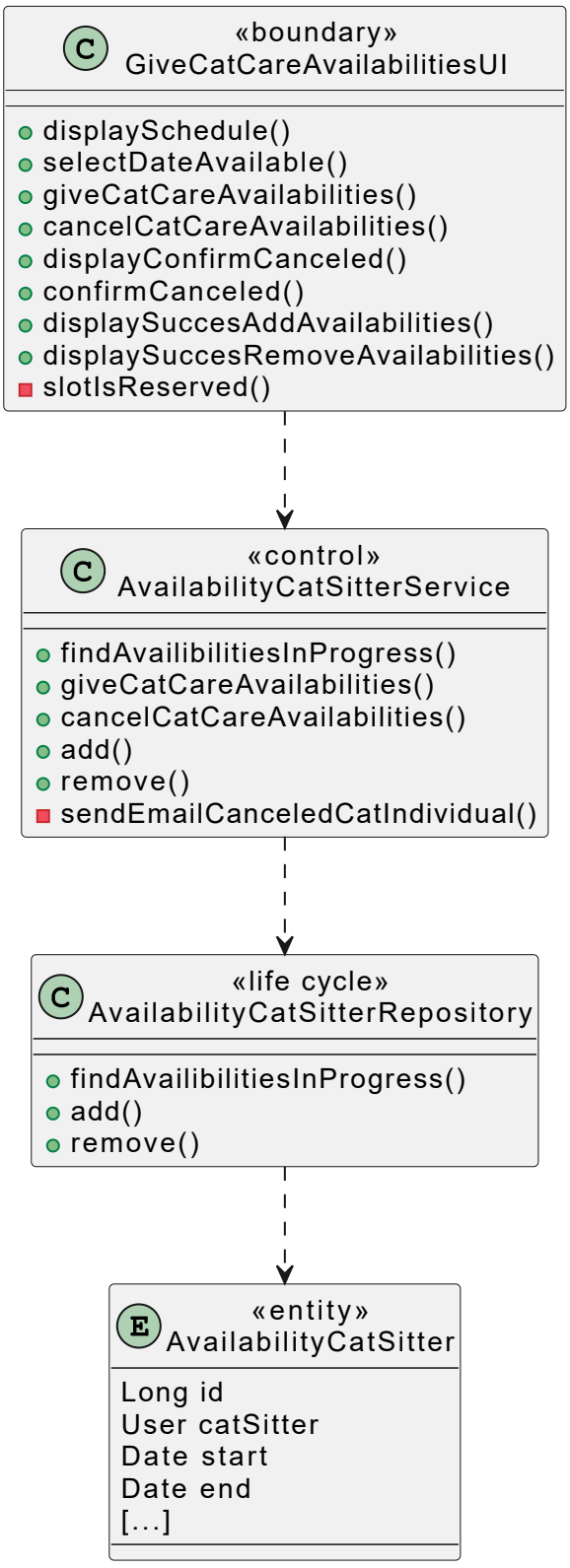


2.2.2 Description des interactions entre objets


Gestion d'un CatSitter - Donner mes disponibilités de gardes
Diagramme de séquence
CatIndividual
«boundary»
GiveCatCareAvailabilitiesUI
«control»
AvailabilityCatSitterService
«life cycle»
AvailabilityCatSitterRepository
«entity»
AvailabilityCatSitter
Affichage du planning
displaySchedule
Affiche le planning
findAvailibilitiesInProgress
Récupération des disponibilités en cours
displayLoader
Affichage d'un loader en attendant l'arrivée des données
selectDateAvailable
Selectionne la date de début et la date de fin de la disponibilité
Ajouter une disponibilité
giveCatCareAvailabilities
Ajoute une plage de dates de disponibilités
alt
[Succès]
add
Ajout de la plage de date pour le CatSitter
displaySuccesAddAvailabilities
Affichage du succès de l'ajout
[Echec : Donner une disponibilité en doublon]
Disponibilité déjà donné
displayError
Affichage du message d'erreur
Annuler une disponibilité déjà mise au préalable
cancelCatCareAvailabilities
Supprime une plage de dates de disponibilités
slotIsReserved
Le créneau supprimé était-il réservé ?
alt
[Succès : Annulation validé avec succès]
slotIsReserved <false>
cancelCatCareAvailabilities
remove
Suppression de la disponibilité du CatSitter
displaySuccesRemoveAvailabilities
Affichage du succès de la suppression
[Echec : Se rendre indisponible sur un créneau déjà réservé]
slotIsReserved <true>
displayConfirmCanceled
Affichage d'une notification de confirmation d'annulation malgré réservation
alt
[Fermeture de la notification sans confirmer]
Retour arrière
[Confirmation de l'annulation]
confirmCanceled
Confirmation de l'annulation
cancelCatCareAvailabilities
Supprime une plage de dates de disponibilités
remove
Suppression de la disponibilité du CatSitter
sendEmailCanceledCatIndividual
Envoi de l'email à l'utilisateur qui avait réservé le créneau
displaySuccesRemoveAvailabilities
Affichage du succès de la suppression

2.2.3 Description des classes

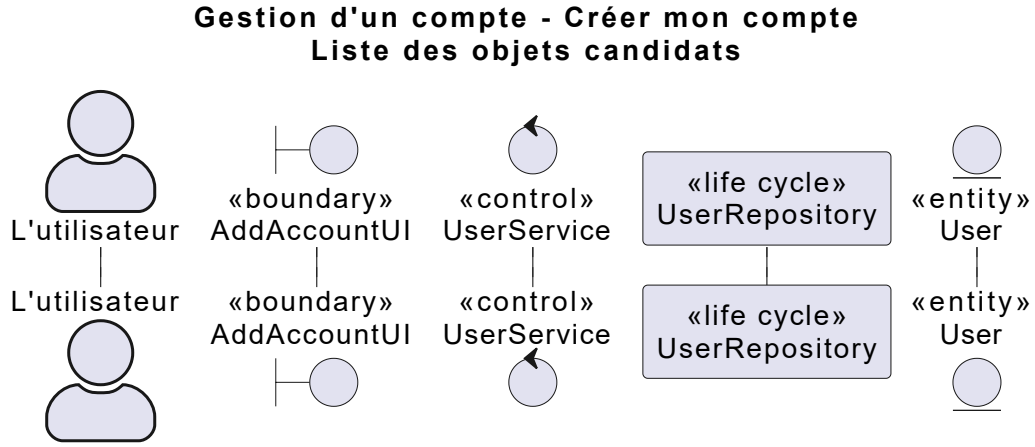
Gestion d'un CatSitter - Donner mes disponibilités de gardes
Diagramme de classes



2.3 Gestion d'un compte - Créer mon compte

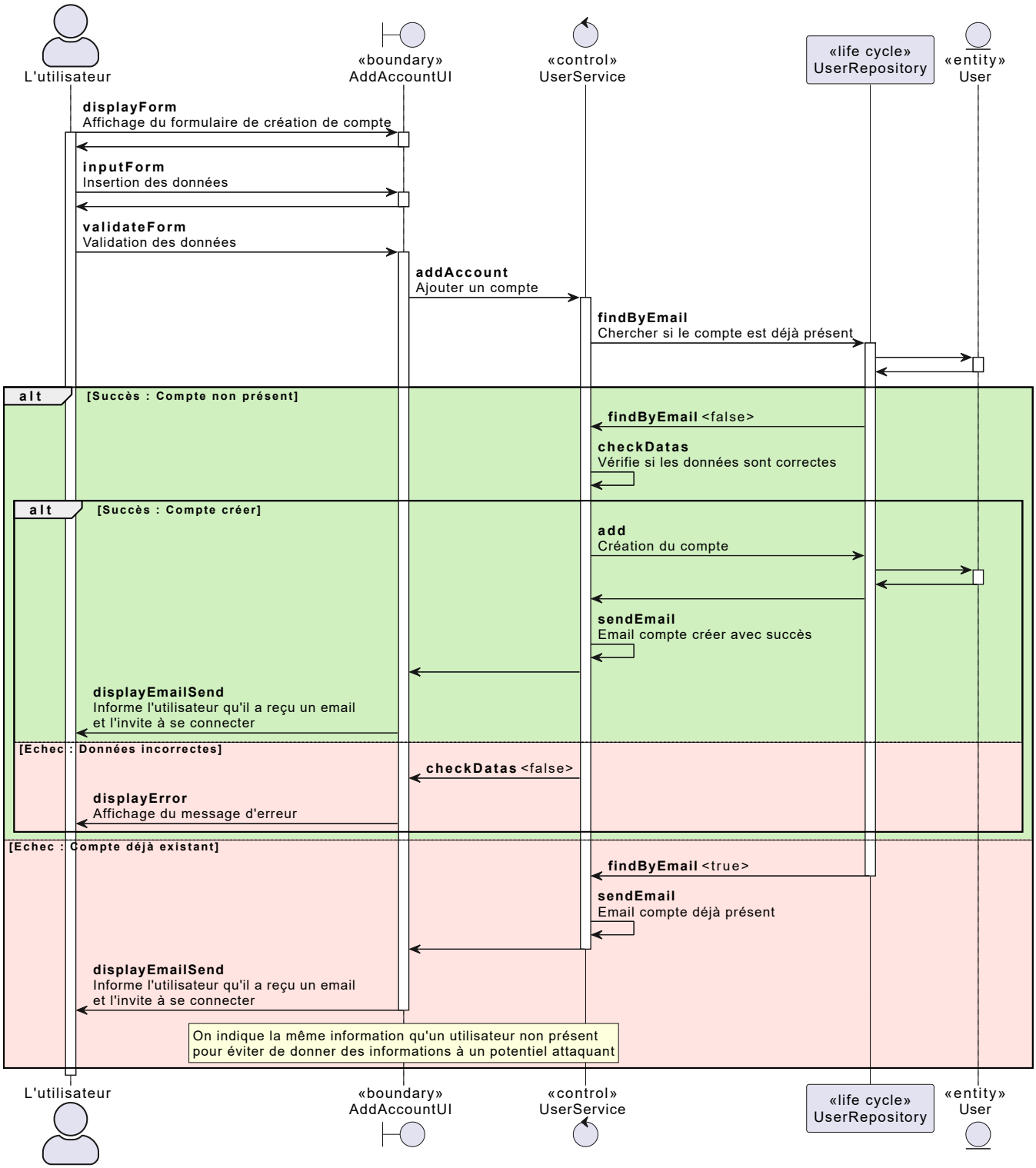
2.3.1 Liste des objets candidats

- `<<boundary>>` AddAccountUI
- `<<control>>` UseService
- `<<entity>>` User
- `<<life cycle>>` UserRepository



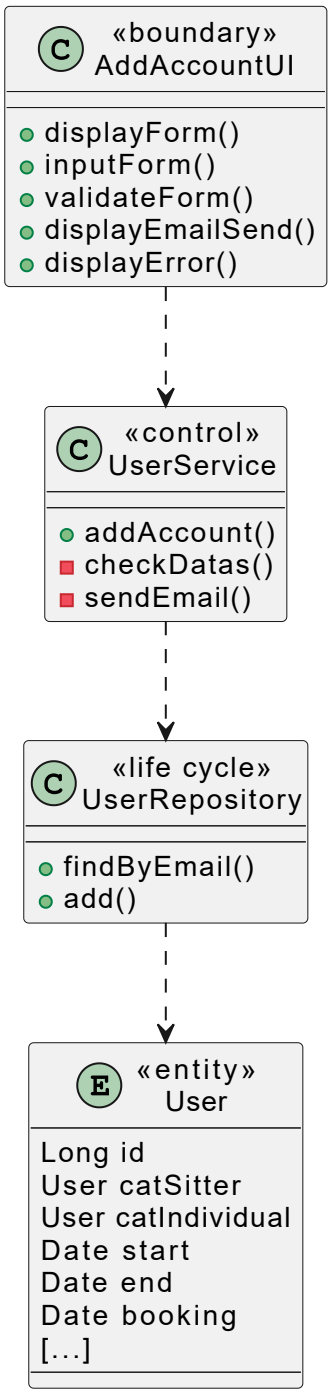
2.3.2 Description des interactions entre objets

Gestion d'un compte - Créer mon compte
Diagramme de séquence



2.3.3 Description des classes

Gestion d'un compte - Créer mon compte
Diagramme de classes



3. Regroupement des classes

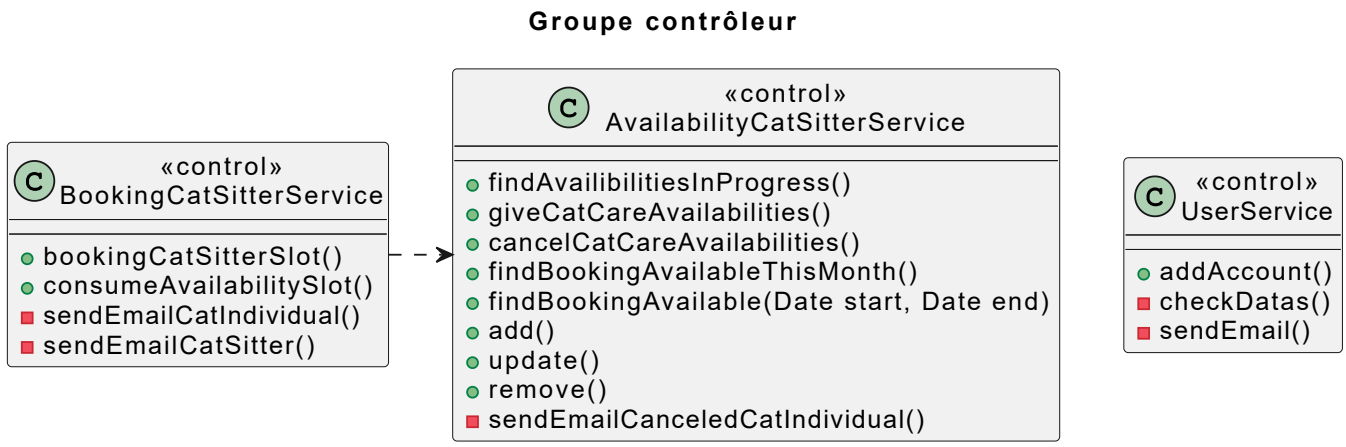
3.1 Groupe domaine

```
PlantUML 1.2023.11beta2
[From string (line 29) ]

@startuml
title Groupe domaine

entity "<<entity>>\nBookingCatSitter" as BookingCatSitter
entity "<<entity>>\nUser" as User
entity "<<entity>>\nAvailabilityCatSitter" as AvailabilityCatSitter

BookingCatSitter : Long id
BookingCatSitter : User catSitter
BookingCatSitter : User catIndividual
BookingCatSitter : Date start
BookingCatSitter : Date end
BookingCatSitter : Date booking
BookingCatSitter : [...]

AvailabilityCatSitter : Long id
AvailabilityCatSitter : User catSitter
AvailabilityCatSitter : Date start
AvailabilityCatSitter : Date end
AvailabilityCatSitter : Integer maxCat
AvailabilityCatSitter : [...]

User : Long id
User : String name
User : String lastname
User : Date birthdate
User : [...]

BookingCatSitter _- User : catSitter
 Syntax Error?
```

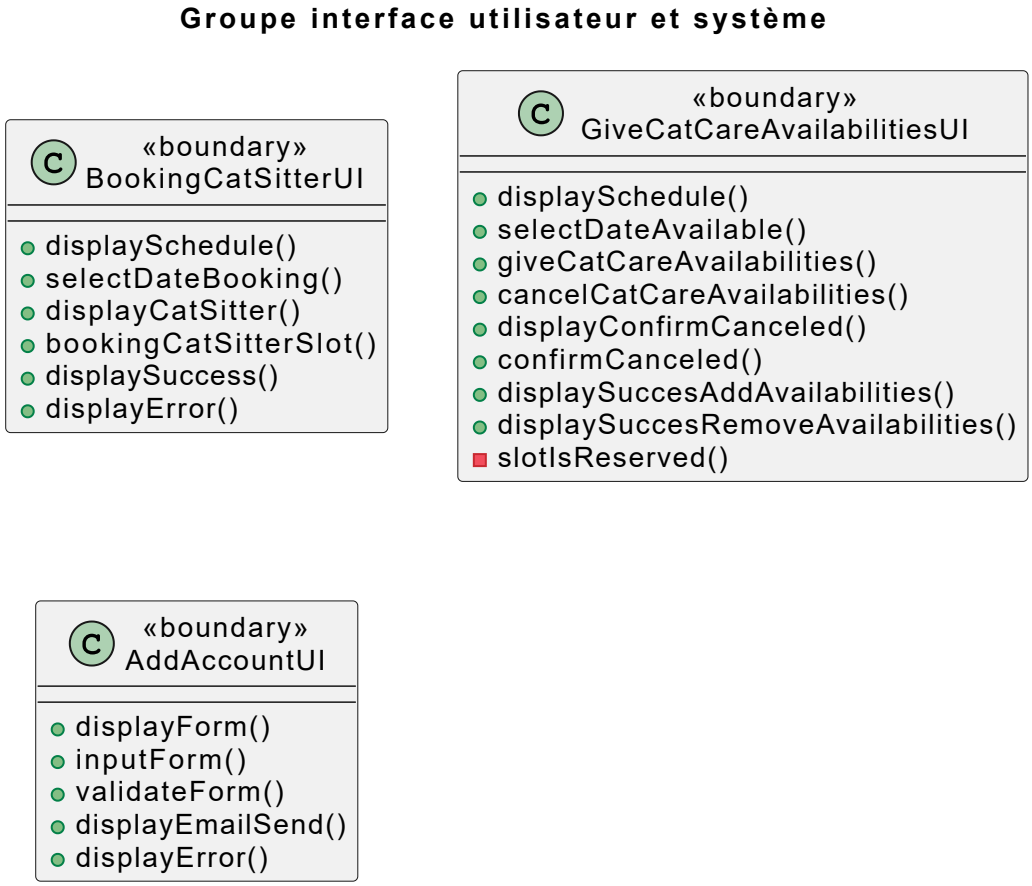
3.2 Groupe domaine et cycle de vie

```
PlantUML 1.2023.11beta2
[From string (line 47) ]

@startuml
title Groupe domaine et cycle de vie

class "<<life cycle>>\nUserRepository" as UserRepository
class "<<life cycle>>\nBookingCatSitterRepository" as BookingCatSitterRepository
...
... ( skipping 22 lines )
...
BookingCatSitter : Date booking
BookingCatSitter : [...]

AvailabilityCatSitter : Long id
AvailabilityCatSitter : User catSitter
AvailabilityCatSitter : Date start
AvailabilityCatSitter : Date end
AvailabilityCatSitter : Integer maxCat
AvailabilityCatSitter : [...]

User : Long id
User : String name
User : String lastname
User : Date birthdate
User : [...]

BookingCatSitterRepository ..> BookingCatSitter
AvailabilityCatSitterRepository..> AvailabilityCatSitter
UserRepository ..> User
BookingCatSitter _- User : catSitter
 Syntax Error?
```

3.3 Groupe contrôleur



3.4 Groupe interface utilisateur et système



Conception

Lors des phases précédentes, nous avons mis en place le sujet et décrit l'ensemble du besoin ainsi que ses contraintes. Nous allons désormais nous atteler à la phase de réalisation, c'est-à-dire comment nous allons implémenter la solution ?

Nous allons tout d'abord commencer par l'architecture en identifiant les contraintes techniques de l'applicatif, puis en modélisant les packages et dépendances de celle-ci. Puis nous terminerons par la manière d'effectuer le déploiement de notre solution.

Par la suite, nous nous pencherons sur les technologies à mettre en place et pour ce faire nous décrirons à chaque étape les arguments en faveur de ce choix. Nous commencerons par les choix des langages et des frameworks à utiliser. Ensuite, nous nous nous focaliserons sur le serveur qui va porter notre applicatif et le stockage des données. Après, nous nous intéresserons aux différentes couches applicatives : persistance, métier, service et présentation. Enfin nous, nous pencherons sur l'environnement de développement ainsi que les recettes à mettre en place.

Pour terminer, nous reviendrons sur les cas d'utilsations réalisés dans la section analyse pour venir enrichir les modèles existants afin de présenter ce qui sera réellement implémentés dans notre applicatif.

2. Architecture

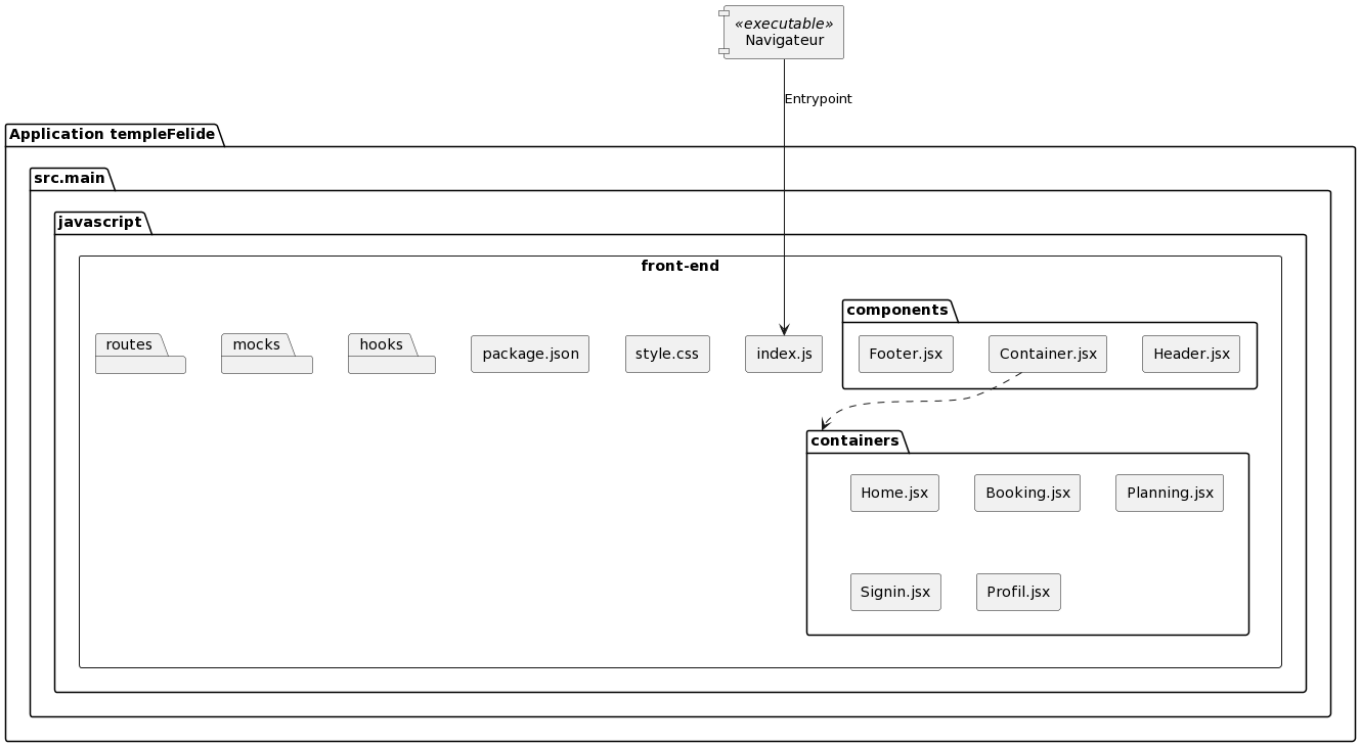
2.1 Contraintes techniques

Les contraintes primordiales lié à un site web sont la création d'un nom de domaine et son hébergement. En effet, il faut que le nom de domaine soit libre, mais également être vigilant à l'extension qui pourrait amener à un cybersquattage.

Ensuite nous avons des obligations liés à une navigation web, c'est-à-dire que le site doit être fluide et rapide. Notre application devra être user-friendly et accessible pour toucher un maximum d'utilisateurs de tout âge et dans des situations différentes. En effet, il ne s'agit pas d'un intranet qui aurait eu des contraintes différentes. Par ailleurs, de par sa visibilité publique une sécurisation minimales doit être mise en place.

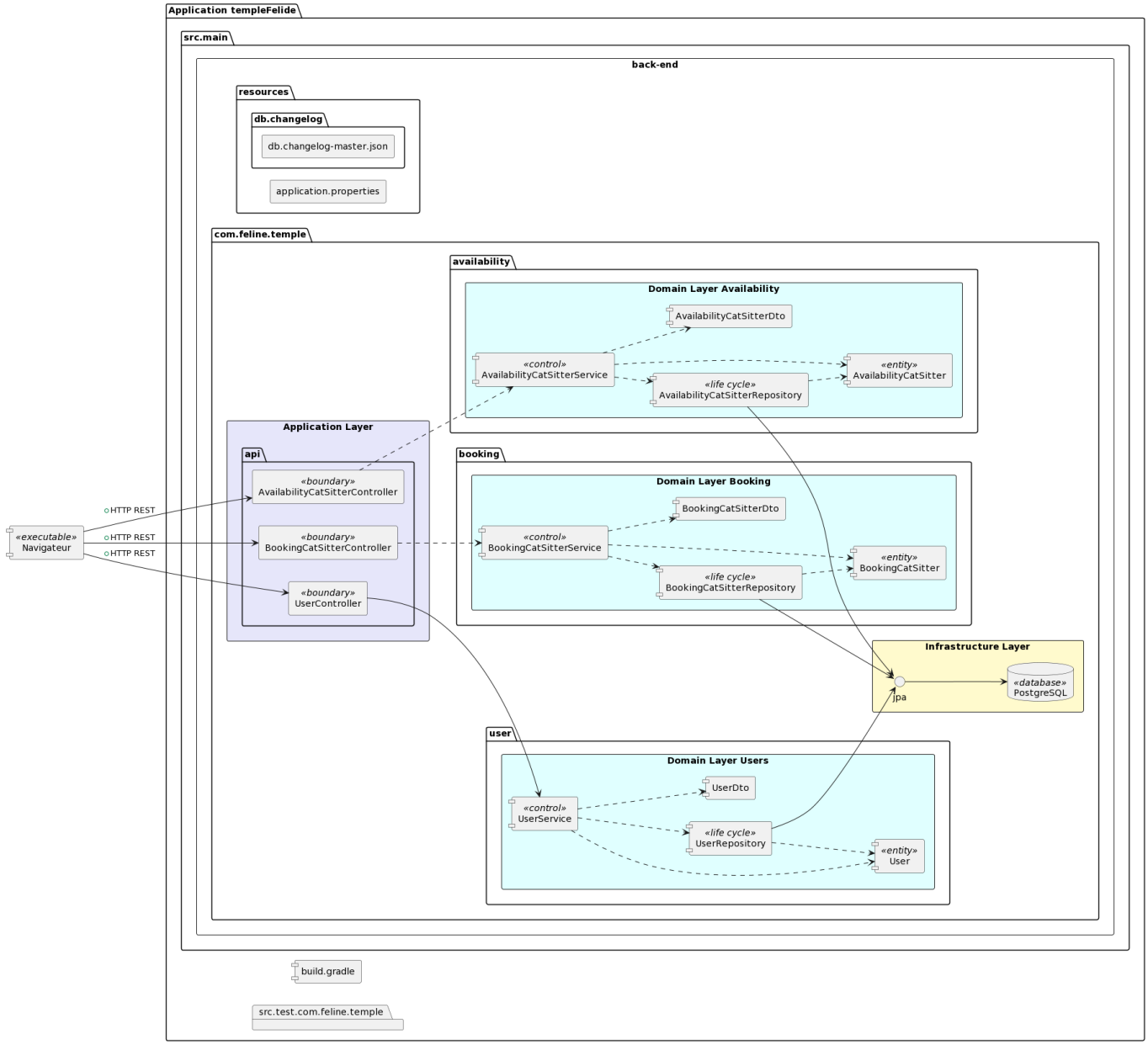
Enfin il y a les contraintes inhérentes à une bonne conception et implémentation ainsi qu'à l'utilisation de technologies répandues, fiables et éprouvées afin de maintenir l'application. Pour ce faire, nous allons mettre en place une architecture héxagonale.

2.2 Packages et dépendances



- 🗎 package.json : Contient les dépendances front
- 🗎 index.js : Point d'entrée (SPA)
- 🗎 style.css : Feuille de style
- 🗀 components : Contient les composants
- 🗀 container : Contient la logique métier
- 🗀 hooks : Contient les appels au back-end
- 🗀 routes : Contient les chemins (URL) permettant la navigation
- 🗀 mocks : Contient les bouchons

⚠ Il s'agit d'une liste non exhaustive et qui pourra être amenée à changer en fonction de l'implémentation.

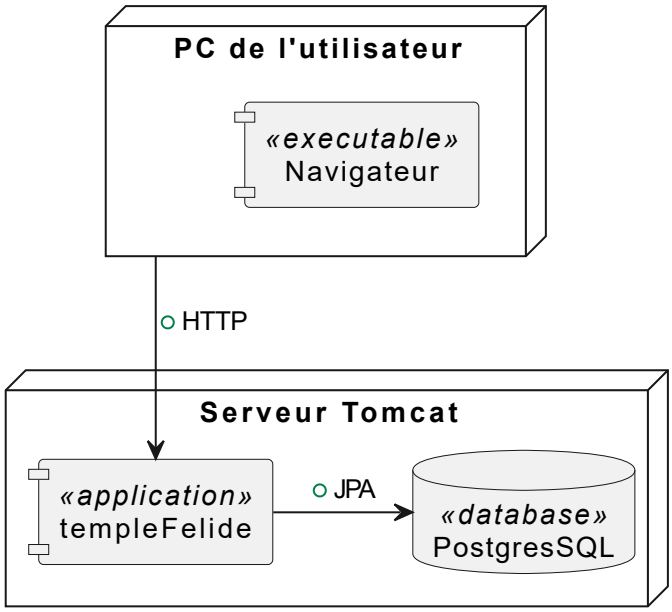


- 🗎 application.properties : Contient les propriétés fonctionnant avec spring boot
- 🗎 db.changelog-master.json : Contient la migration et le versionnage de la base de donnée
- 🗎 build.gradle : Contient les dépendances back
- 🗀 availability / booking / user (Domain Layer) : Contient la logique métier
- 🗀 api (Application Layer) : Permet la communication avec le front-end
- 🗀 routes : Contient les chemins (URL) permettant la navigation
- 🗀 test.com.feline.temple : Contient les tests unitaires et d'intégrations

⚠ Il s'agit d'une liste non exhaustive et qui pourra être amenée à changer en fonction de l'implémentation.

2.3 Déploiement

Le déploiement se fera dans un premier temps sur un pc en local de la manière suivante :



2.3.1 Le nom de domaine et son hébergement

Si les délais le permettent, l'application pourra être déployé dans le cloud avec kubernetes ou un autre provider qu'il faudra sélectionner. Il sera également possible d'acheter le nom de domaine suivant : www.templefelide.fr.

3. Choix technologiques

D'un point de vue global, le choix se portera généralement sur des langages open source de part la gratuité de ceux-ci. En effet, on pourrait choisir des langages libres, cependant cela pourrait amener des coûts supplémentaires. La différence entre open-source et libre ne sera pas expliqué ici. J'invite le lecteur à voir la bibliographie. De plus, il faut choisir des langages et bibliothèques robustes, maintenues et portés par une communauté afin d'avoir une maintenance acceptable. Enfin, il faut prendre en considération la rapidité d'apprentissage du langage ou de la bibliothèque.

Nous allons partir sur une architecture et un choix de langage fréquent dans les applications web de nos jours.

3.1 Front-End

Concernant la partie visuel de notre applicatif, nous faisons le choix de ReactJS et Ant Design.

React JS est un langage performant de part sa virtualisation du DOM en mémoire et sa manière de calculer intelliger pour gérer les mises à jours et changements du DOM. Toute cette gestion du DOM se fait discrétement ce qui permet un rendu rapide, intelligent voir invisible pour l'utilisateur. De plus il s'agit d'un langage mature, qui existe depuis maintenant 10 ans et qui est maintenu avec des versions récurrentes (même si la dernière version commence à dater de juin 2022). Par ailleurs c'est un langage simple à comprendre, flexible qui permet une bonne réutilisation des composants. Enfin, il est rattaché à un ensemble vaste d'outils qui permettent d'inspecter et maintenir le code.

L'utilisation de la bibliothéque Ant Design va permettre d'avoir un ensemble de composant de haute qualité, afin de faire rapidement un site esthétique et bien construit.

3.2 Back-End

Nous allons désormais choisir le langage ainsi que le framework lié au back-end de notre applicatif. Nous avons décidés de mettre en place Kotlin avec les bibliothèques de spring, qui est un des acteurs les plus reconnus dans ce domaine.

Le choix de Kotlin est porté par les éléments suivants : c'est un langage multi-paradigme, c'est-à-dire à capacité fonctionnelle et orentiée objet. D'une part il à un typage statique qui fonctionne à la compilation. D'autre part, il est concis, explicite et permet donc une meilleure lisibilité, on gagne donc en qualité plutôt qu'en quantité de code. De plus, c'est un langage qui évite la nullité comme dans Java qui entraîne souvent des erreurs de type NullPointerException lors de l'éxécution. Enfin c'est un langage facile à apprendre et toujours interopérable avec Java si besoin.

Concernant le framework spring, nous allons passer par spring initializr pour démarrer notre projet. Nous allons ajouter comme dépendance spring boot qui permet de créer rapidement une application ainsi que spring MVC qui permet de gérer comme son acronyme l'indique le modèle, la vue et le contrôleur. Puis nous ajouterons également spring-security pour avoir une sécurité minimale ainsi qu'une gestion de l'authentification et autorisation. Enfin nous utiliserons spring Data JPA pour la persistance des données.

3.3 Gestionnaire de dépendances

Les gestionnaires de dépendances les plus couramment utilisés sur le marché sont maven et gradle. Ils ont chacuns leurs avantages et inconvénients, cependant le choix va se porter sur gradle qui est plus flexible et rapide que maven.

3.4 Serveur web

Le choix du serveur web se porte tout naturellement sur Tomcat. En effet, il est embarqué et recommandé avec spring boot et spring MVC.

3.5 Stockage des données

Au vu de l'application et de son modèle de donnée, il paraît assez évident qu'il faut utiliser un système de gestion de base de donnée relationnelle (SGBDR) afin de mettre des relations rigoureuses entre nos objets. Le choix du SGBD va se porter sur PostgreSQL, qui permet malgré tout de gérer du NOSQL s'il y avait un besoin futur. Cette décision répond aux critères principaux d'open source et d'avoir une communauté importante. Cependant, il faudra être vigilant sur la manière d'indexer les données et les slow query, car PostgreSQL n'est pas le système le plus rapide.

Par ailleurs, nous allons utiliser liquibase qui est un migrateur de base de donnée open-source. Il va nous permettre de versionner notre base de donnée.

3.6 Couche de persistance et métier

Au vu des technologies selectionnés sur le back-end et la base de données : Kotlin, Spring et PostgresSQL ; il paraît naturel de porter notre choix sur l'api JPA pour la persistence des données ainsi qu'Hibernate pour l'implémentation de celle-ci. Pour les entités, cela sera fait au travers de l'annotation @Entity comme défini par JPA.

3.7 Couche service et composant

Les couches services et composant utiliseront les annotations @Service et @Component proposé par le framework spring.

3.8 Couche présentation

Au niveau de la communication entre le front-end et le back-end nous appliquerons l'architecture REST. La couche présentation se fera au travers de l'API Fetch intégré par défaut à ReactJS pour le front-end. Concernant la partie back-end nous utiliserons les annotations @Controller et @RequestMapping.

3.9 Environnement de développement

Aucune contrainte ne se fera sur ce sujet. Les développeurs doivent être libre d'utiliser l'environnement qu'ils veulent.

3.10 Recette de l'application

Il est important de se soucier de tester l'application pour éviter de potentiels erreurs. La recette pourrait se faire au travers d'une intégration continue, mais cela nécessite beaucoup de moyens et un coût non négligeable. Nous passerons donc par des tests unitaires et intégrations lancés manuellement par les développeurs, puis par des tests manuels.

3.10.1 Tests unitaires et intégrations

Les tests unitaires et intégrations seront réalisés avec Junit 5, MockK et Wiremock pour le back-end. Quant au front-end on utilisera Jest et MSW.

3.10.2 Test end to end

Si les délais et les coûts nous avaient permis d'aller plus loin, nous aurions pu utiliser cypress, mais afin d'être raisonnable les tests de parcours complets seront réalisés manuellement.

4. Cas d'utilisation

Les cas d'utilisation analysé dans la section précédente sont repris en appliquant les critères des technologies selectionnés. Pour rappel les cas d'utilisations suivants avaient été analysé :

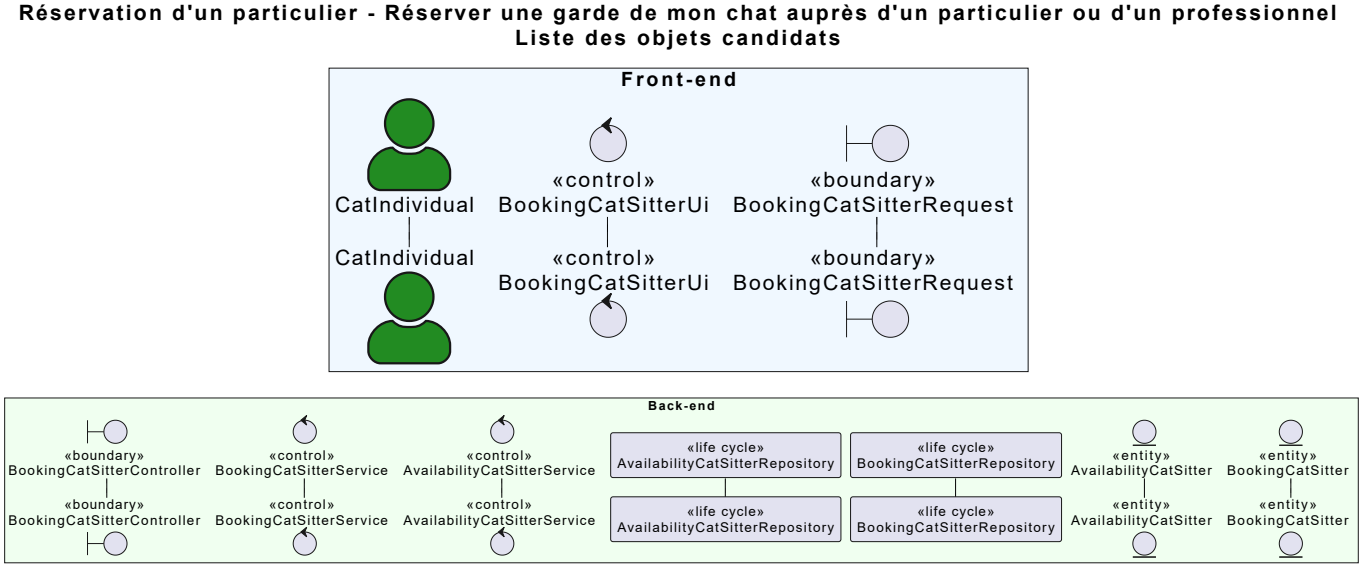
- Réservation d'un particulier
  - Réserver une garde de mon chat auprès d'un particulier ou d'un professionnel ☑
  - Consulter ma réservation ❌
  - Annuler ma réservation ❌
- Gestion d'un CatSitter
  - Donner mes disponibilités de gardes ☑
  - Consulter mon planning de réservation ❌
- Gestion d'un compte
  - Créer mon compte ☑
  - Se connecter ❌
  - Se déconnecter ❌
  - Modifier mon profil ❌

Analysé oui : ☑ / Analysé non : ❌

4.1 Réservation d'un particulier - Réserver une garde de mon chat auprès d'un particulier ou d'un professionnel

4.1.1 Liste des objets candidats

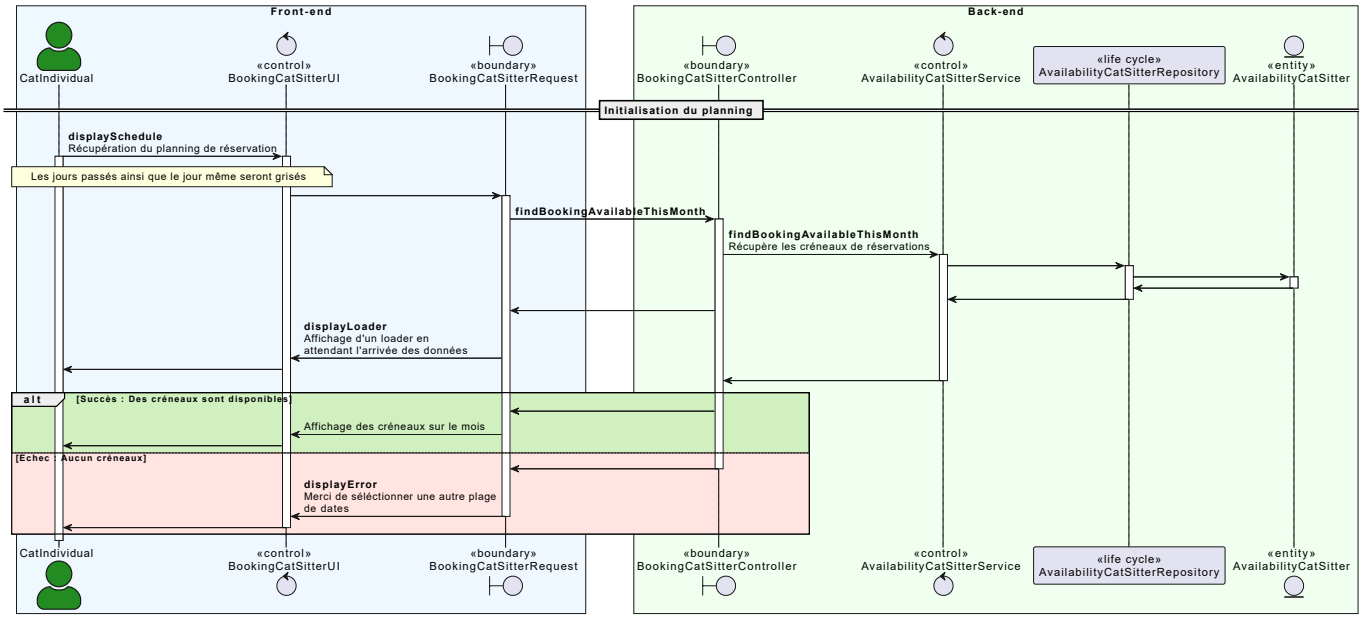
- `<<control>>` BookingCatSitterUi
- `<<boundary>>` BookingCatSitterRequest
- `<<boundary>>` BookingCatSitterController
- `<<control>>` BookingCatSitterService
- `<<control>>` AvailabilityCatSitterService
- `<<entity>>` BookingCatSitter
- `<<entity>>` AvailabilityCatSitter
- `<<life cycle>>` BookingCatSitterRepository
- `<<life cycle>>` AvailabilityCatSitterRepository



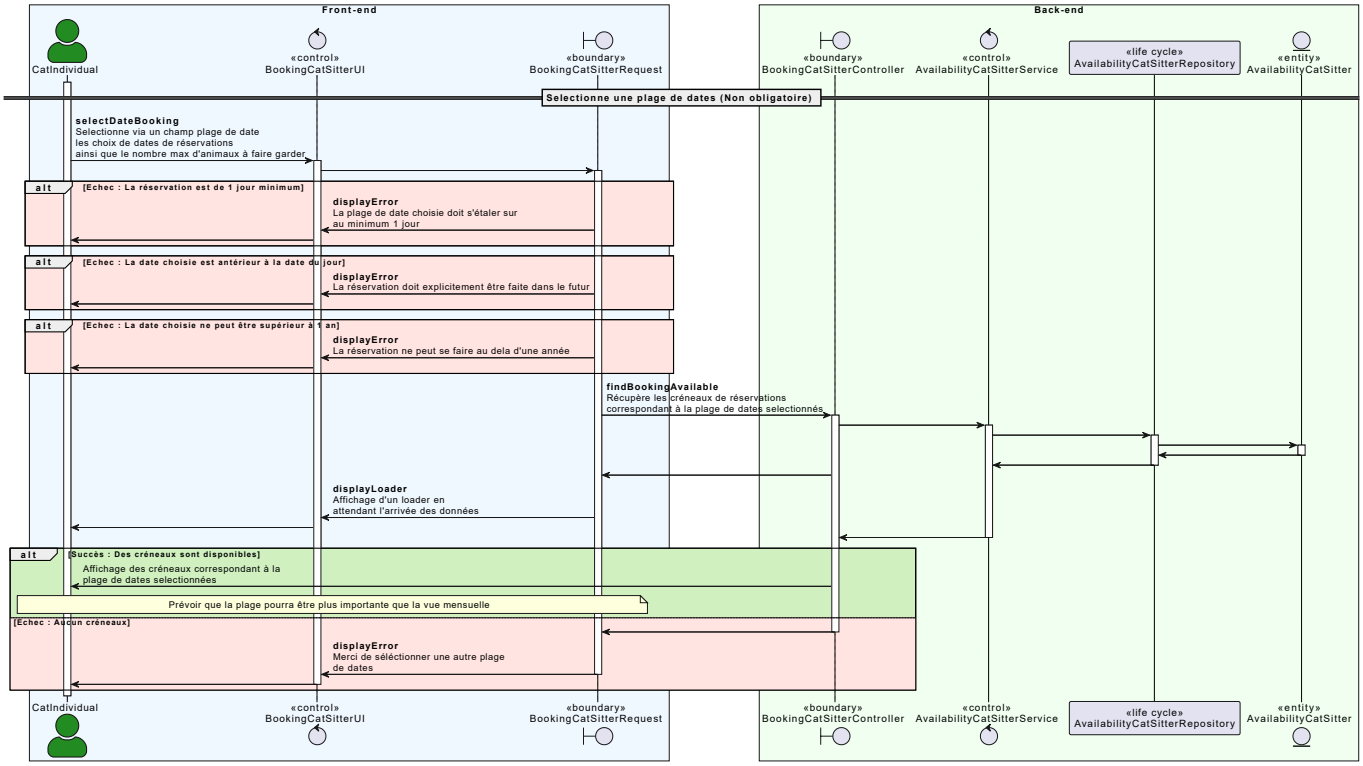
4.1.2 Description des interactions entre objets

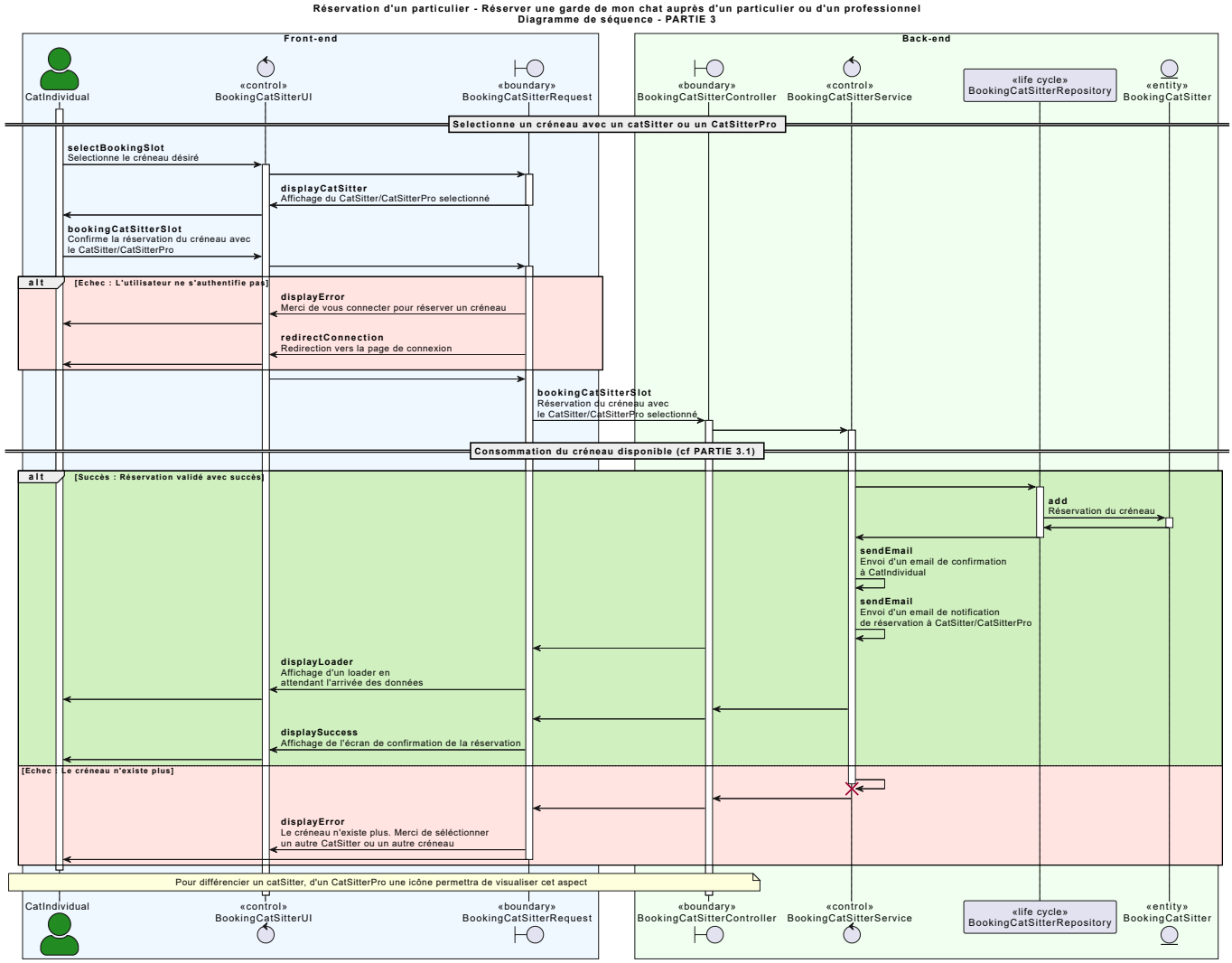
Pour gagner en lisibilité, les diagrammes ont été séparés en 4 étapes. Les trois premières sont ordonnancées par ordre chronologique. Pour la dernière étape, il s'agit d'une sous-partie de l'étape 3.

Réservation d'un particulier - Réserver une garde de mon chat auprès d'un particulier ou d'un professionnel
Diagramme de séquence - PARTIE 1

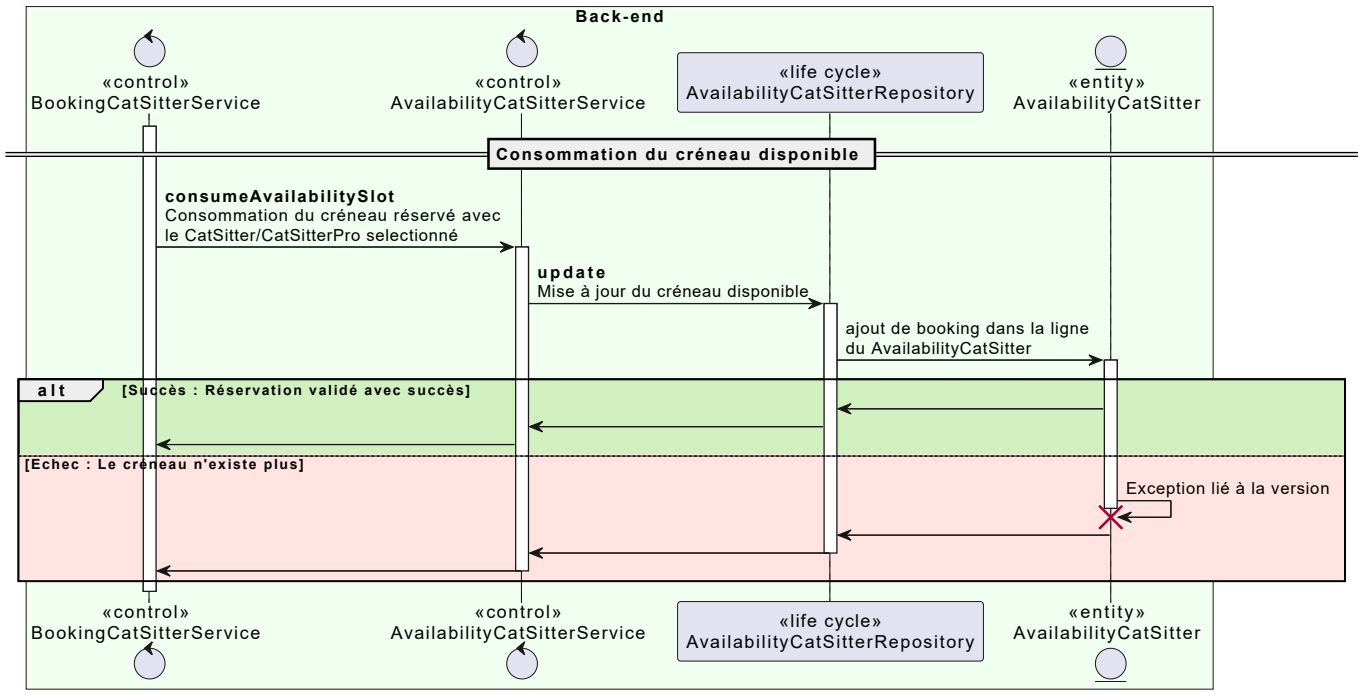


Réservation d'un particulier - Réserver une garde de mon chat auprès d'un particulier ou d'un professionnel
Diagramme de séquence - PARTIE 2





Réservation d'un particulier - Réserver une garde de mon chat auprès d'un particulier ou d'un professionnel
Diagramme de séquence - PARTIE 3.1



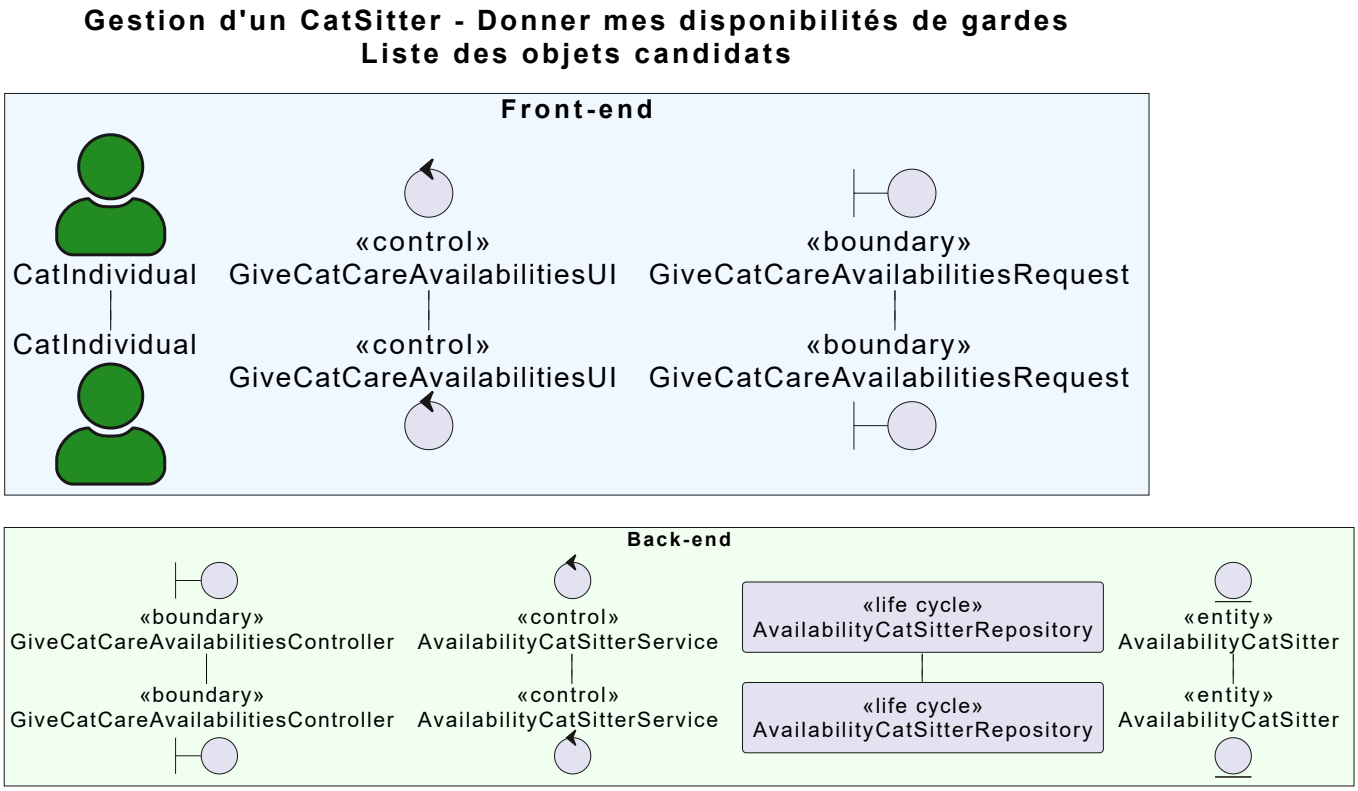
4.1.3 Description des classes

```
PlantUML 1.2023.11beta2
[From string (line 65) ]

@startuml
title Réservation d'un particulier - Réserver une garde de mon chat auprès d'un particulier ou d'un professionnel\nDiagr ...

class "<<control>>\nBookingCatSitterUI" as BookingCatSitterUI #AliceBlue
class "<<boundary>>\nBookingCatSitterRequest" as BookingCatSitterRequest #AliceBlue
...
... ( skipping 40 lines )
...
BookingCatSitter : [...]

AvailabilityCatSitter : User catSitter
AvailabilityCatSitter : Date start
AvailabilityCatSitter : Date end
AvailabilityCatSitter : Integer maxCat
AvailabilityCatSitter : [...]

User : [...]

BookingCatSitterUI ..> BookingCatSitterRequest
BookingCatSitterRequest ..> BookingCatSitterController
BookingCatSitterController ..> BookingCatSitterService
BookingCatSitterController ..> AvailabilityCatSitterService
BookingCatSitterService ..> BookingCatSitterRepository
BookingCatSitterService .> AvailabilityCatSitterService
AvailabilityCatSitterService..> AvailabilityCatSitterRepository
BookingCatSitterRepository ..> BookingCatSitter
AvailabilityCatSitterRepository..> AvailabilityCatSitter
BookingCatSitter _- User : catSitter
 Syntax Error?
```

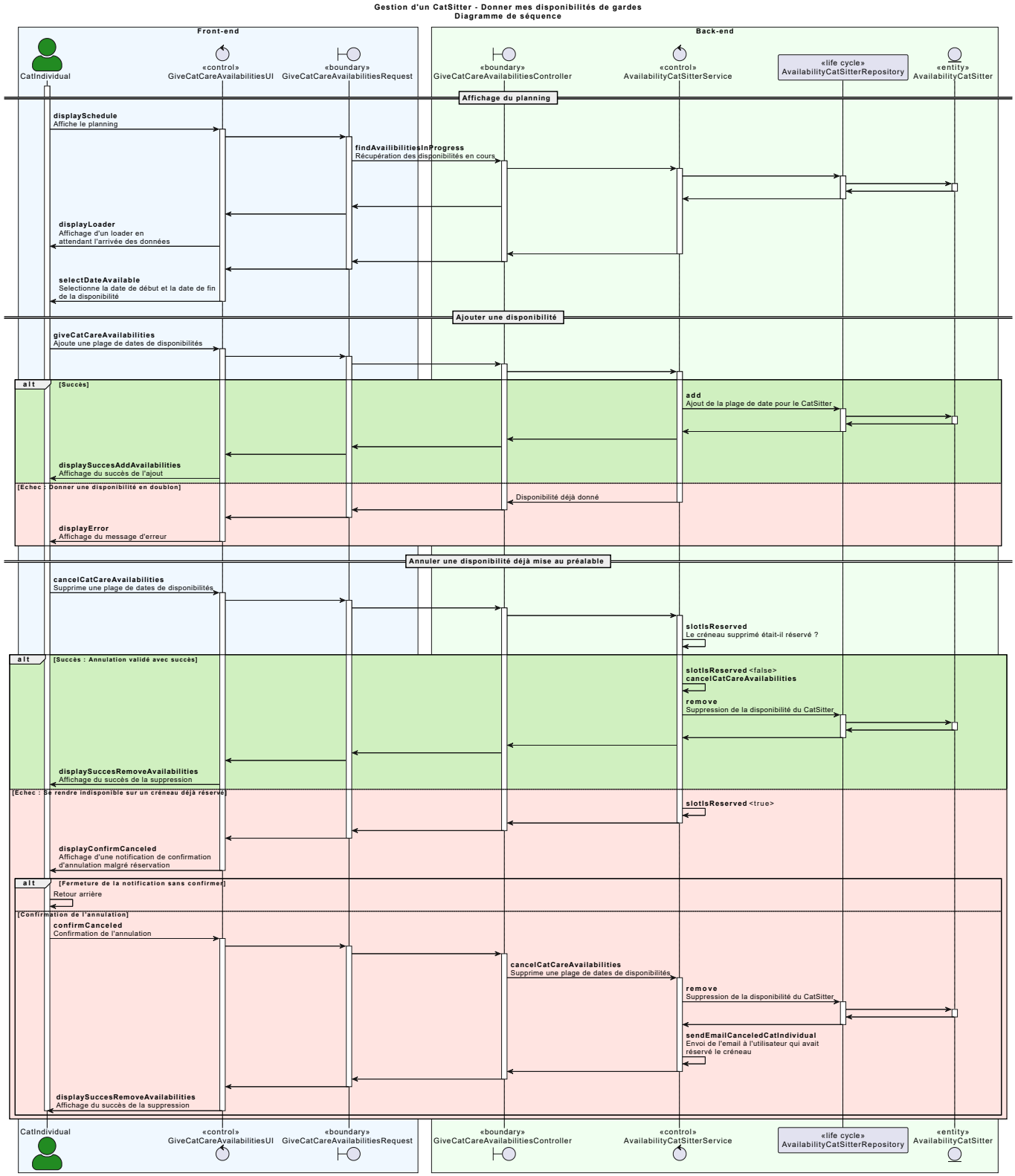
4.2 Gestion d'un CatSitter - Donner mes disponibilités de gardes

4.2.1 Liste des objets candidats

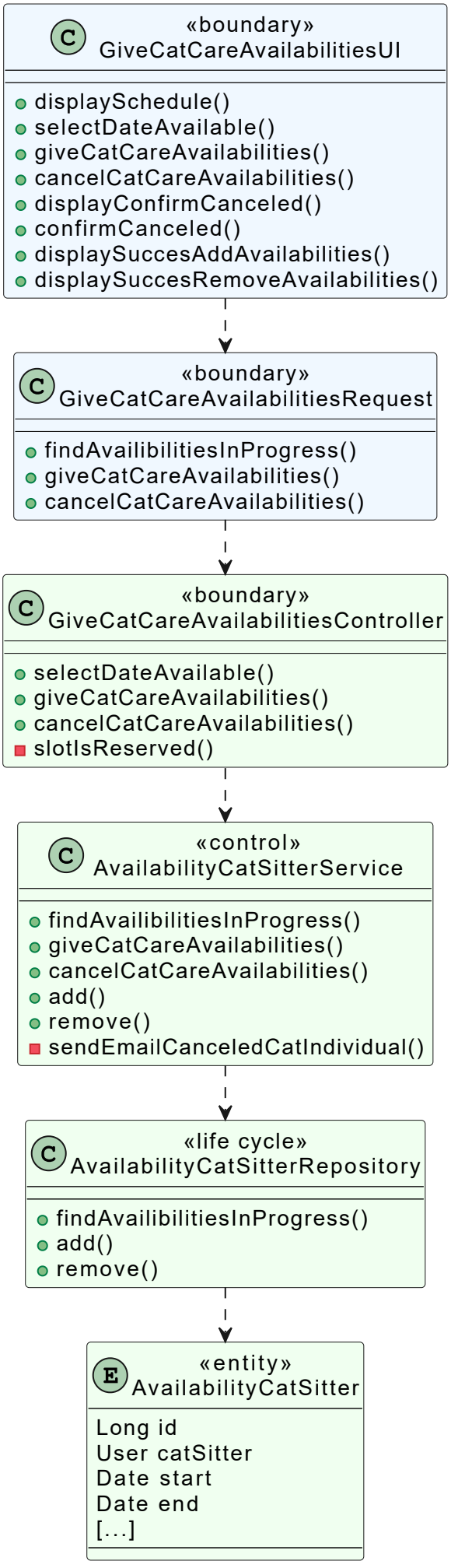
- `<<control>>` GiveCatCareAvailabilitiesUI
- `<<boundary>>` GiveCatCareAvailabilitiesRequest
- `<<boundary>>` GiveCatCareAvailabilitiesController
- `<<control>>` AvailabilityCatSitterService
- `<<entity>>` AvailabilityCatSitter
- `<<life cycle>>` AvailabilityCatSitterRepository



4.2.2 Description des interactions entre objets



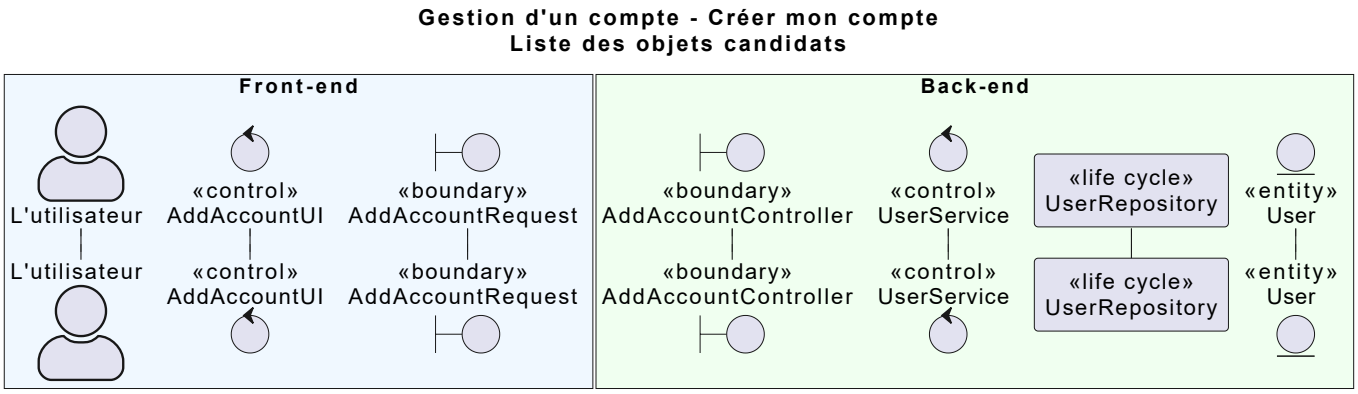
4.2.3 Description des classes


«boundary»
GiveCatCareAvailabilitiesUI
displaySchedule()
selectDateAvailable()
giveCatCareAvailabilities()
cancelCatCareAvailabilities()
displayConfirmCanceled()
confirmCanceled()
displaySuccesAddAvailabilities()
displaySuccesRemoveAvailabilities()
«boundary»
GiveCatCareAvailabilitiesRequest
findAvailibilitiesInProgress()
giveCatCareAvailabilities()
cancelCatCareAvailabilities()
«boundary»
GiveCatCareAvailabilitiesController
selectDateAvailable()
giveCatCareAvailabilities()
cancelCatCareAvailabilities()
slotIsReserved()
«control»
AvailabilityCatSitterService
findAvailibilitiesInProgress()
giveCatCareAvailabilities()
cancelCatCareAvailabilities()
add()
remove()
sendEmailCanceledCatIndividual()
«life cycle»
AvailabilityCatSitterRepository
findAvailibilitiesInProgress()
add()
remove()
«entity»
AvailabilityCatSitter
Long id
User catSitter
Date start
Date end
[...]

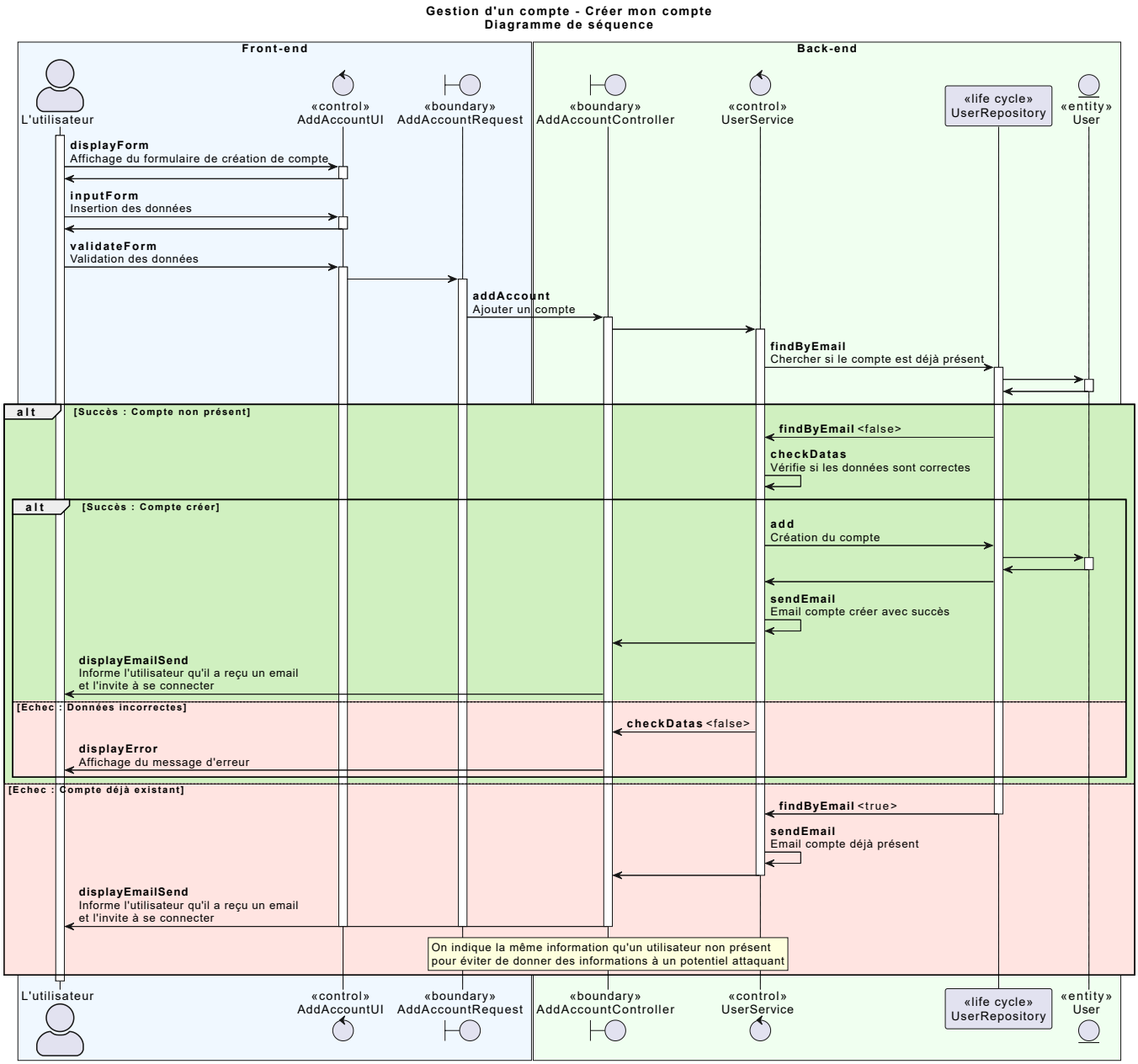
4.3 Gestion d'un compte - Créer mon compte

4.3.1 Liste des objets candidats

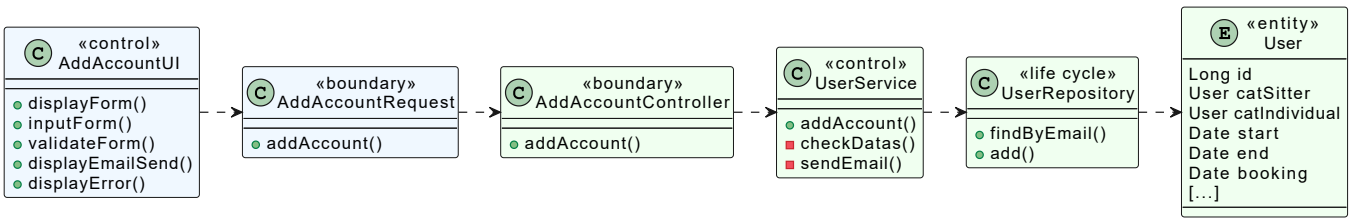
- `<<control>>` AddAccountUI
- `<<boundary>>` AddAccountRequest
- `<<boundary>>` AddAccountController
- `<<control>>` UseService
- `<<entity>>` User
- `<<life cycle>>` UserRepository



4.3.2 Description des interactions entre objets



4.3.3 Description des classes



4. Regroupement des classes

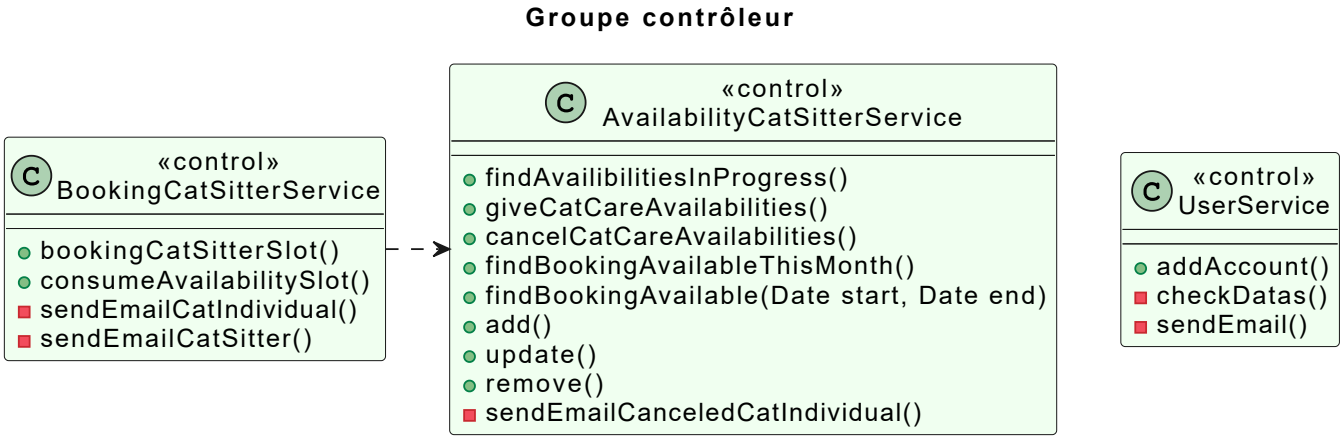
4.1 Groupe domaine

```
PlantUML 1.2023.11beta2
[From string (line 29) ]

@startuml
title Groupe domaine

entity "<<entity>>\nBookingCatSitter" as BookingCatSitter #HoneyDew
entity "<<entity>>\nUser" as User #HoneyDew
entity "<<entity>>\nAvailabilityCatSitter" as AvailabilityCatSitter #HoneyDew

BookingCatSitter : Long id
BookingCatSitter : User catSitter
BookingCatSitter : User catIndividual
BookingCatSitter : Date start
BookingCatSitter : Date end
BookingCatSitter : Date booking
BookingCatSitter : [...]

AvailabilityCatSitter : Long id
AvailabilityCatSitter : User catSitter
AvailabilityCatSitter : Date start
AvailabilityCatSitter : Date end
AvailabilityCatSitter : Integer maxCat
AvailabilityCatSitter : [...]

User : Long id
User : String name
User : String lastname
User : Date birthdate
User : [...]

BookingCatSitter _- User : catSitter
 Syntax Error?
```

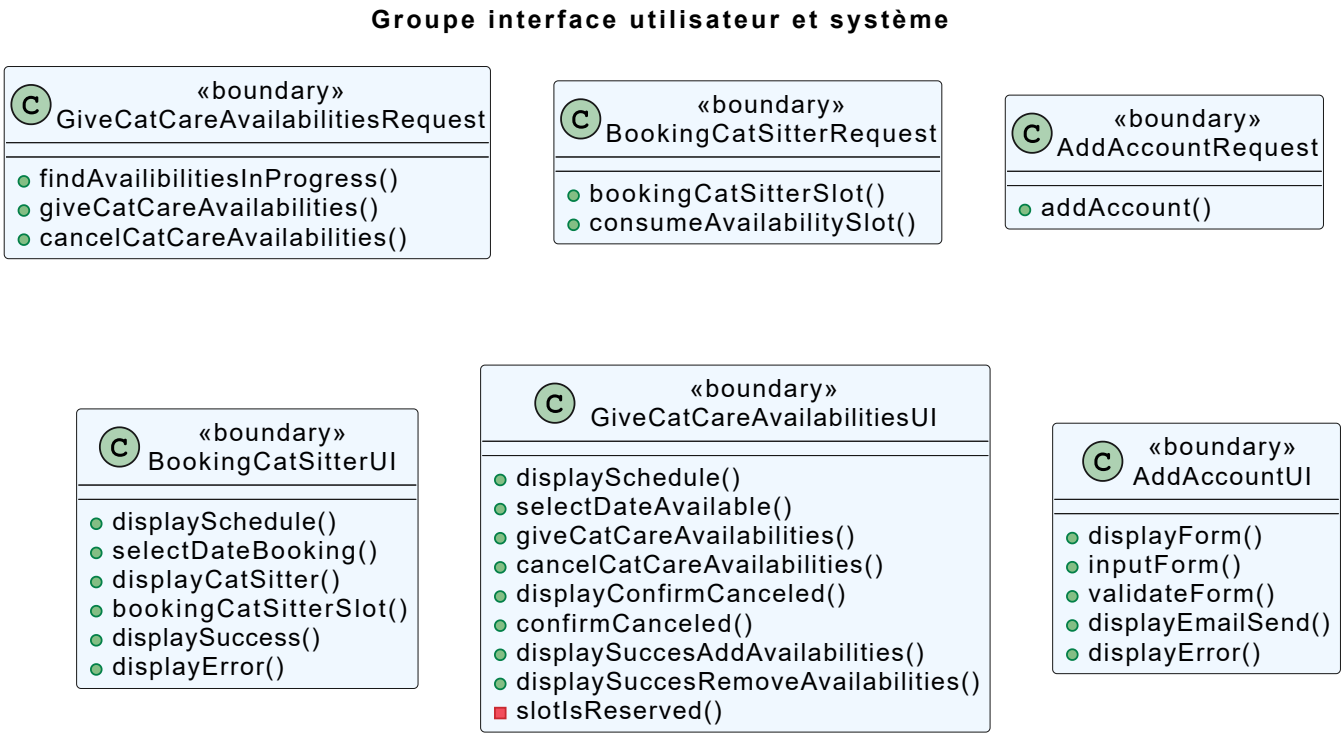
4.2 Groupe domaine et cycle de vie

```
PlantUML 1.2023.11beta2
[From string (line 47) ]

@startuml
title Groupe domaine et cycle de vie

class "<<life cycle>>\nUserRepository" as UserRepository #HoneyDew
class "<<life cycle>>\nBookingCatSitterRepository" as BookingCatSitterRepository #HoneyDew
...
... ( skipping 22 lines )
...
BookingCatSitter : Date booking
BookingCatSitter : [...]

AvailabilityCatSitter : Long id
AvailabilityCatSitter : User catSitter
AvailabilityCatSitter : Date start
AvailabilityCatSitter : Date end
AvailabilityCatSitter : Integer maxCat
AvailabilityCatSitter : [...]

User : Long id
User : String name
User : String lastname
User : Date birthdate
User : [...]

BookingCatSitterRepository ..> BookingCatSitter
AvailabilityCatSitterRepository..> AvailabilityCatSitter
UserRepository ..> User
BookingCatSitter _- User : catSitter
 Syntax Error?
```

4.3 Groupe contrôleur



4.4 Groupe interface utilisateur et système



Bibliographies

Advantages and Disadvantages of Gradle and Maven : https://www.interviewbit.com/blog/gradle-vs-maven/

Ant Design : https://ant.design/

Cypress : https://www.cypress.io/

Gradle : https://gradle.org/

Liquibase : https://www.liquibase.org/

MockK : https://mockk.io/

MSW : https://mswjs.io/

React JS : https://react.dev/

React Testing Library : https://testing-library.com/docs/react-testing-library/intro/

Spring initializr : https://start.spring.io/;

Vite : https://vitejs.dev/guide/

Wiremock : https://wiremock.org/