

# Projet EggFriend

Equipe - Les Amis Des Omelettes :

<https://github.com/TheoGicquel/EggFriend>

## Rôles assignés

- Ryan Barrault : développeur
- Théo Gicquel : architecte logiciel, chef de projet, gestionnaire de la qualité
- Emilie Kermorvant : analyste besoin
- Corentin Quiniou : chef de projet, analyste de besoin

## Répartition du travail

- Ryan Barrault
  - prototypage interfaces graphique de jeu
  - Assistance rédaction cahier des charges
- Théo Gicquel
  - Gestion du contrôle de version sous git
  - Mise en place de l'architecture de développement du projet (maven + Junit)
  - Développement de l'interface utilisateur
  - Développement de la logique interne du jeu
  - Mise en place et rédaction des Tests unitaires
  - Documentation du code source
  - Correction cahier des charges
  - Rédaction diagramme de Classe
  - Rédaction diagrammes des cas d'utilisation
  - Rédaction diagramme Gantt
  - Mise en place CI/CD minimal
  - Mise en place production de rapports de coverage
- Emilie Kermorvant
  - Rédaction cahier des charges
  - Rédaction scénarios de tests
  - Rédaction cas d'utilisations
  - Assistance tests unitaires
- Corentin Quiniou : chef de projet, analyste de besoin
  - Développement de la classe du système de sauvegarde/chargement
  - Rédaction schéma conceptuel de données persistantes
  - Rédaction diagrammes de séquence
  - Assistance rédaction cahier des charges

# CAHIER DES CHARGES

Version : 4

Date : 04/01/2021

<i>Client</i>	<i>Prestataire</i>
Rémy Kessler	Ryan Barrault : développeur Théo Gicquel : architecte logiciel, chef de projet, gestionnaire de la qualité Emilie Kermorvant : analyste besoin Corentin Quiniou : chef de projet, analyste de besoin
<b>Cahier des charges approuvé dans sa version</b> <b>le 04/01/2022 par</b>	

*zone réservée*

# 1 INTRODUCTION

## 1.1 Objet du document

Ce document décrit **tous** les services que doivent rendre le produit et ses livrables et toutes les exigences qu'ils doivent satisfaire.

## 1.2 Portée du document

Ce document est destiné à formaliser le besoin du client Kessler dans le cadre du projet Tamagotchi.

## 1.3 Terminologie

Terme	Description
Tamagotchi	Petit animal virtuel dont on doit s'occuper.
Manger	Permet de recharger la barre de vie du tamagotchi.
Dormir	Permet de recharger la barre de sommeil du tamagotchi.
Jouer	Permet de recharger la barre de fun, d'augmenter la qualité de vie du Tamagotchi.
Humeur	Témoigne de son bonheur et de sa viabilité.
Type	Espèce de l'animal.
Git	Système de contrôle de version assurant la traçabilité et l'organisation collaborative du code source du projet.
Joueur	Utilisateur final de l'application

## 1.4 Abréviations

Abréviations	Signification	Libellé
VIT	Vitale	Exigences fonctionnelles ou non fonctionnelles indispensables
IMP	Importante	Exigences souhaitées mais non exigées

MIN	Mineure	Exigences non exigées immédiatement, mais qui devront être prises en compte ultérieurement par le produit (impact sur l'évolutivité)
-----	---------	--

## 2 LES OBJECTIFS DU PRODUIT

---

### 2.1 Définition du produit

Le produit est un jeu vidéo mobile dont le but est d'élever un animal virtuel.

Il est possible de lui donner un nom et de choisir son ~~type~~ espèce.

Il est possible de le nourrir, le faire dormir ou le faire jouer, afin de le maintenir en vie.

Le jeu s'arrête lorsque l'animal meurt, c'est-à-dire lorsque sa barre de vie tombe à 0.

### 2.2 Contexte économique du produit

La première version du produit sera faite pour ordinateur et pour tout type de système d'exploitation, l'objectif final sera de le déployer sur toutes les plateformes, mobile ou fixe et ainsi, pouvoir toucher le maximum de monde.

Notre cœur de cible sera les joueurs mobiles, jeunes (7-30 ans).

L'application sera proposée gratuitement ~~avec des cosmétiques payants~~.

### 2.3 Contexte d'exploitation du produit

Le logiciel va être déployé dans le cadre d'une campagne de sensibilisation à la qualité de vie des animaux. Le but est de montrer au grand public qu'un animal a des besoins physiques et émotionnels.

Le produit sera disponible gratuitement ~~sur tous les stores~~, notre objectif étant de soutenir la cause animale, une partie des recettes sera versée à des associations de protection des animaux.

## 3 EXIGENCES SUR LE PRODUIT

---

### 3.1 Capacités Fonctionnelles

#### 3.1.1 Description des fonctionnalités

##### **-donner un nom à l'animal :**

Le nom sera choisi au démarrage d'une nouvelle partie, sera composé de 20 caractères maximum (espace inclus) et utilisé lors de dialogues. Il ne pourra être modifié.

##### **-choisir son espèce :**

Au même titre que le nom, l'espèce va être choisie au début de la partie et aura une incidence sur les caractéristiques du jeu.

##### **-nourrir :**

Le pourcentage de la barre de nourriture va se réduire au fil du temps. L'augmentation du pourcentage de la barre de nourriture se fera par le biais de plats et de différents aliments. Selon le choix fait par

l'utilisateur, la barre se remplira plus ou moins. Chaque espèce a des spécificités, certains plats augmenteront plus manifestement leur pourcentage de satiété que d'autres.

Par exemple, pour le mouton, la nourriture optimale à lui donner est de l'herbe.

De même, le chien verra son pourcentage de satiété augmenter plus rapidement avec un os.

Le chat sera très satisfait avec des croquettes.

Enfin, le robot ne mange que des écrous.

Si la nourriture choisie correspond à celle préférée par l'espèce, la consommation d'un repas augmenterait son niveau de satiété actuel de 40%.

#### **-dormir :**

Le pourcentage de la barre de sommeil va se réduire au fil du temps. Actionner le bouton "dormir" augmentera le pourcentage de 50% ~~interrompt toute action réalisée par le tamagotchi et rend impossible toute autre activité tant qu'il dort.~~

~~L'augmentation du pourcentage de sommeil est proportionnelle au temps de sommeil.~~

~~Il est possible de réveiller le Tamagotchi grâce au bouton "stop". Sinon, celui-ci se réveille tout seul lorsque le pourcentage est au maximum (100%).~~

#### **-jouer :**

Faire jouer le Tamagotchi augmentera sa barre de joie, mais causera une diminution de 20% de sa barre d'énergie et de 10% de sa barre de propreté, et de 5% de sa barre d'énergie.

~~L'augmentation du pourcentage de joie est proportionnelle au temps de jeu.~~

~~L'activité changera en fonction de l'animal choisi :~~

~~Chats : souris en plastique~~

~~Chiens : frisbee~~

~~Chèvre : la faire courir, station de grattage~~

~~Robot : calcul de probabilités quantiques~~

#### **-laver :**

Laver le tamagotchi fera augmenter sa barre de propreté de 45%, ~~il en sortira quand sa barre de propreté sera à 100%.~~

~~L'augmentation du pourcentage de propreté est proportionnelle au temps passé sous la douche.~~

#### **-rafraîchir :**

~~L'utilisateur peut demander une mise à jour des données de jeu et de l'interface graphique~~

~~La mise à jour de l'interface graphique est effectuée à chaque action de l'utilisateur et a un délai de quelques secondes.~~

#### **-stopper :**

~~Appuyer sur le bouton "Stop" cause l'arrêt des activités non instantanées du Tamagotchi.~~

#### **-sauvegarder :**

Les données du jeu sont sauvegardées automatiquement toutes les 5 minutes.

Les données seront également sauvegardées lorsque le joueur quitte la partie via le bouton "Quitter"

#### **-quitter le jeu :**

L'utilisateur peut quitter le jeu en actionnant le bouton "quitter".

### 3.1.2 Interopérabilité

### 3.1.3 Conformité réglementaire

Le jeu est PEGI 7 ~~avec la mention d'achat intégrés.~~

## 3.2 Exigences non fonctionnelles

### 3.2.1 Fiabilité

Le jeu intégrera un système de sauvegarde automatique s'exécutant toutes les 5 minutes.

### 3.2.2 Sécurité

Aucune donnée d'utilisateur (nom, temps de jeu, données bancaires..) ne sera diffusée et aucun accès internet ne sera nécessaire pour le fonctionnement du programme.

~~Entre-autres, toutes les données liées au jeu seront conservées localement sur la machine de l'utilisateur final.~~

Le jeu aura la capacité de créer et de modifier ~~un fichier~~ de ~~sauvegarde~~ ~~situé~~ dans le répertoire de l'application.

### 3.2.3 Facilité d'utilisation

L'application est accessible pour toutes les tranches d'âge autorisées.

### 3.2.4 Rendement

Un seul utilisateur à la fois, le temps de fonctionnement est illimité dans la limite du raisonnable.

### 3.2.5 Maintenabilité

L'application sera régulièrement mise à jour afin d'apporter une correction aux éventuels bugs et de nouvelles fonctionnalités.

### 3.2.6 Portabilité

Le jeu sera disponible sur PC/Mac OS/Linux, ~~et nécessite la présence d'une machine virtuelle Java en version supérieure ou égale à 1.8 afin de pouvoir exécuter le programme.~~

## 3.3 Exigences concernant le développement du produit

### 3.3.1 Objectifs de délais

L'application sera développée pour le 4 janvier 2022.

### 3.3.2 Objectifs de coûts

Les coûts doivent être contrôlés, un plafond est fixé à 15 000€ pour le lancement, 30 000€ pour la maintenabilité, les mises à jour et les versions futures.

Les coûts sont répartis dans le salaire de 4 personnes, l'infrastructure et le matériel.

### 3.3.3 Exigences de réalisation

L'application doit être fluide et ne pas contenir de bugs. Elle doit être claire et facile à prendre en main. En clair, elle doit être adaptée aux plus jeunes.

## 4 SYNTHÈSE DES EXIGENCES

### 4.1 Hiérarchisation des exigences fonctionnelles

Nom fonction	Importance
Create	100%
Type	100%
Name	100%
Eat	50%
Sleep	50%
Play	50%
Wash	50%
Get	30%
Set	30%

### 4.2 Hiérarchisation des exigences non fonctionnelles

Exigence	Importance
Sécurité	100%
Fiabilité	100%
maintenabilité	70%
facilité d'utilisation	50%
Rendement	30%
portabilité	10%

humeur : très joyeux, joyeux, normal, triste, très triste

boutons :

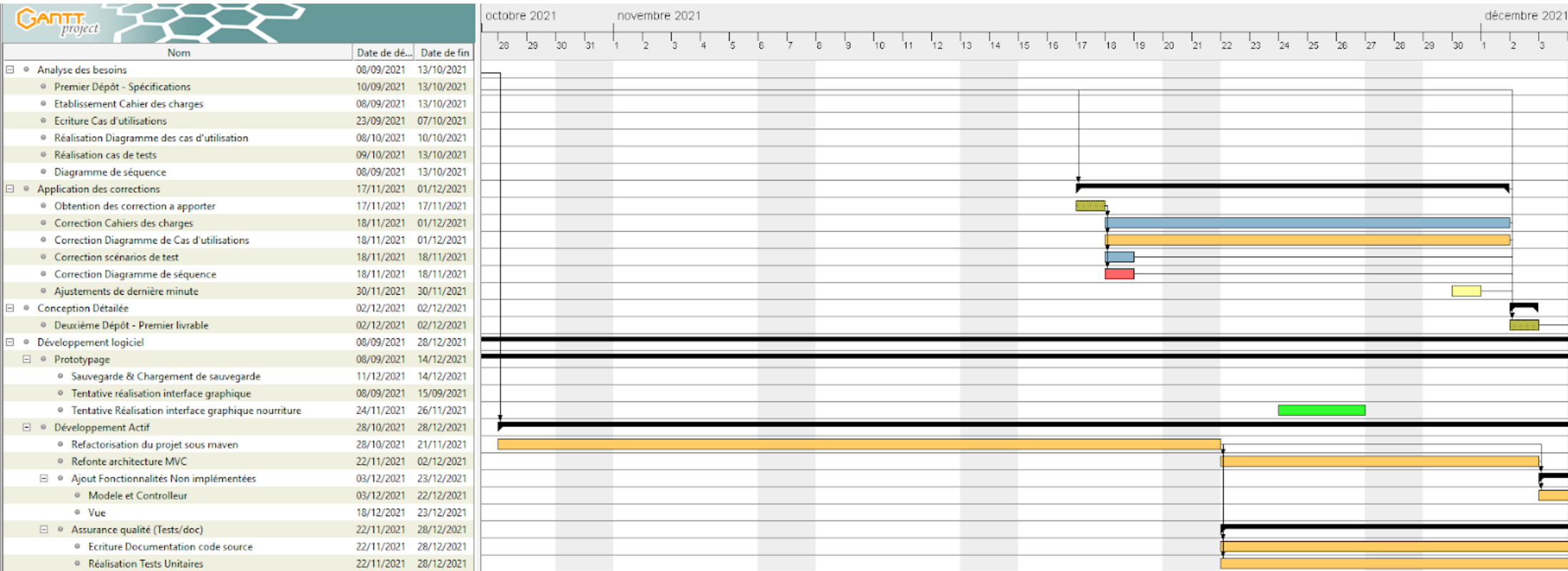
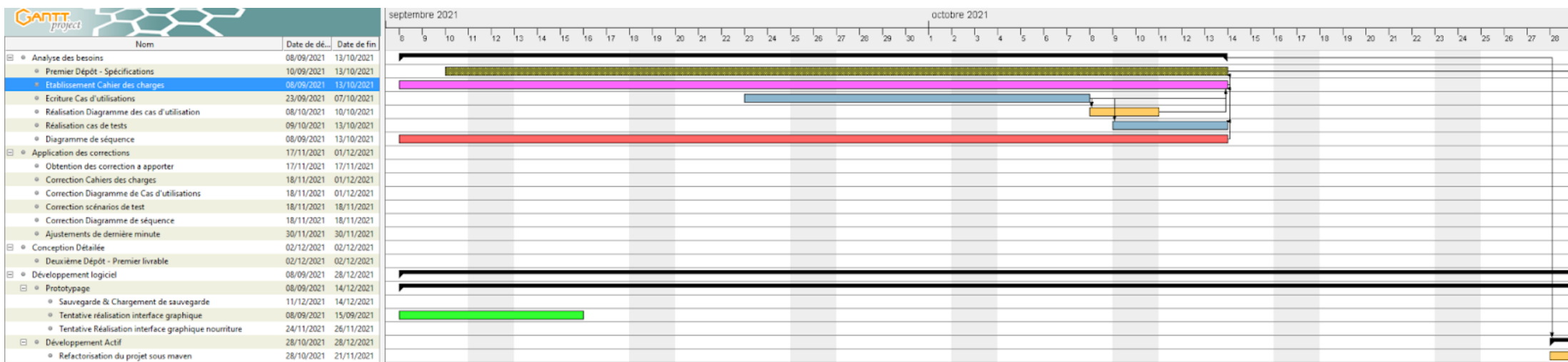
Dans le classe Animal :

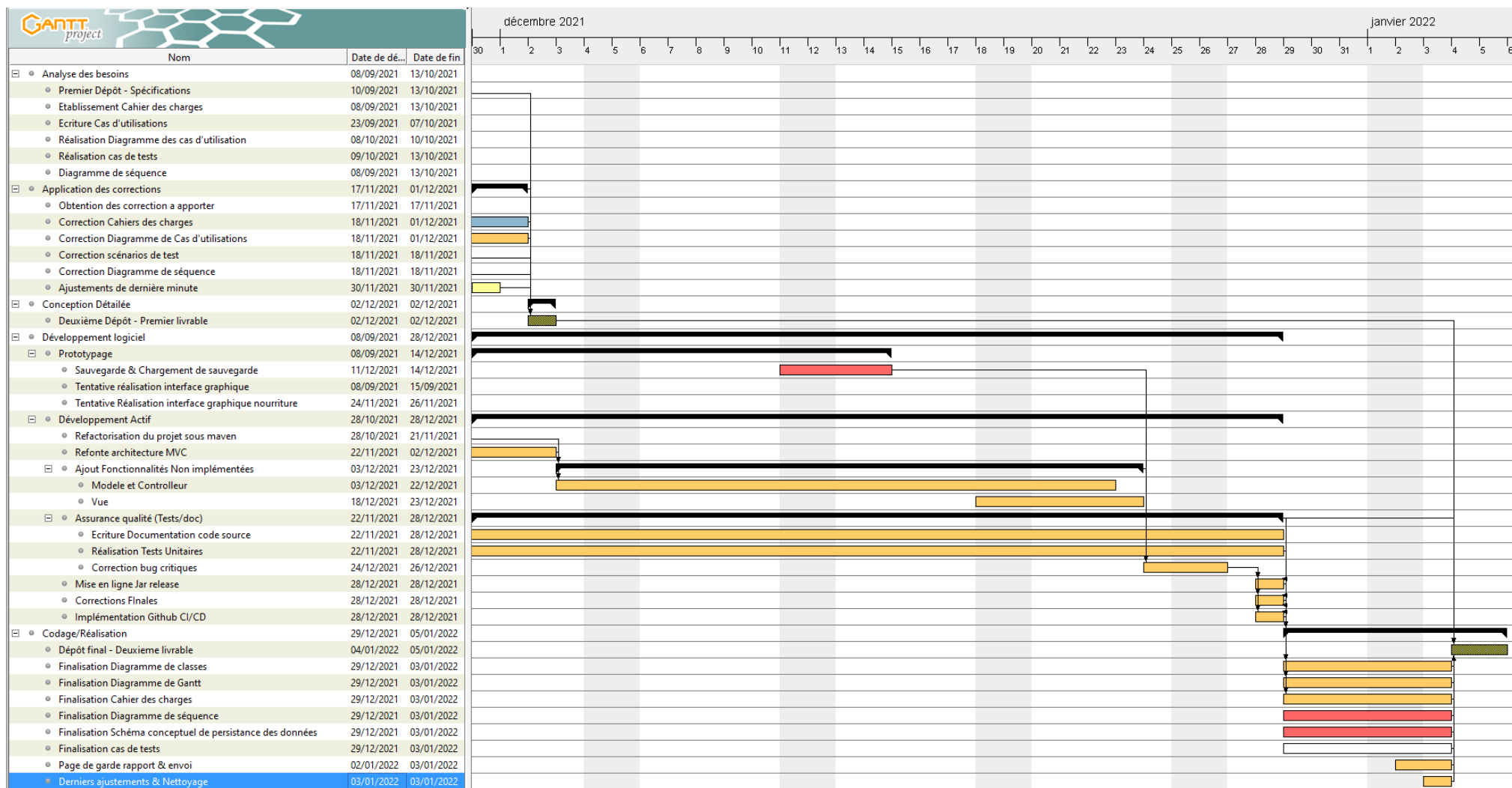
manger : +40% barre nourriture +5% barre bonheur (si bonne nourriture) ,  
dormir : +50% barre énergie, ~~met en veille les boutons pendant 300-sec,~~ +5% barre bonheur  
laver : +20% barre de propreté -5% barre bonheur  
jouer : +40% barre de bonheur, -20% barre energie, -10% barre de propreté

Dans la classe Frame :









quitter : ferme la fenêtre de jeu et sauvegarde la partie  
~~rafraîchir : mets à jour les barres~~  
~~sélectionnez : choisir le type de nourriture~~  
~~effectuer : valide l'action~~  
~~à propos : ouvre une fenêtre donnant des informations sur le type de l'animal, sa durée de vie etc~~







Légende :

 Contact Client	 Corentin Quiniou	 Théo Gicquel	 Emilie Kermorvant	 Ryan Barrault
 Gicquel T & Corentin Q	 Emilie K & Ryan B	 Groupe Complet		

### **Cas d'utilisation 1** : jouer en créant une partie

Acteurs	Usager
Événement déclencheur	Double cliquer sur l'icône
Parties prenantes et leurs intérêts	Usager : s'amuser et se détendre Entreprise :
Niveau	
Portée	Données usager pour la création d'une nouvelle partie -nom : pseudo du Tamagotchi, longueur maximale 20 caractères -type : au choix parmi 4 types(chat, chien, chèvre, robot)
Pré-conditions	Pouvoir chercher les autres Tamagotchi. Le nouveau Tamagotchi n'est pas déjà existant.
Post-conditions	Le nouveau Tamagotchi est créé.
Scénario nominal	1) L'utilisateur remplit les champs données. 2) La partie commence, l'utilisateur peut actionner différents boutons qui auront une influence sur l'état du Tamagotchi. 3) L'utilisateur joue un certain temps puis sauvegarde la partie et quitte le jeu.
Extensions	<del>Si l'utilisateur veut par exemple nourrir le Tamagotchi alors que sa barre de nourriture est pleine, un message s'affiche pour lui dire que la barre est pleine.</del> <del>Si l'utilisateur veut le faire jouer alors que sa barre de vie est proche de 0, un message s'affiche pour lui dire qu'il faut plutôt manger ou dormir.</del>
Contraintes	Aucune réquisition des données de l'utilisateur

### **Cas d'utilisation 2** : reprendre sa partie

Acteurs	Usager
Événement déclencheur	Double cliquer sur l'icône
Parties prenantes et leurs intérêts	Usager : s'amuser et se détendre Entreprise :
Niveau	
Portée	Données usager reconduites
Pré-conditions	Une première session de jeu a eu lieu.
Post-conditions	Les nouvelles données sont sauveées.
Scénario nominal	1) L'usager retrouve son Tamagotchi en ouvrant la fenêtre de jeu. 2) La partie recommence, les données sont re téléchargées, l'usager peut actionner différents boutons qui auront une influence sur l'état du Tamagotchi. 3) L'usager joue un certain temps puis sauvegarde la partie et quitte le jeu.
Extensions	<del>Si l'usager veut par exemple nourrir le Tamagotchi alors que sa barre de nourriture est pleine, un message s'affiche pour lui dire que la barre est pleine.</del> <del>Si l'usager veut le faire jouer alors que sa barre de vie est proche de 0, un message s'affiche pour lui dire qu'il faut plutôt manger ou dormir.</del>
Contraintes	Aucune réquisition des données de l'utilisateur

### **Cas d'utilisation 3** : fin de partie

Acteurs	Usager/temps
Événement déclencheur	
Parties prenantes et leurs intérêts	Usager : Entreprise :
Niveau	
Portée	Données usager reconduites
Pré-conditions	Une ou plusieurs sessions de jeu ont eu <del>es</del> lieu.
Post-conditions	Les données sont supprimées.
Scénario nominal	La fin de jeu a lieu lorsque le Tamagotchi meurt. Il peut mourir <del>de mort naturelle au bout d'un certain temps, ou alors</del> suite à la négligence du joueur (pas de nourriture, pas de sommeil...)
Extensions	
Contraintes	Aucune réquisition des données de l'utilisateur

Scénario test 1 : Création d'une partie.

Acteurs	Usager
Évènement déclencheur	Cf cas d'utilisation 1: création d'une partie
Parties prenantes et leurs intérêts	Usager qui souhaite commencer à jouer
Niveau	Stratégique
Portée	Nouveau Tamagotchi
Pré conditions	Aucune
Post conditions	Nouveau Tamagotchi créé
Test	Réponse
Toutes les données d'entrées vide : prénom = « » (chaîne vide)	Le tamagotchi est nommé «???».

Scénario test 2 : Aucune nourriture sélectionnée.

Acteurs	Usager
Évènement déclencheur	L'utilisateur souhaite nourrir le Tamagotchi.
Parties prenantes et leurs intérêts	Usager : donner à manger à son Tamagotchi
Niveau	Important
Portée	Partie en cours
Pré conditions	Le Tamagotchi existe, la barre de nourriture est en dessous de 100 %.
Post conditions	La partie reprend son cours.
Test	Réponse
Aucune nourriture n'est sélectionnée dans le menu déroulant et bouton « À table ! » actionné	Un message d'erreur s'affiche. Le pourcentage d'énergie et de nourriture est inchangé.

Scénario test 3 : Bouton « dormir actionné ».

Acteurs	Usager
Évènement déclencheur	Souhait de faire dormir le Tamagotchi
Parties prenantes et leurs intérêts	Usager : augmenter le pourcentage de la barre de sommeil du Tamagotchi
Niveau	Important
Portée	Partie en cours
Pré conditions	Le Tamagotchi existe, sa barre de sommeil est en dessous de 100 %.
Post conditions	Le Tamagotchi peut reprendre ses activités.
<b>Test</b>	<b>Réponse</b>
Bouton « dormir » actionné	Le pourcentage de sommeil et de bonheur du tamagotchi augmente.

Scénario test 4 : Méthode sur une donnée qui est déjà à 100 %.

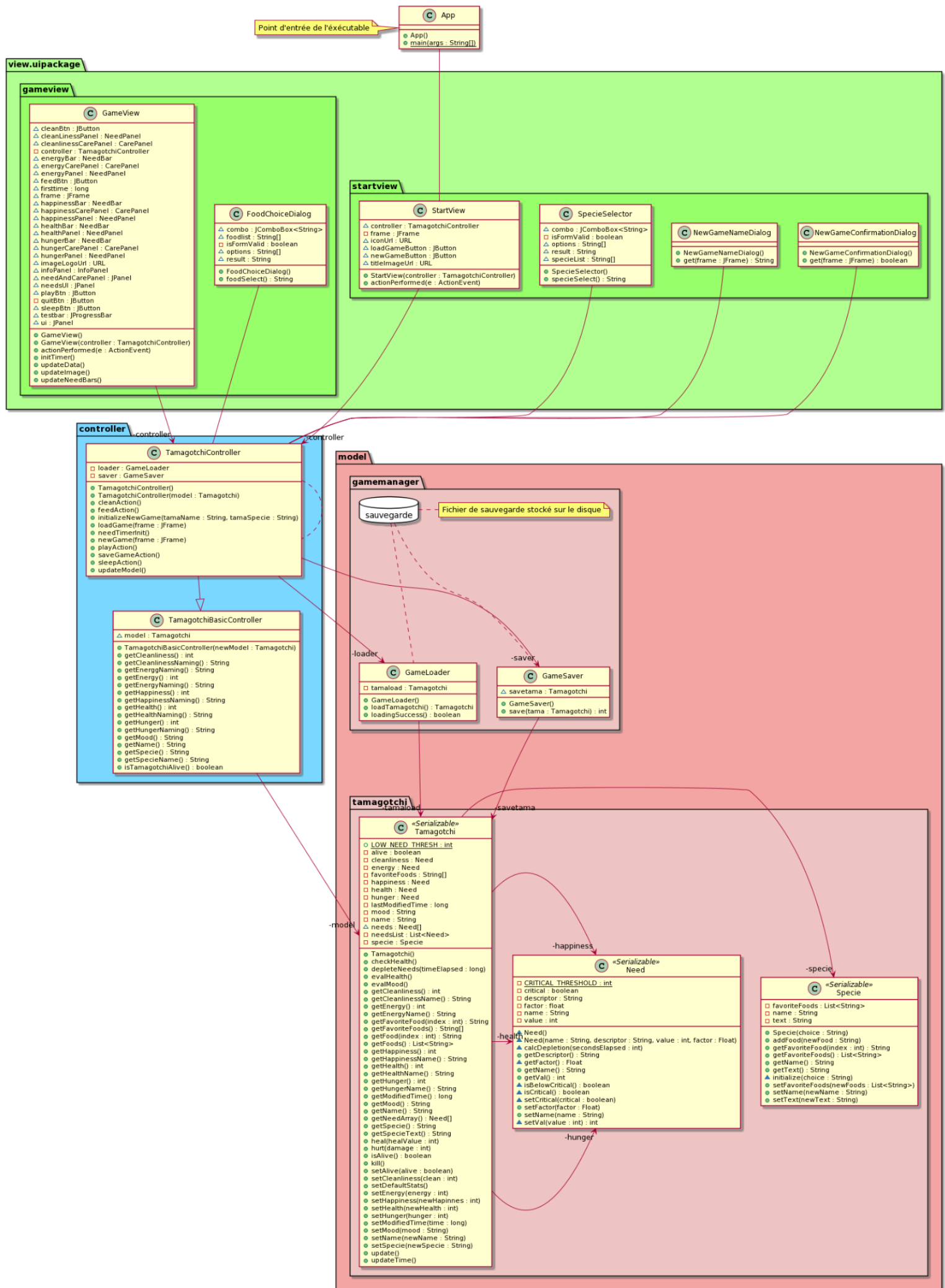
Acteurs	Usager
Évènement déclencheur	Souhait d'améliorer la qualité de vie du Tamagotchi
Parties prenantes et leurs intérêts	Usager : jouer et maintenir des pourcentages corrects
Niveau	Inutile
Portée	Partie en cours
Pré conditions	Le Tamagotchi existe, la donnée concernée est à 100 %
Post conditions	Le Tamagotchi peut reprendre ses activités.
<b>Test</b>	<b>Réponse</b>
Actionner un bouton qui concerne une donnée qui a déjà 100 %.	Il n'y a pas de changement.

### Scénario de tests

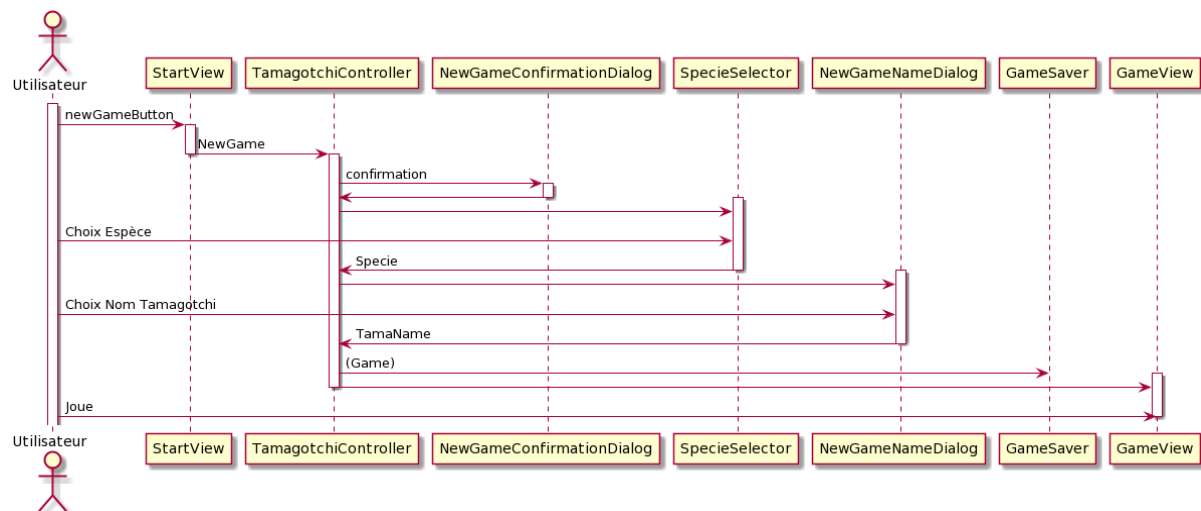
Test	Réponse
Actionner le bouton “charger partie” sans avoir déjà une partie enregistrée	Message d’erreur
Bouton “ nouvelle partie “ actionné, puis bouton “no” sélectionné.	Le nouvelle fenêtre se ferme, retour au menu d'accueil.
Nombre de données d’entrées supérieur à 20, prénom avec plus de 20 caractères	Saisie effacée Demande d’une nouvelle saisie
Toutes les données d’entrées vide : prénom = “ ” (chaîne vide)	Prénom par défaut : “???”
Bouton “ Nourrir ” actionné	Une nouvelle fenêtre avec un menu déroulant apparaît. L’utilisateur choisit la nourriture à donner. Dans le cas où la nourriture est adaptée à l’espèce, la barre de faim augmente de 40%. Si la nourriture n’est pas adaptée, rien ne se passe (comme si l’action ne s’est pas et ne peut pas s’effectuer).
Bouton “ Sieste “ actionné.	Le pourcentage de la barre d’Énergie augmente de 50%. Le pourcentage de la barre de Bonheur augmente de 5%.
Bouton “Jouer” actionné.	Le pourcentage de la barre de faim diminue de 10%.

	<p>Le pourcentage de la barre de faim diminue de 20%.</p> <p>Le pourcentage de la barre de bonheur augmente de 40%.</p>
<p>Actionner un bouton qui concerne une donnée qui a déjà 100 %.</p>	<p>Il n'y a pas de changement sur cette donnée. De plus une barre d'action non remplie qui doit atteindre plus que 100% suite à une action stagne à 100% qui est le maximum.</p> <p>Cependant, les diminutions liées à l'abaissement du besoin sont effectuées</p>
<p>Ne rien faire.</p>	<p>-Le pourcentage de la barre de Vie augmente de 10% toutes les 60</p> <p>-Le pourcentage de la barre de faim diminue de 4% toutes les 60 secondes.</p> <p>-Le pourcentage de la barre d'Énergie diminue de 4% toutes les 60 secondes.</p> <p>-Le pourcentage de la barre de Propreté diminue de 3% toutes les 60 secondes.</p> <p>-Le pourcentage de la barre de Bonheur diminue de 3% toutes les 60 secondes.</p> <p><u>Concernant le statut:</u></p> <p>-Lorsque la barre de Faim atteint 35% ou moins, le statut affiche "Affamé"</p> <p>-Lorsque la barre d'Énergie atteint 35% ou moins, le statut affiche "Fatigué"</p> <p>-Lorsque la barre de Propreté atteint 35% ou moins, le statut affiche "sale"</p> <p>-Lorsque la barre de Bonheur atteint 35% ou moins, le statut affiche "déprimé"</p>
<p>Tamagotchi avec un pourcentage de vie égal à 0.</p>	<p>Le statut du Tamagotchi est passé à "mort", tous les boutons n'ont aucun effet sauf le bouton quitter.</p>
<p>Bouton "Quitter" actionné.</p>	<p>Ferme l'application en faisant une sauvegarde des données.</p>

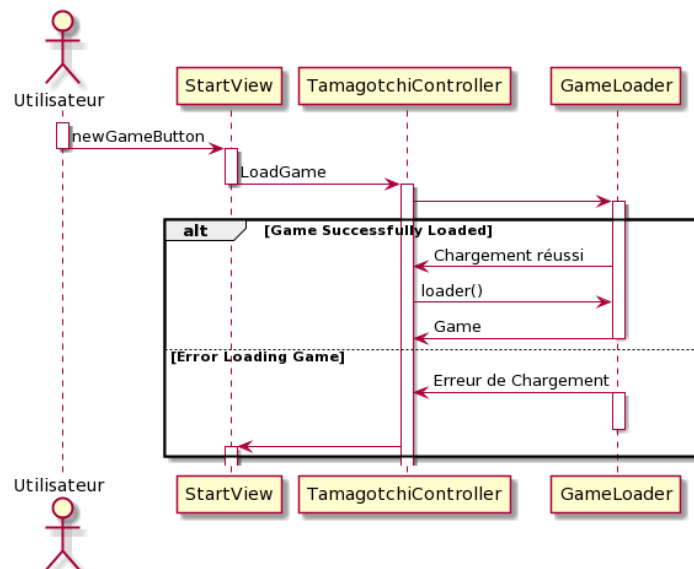




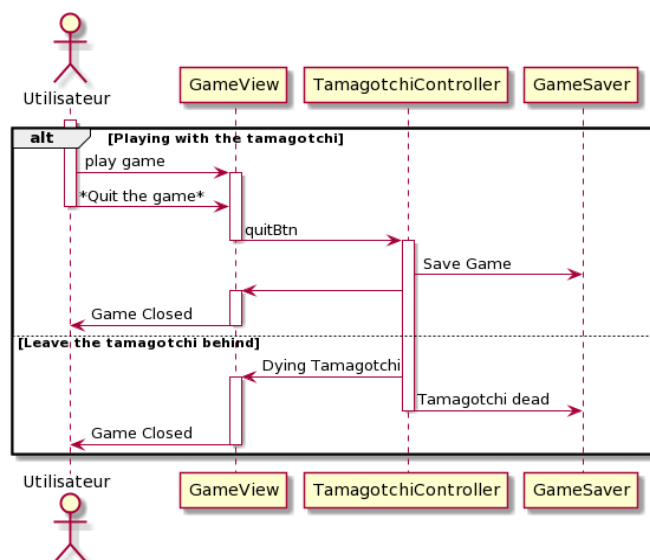
## Création d'une partie



## Charger une partie



## Fin de partie



# Schéma de persistance des données

