Étude des données, persistances et tests

DIBERDER Evan
BUAN Kilian
REGNIER Alix
COSNIER Quentin

I - Identification des données persistantes

États et statistiques

Un tamagotchi a un rythme de vie calqué sur celui d'un humain. Il a besoin de manger régulièrement et en quantité raisonnable, de dormir une fois par jour, de jouer afin d'être heureux, de se laver, etc...Les statistiques d'un tamagotchi sont réparties selon ses besoins:

- Sommeil
- Faim
- Joie
- Hygiène
- Besoins naturels

Afin de déterminer si le tamagotchi est en bonne santé, ces statistiques permettent de définir son état physique et son état psychique. Ces états peuvent prendre 5 valeurs différentes:

1. Très bon
2. Bon
3. Moyen
4. Mauvais
5. Très mauvais

Au début de la partie, les états du tamagotchi sont à **Très bon**; selon les actions du joueur, les statistiques du tamagotchi vont évoluer. L'évolution de ces statistiques influencera l'état physique et psychique du tamagotchi.
Lorsqu'une statistique tombe à 0, l'état qui lui correspond descend d'un cran. Lorsqu'une statistique est remontée au maximum, l'état qui lui correspond remonte d'un cran.

État	**Besoin**		
Physique	Sommeil	Faim	Hygiène
Psychique	Besoins naturels	Joie	Hygiène

Toutes les 24h, si un état est négatif (en dessous de **Moyen**), le tamagotchi perd de la santé en fonction du niveau de cet état.
-1 lorsque Mauvais et -2 lorsque **Très mauvais**.
Quand l'état est positif, la santé remonte selon le même système.

Lorsque la santé du tamagotchi tombe à 0, la partie est perdue.

Temps et statistiques

Les statistiques ont des valeurs maximales différentes et elles sont régies par la fréquence à laquelle le tamagotchi est censé dormir, manger, jouer,...
Le jeu effectue une actualisation des statistiques du tamagotchi toutes les 5 minutes. Une actualisation consiste à enlever 7 à chaque statistique (explication ci-dessous du choix des nombres).

Nous allons prendre l'exemple de la faim; on estime qu'un tamagotchi peut tenir 24h sans manger sans que cela soit néfaste pour son état physique, ceci est son cycle limite de faim. On prend une valeur arbitraire de 2000 pour la valeur maximale d'une statistique qui a donc un cycle limite de 24h et on cherche pour quelle valeur de décrémentation à chaque actualisation la valeur passera en dessous de 0.

Soit **N** le nombre d'actualisation en 24h. Sachant qu'une actualisation s'effectue toutes les 5 minutes, cela fait:

$$N := \frac{24\times60}{5} = 288$$

On cherche alors un entier naturel **x**, notre valeur de décrémentation:

$$2000 - x \cdot N \leq 0$$
$$2000 - 288x \leq 0$$
$$x \geq \frac{2000}{288} \approx 6.944..$$

x étant un entier naturel, x prend alors la valeur entière supérieure ou égale à 6.944, c'est-à-dire 7.

À partir de ce raisonnement on a juste à choisir la valeur maximale des statistiques selon la fréquence à laquelle on doit satisfaire les besoins du tamagotchi *(e.g. pour un cycle limite de 12h, on prendra une valeur maximale de 1000)*.

Le principe d'un tamagotchi est que celui-ci continue de "vivre" même lorsque le jeu est fermé. Nous rappelons que le temps influe uniquement sur les statistiques des tamagotchi; il faut donc pouvoir calculer les nouvelles statistiques d'un tamagotchi entre le moment où le

joueur a arrêté pour la dernière fois une partie et le moment où il la charge. Il suffit donc de calculer le nombre d'actualisation qu'il aurait dû se produire entre ces 2 moments. Le jeu considère un moment, une date comme un *Unix timestamp*[1]; la différence de temps **Δt** s'exprime donc en secondes et on peut calculer la nouvelle valeur d'une statistique **x'** à partir de son ancienne valeur **x**.

$$x' := x \ - \ 7 \cdot \frac{\Delta t}{300}$$

Notons que 300 est la conversion de l'intervalle d'actualisation en secondes (300 = 5x60).

Les nouvelles statistiques calculées, on peut donc calculer les états du tamagotchi en conséquence, puis déterminer s'il a succombé, etc...
Cela nous permet donc d'assurer la reprise du jeu, comme si le joueur n'avait pas quitté la partie ou éteint le jeu.

II - Persistance des données

Voici une liste des données persistantes qui correspondent au tamagotchi :

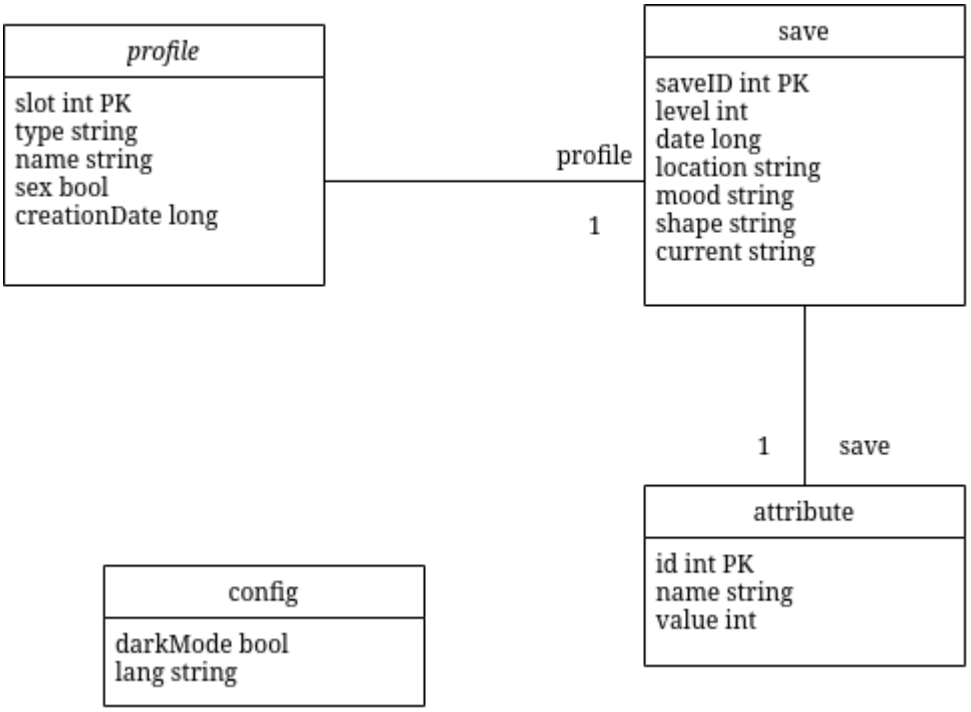
Données d'une partie	
Données	**Type**
Identifiant de la partie	entier non signé
Type du tamagotchi	entier
Étape d'évolution du tamagotchi	entier
Nom du tamagotchi	chaîne de caractères
Sommeil	entier
Faim	entier
Joie	entier
Hygiène	entier
Besoins naturels	entier
État physique	entier
État mental	entier

[1] https://fr.wikipedia.org/wiki/Heure_Unix

Santé	entier
Date du début de la partie	entier long non signé
Date du dernier arrêt de la partie	entier long non signé

Le jeu manipule plusieurs données différentes comme la langue ou les parties; l'utilisation d'une base de données permet d'en assurer la persistance. Une partie comporte elle-même plusieurs données.

Diagramme représentant la structure de la base de donnée



La partie stockant les données liée à la sauvegarde est constituée de trois tables. La table **profile** contient les données définissant la sauvegarde d'un joueur. La table **save** est incrémentée à chaque sauvegarde d'un profil en ajoutant un nouveau tuple qui enregistre le nouvel état du tamagotchi. La valeur de chaque attribut d'une sauvegarde est stockée dans une table **attribute**.

La table **config** constitue les données de l'application comme la langue sélectionnée ou encore si l'application est en mode sombre. Cette persistance des données permet

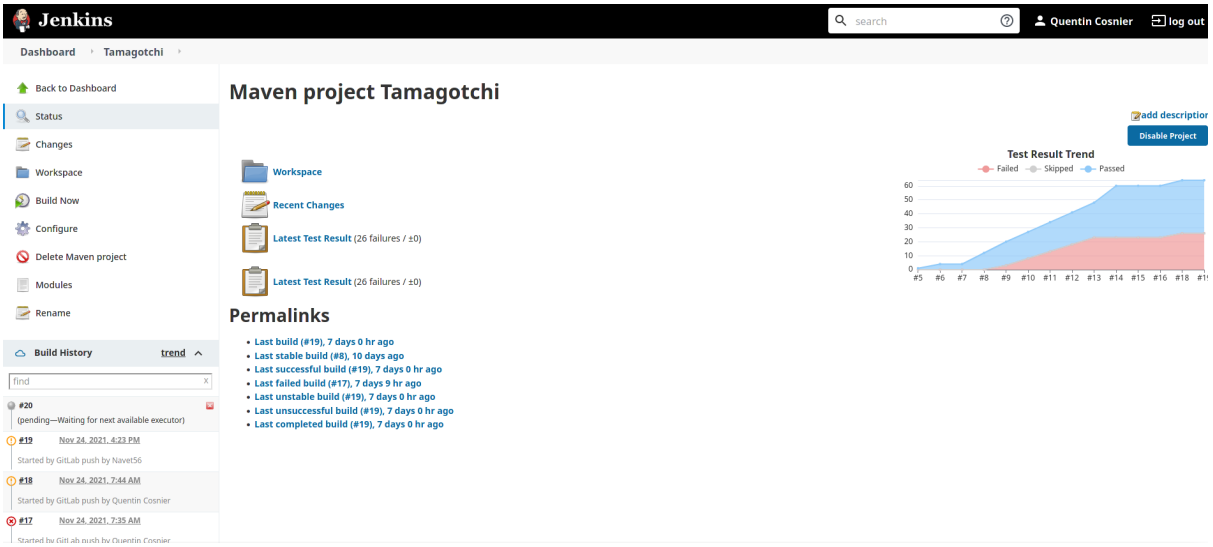
d’enregistrer les préférences de l’utilisateur pour ne pas avoir à les définir à chaque lancement de l’application.

Le choix d’une base de données plutôt qu’une sauvegarde unique peut être justifié par la volonté de garder une trace de l’évolution du tamagotchi. Cette implémen=!
$tation par base de données peut permettre l’intégration d’un journal. Ce journal pourra permettre de retracer l’évolution du tamagotchi au fil des jours ou des semaines.

III - Tests d’intégration

Les tests sont réalisés avec l’outil Maven. Il permet l’automatisation de la production ainsi que des tests unitaires. Les tests sont réalisés avec JUnit.

Le projet étant hébergé via git, il est aussi lié à un projet Jenkins. Cet outil Jenkins effectue les tests unitaires et vérifie l’intégrité du code à chaque push via git. Cela permet une trace des différents changements effectués par les membres de l’équipe et ainsi détecter l’implémentation de bugs lors d’un ajout de code.


Jenkins
search
Quentin Cosnier
log out
Dashboard
Tamagotchi
Back to Dashboard
Status
Changes
Workspace
Build Now
Configure
Delete Maven project
Modules
Rename
Build History
trend
find
#20
(pending—Waiting for next available executor)
#19
Nov 24, 2021, 4:23 PM
Started by GitLab push by Navet56
#18
Nov 24, 2021, 7:44 AM
Started by GitLab push by Quentin Cosnier
#17
Nov 24, 2021, 7:35 AM
Started by GitLab push by Quentin Cosnier
Maven project Tamagotchi
add description
Disable Project
Workspace
Recent Changes
Latest Test Result (26 failures / ±0)
Latest Test Result (26 failures / ±0)
Test Result Trend
Failed
Skipped
Passed
60
50
40
30
20
10
0
#5
#6
#7
#8
#9
#10
#11
#12
#13
#14
#15
#16
#18
Permalinks
Last build (#19), 7 days 0 hr ago
Last stable build (#8), 10 days ago
Last successful build (#19), 7 days 0 hr ago
Last failed build (#17), 7 days 9 hr ago
Last unstable build (#19), 7 days 0 hr ago
Last unsuccessful build (#19), 7 days 0 hr ago
Last completed build (#19), 7 days 0 hr ago