BillySmart

Estelle Fayolle

Crystal Bouchez

Alexandre Moreau

Victor Grandclement

Lucas Azoulay

1. Présentation de l’objet

BillySmart est une gamelle connectée à une application qui permet de réguler et contrôler la consommation de croquettes de votre animal.

Dans un premier temps nous avons cherché une identité visuelle pour la marque.

Recherche de l’identité visuelle de la marque :

Propositions de logo :

Notre choix final s’est porté sur :

Nous avons choisi le vert car c’est la couleur qui se rapprochait le plus de notre identité visuelle.

Le vert représente la bonne santé, l’écologie, la vie et la nature. Nous voulons promouvoir un mode de vie sain pour les animaux domestiques.

2. Manuel d’utilisation

A. Introduction

Le BillySmart se connecte à une application mobile pour pouvoir contrôler au mieux votre gamelle. Des options permettant de régler la fréquence et la quantité de croquettes versées sont accessibles via la page de configuration.

B. Mise en route

Pour mettre en route BillySmart, il faut rentrer le code de votre gamelle dans votre application, l’associer à un animal via un collier actuel, remplir le réservoir de croquettes et programmer vos créneaux et quantités de distribution.

C. Vos animaux et BillySmart

Votre animal a besoin de réguler son alimentation : BillySmart peut vous aider. Un fois la gamelle connectée à votre application, il suffit d’équiper votre animal avec le collier fourni. Par la suite, nous envisageons une reconnaissance de la puce RFID présente dans le cou de l’animal s’il est pucé.

D. Les futures fonctionnalités

Balance :

L’ajout de la balance permettra de peser les croquettes réellement mangées par l’animal, et aussi de rendre la gamelle “multi-animal”. Grâce au RFID, la gamelle saura quel animal a mangé et en quelle quantité.

Une Webcam :

Prendre des photos de votre chat en train de manger ou quand il a fini, avoir une galerie photos et vidéos en ligne de votre animal.

Notifications :

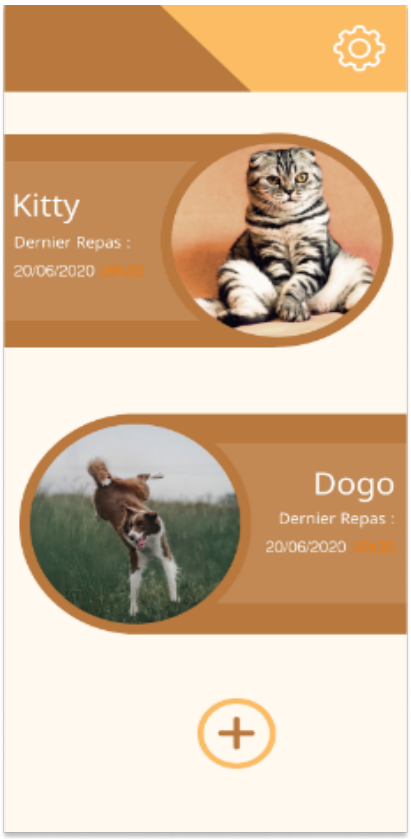
Une notification apparaîtra lorsque l’animal aura ouvert la trappe de sa gamelle, permettant ainsi à l’utilisateur de se connecter à la caméra s’il le souhaite.

Lancer de friandise :

Quand vous détecté que votre animal est proche de votre BillySmart vous pouvez le visualiser avec la webCam et lui lancer des friandises.

Parler avec son chat :

Avec un micro intégré et un haut parleur, vous pourrez communiquer avec votre animal.



MultiChat :

Avoir plusieurs chats sur l'application et pouvoir connecter plusieurs BillySmart à votre application.

Scanner la gamelle :

Mise en place d'un QrCode sur la gamelle pour faciliter l'inscription .

Pucer votre animal :

Nous voulons mettre un lecteur de puce pour que l'animal (si il est pucé) soit connecté directement à la gamelle grâce à sa puce.

Connexion à une api :

Grâce à cela, vous aurez accès à un régime préconisé par des vétérinaires en fonction de la marque et des spécificités de l'alimentation choisie.

Régimes prédéfinis :

Établis avec des vétérinaires, des régimes prédéfinis prenant en compte la race, le poids actuel et le poids souhaité.

Fontaine à eau :

Du tout en un ? Cela sera possible avec l'ajout d'une réserve d'eau et d'une fontaine qui permettra d'éviter la stagnation de l'eau.

Conseils quotidiens :

Par la suite vous recevrez si vous le souhaitez des conseils quotidiens pour votre animal prononcés par nos vétérinaires.

3. Spécifications techniques

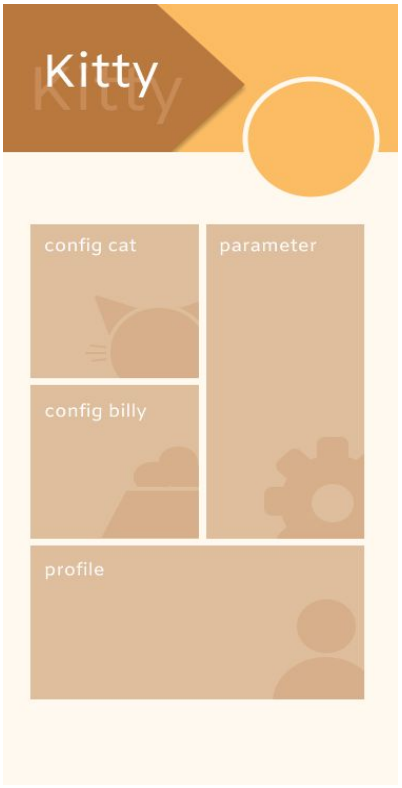
A. L'application mobile

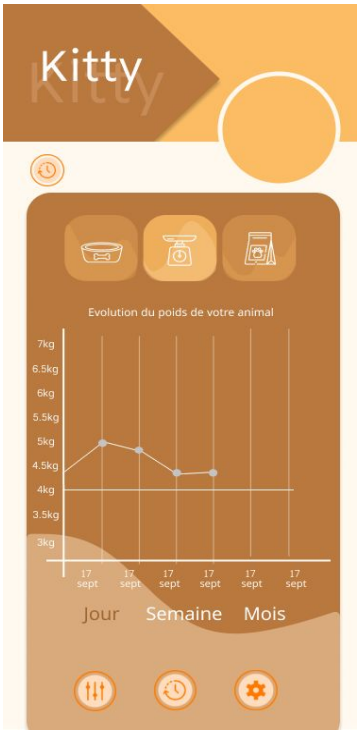
1. L'UX/UI

Nous avons développé plusieurs styles pour l'application, nous sommes partis sur le background avec la vague, pour un aspect plus convivial.

Nous avons aussi testé plusieurs couleurs

Les recherches de l'UX/UI était très importantes pour la partie paramètres, c'est le coeur du fonctionnement de notre application.





2. Les technologies utilisées

Nous avons choisi de développer l'application avec React Native, afin de pouvoir adapter facilement le visuel pour le mobile, mais aussi pour en faire une réelle application mobile et utiliser des éléments comme la notification, la navigation mobile, etc.

3. Le fonctionnement

Actuellement en PoC (Proud of Concept), l'application mobile ne possède pas encore de système de login. Néanmoins, elle récupère ses données via notre api hébergée à distance et est capable de mettre à jour des informations comme le nom de l'animal.

B. Le serveur

1. Les technologies utilisées et fonctionnement

La connexion entre l'application mobile et la gamelle connectée passe directement par une API. Celle-ci permet de faire transiter des données. Nous avons décidé d'utiliser l'hébergeur "**Heroku**" afin de mettre l'API en ligne.

En terme de langage de développement, Node.JS à été utilisé, plus précisément la libraire ExpressJS pour sa capacité à être modifié dans sa structure et sa rapidité de déploiement / développement. De plus il possède une bonne communauté ce qui permet de rapidement trouver réponse à ses problèmes.

Pour ce qui est de la base de données, Heroku permet d'héberger une base de donnée en ligne. le moteur de BDD s'avère être en PostgreSQL.

Le seul défaut de la **version gratuite** d'**Heroku** est sa limitation sur le nombre de requête par minutes autorisé. Soit 10. Le développement à donc été axé sur une utilisation de la base de données maximal avec un nombre de requêtes minimal.

2. API

URL : https://calm-ravine-68746.herokuapp.com

GET : */utilisateur?id=<id>*

Permet de récupérer toutes les informations de l'utilisateur passé en paramètres.

POST : */utilisateur*

{ mail, nom, prénom, photo, pwd }

Permet de créer un utilisateur.

PUT : */utilisateur?id=<id>*

{ mail, nom, prénom, photo, pwd }

Permet de modifier un utilisateur.

GET : */gamelle?id_utilisateur=<id_utilisateur>*

Permet de récupérer le paramètres de la gamelle.

POST : */gamelle*

{ id_utilisateur, reserve, rfid, actif }

Permet de créer une gamelle et de l'attribuer à un utilisateur.

PUT : */gamelle?id_utilisateur=<id_gamelle>*

2{ rfid, actif }

Permet de modifier une gamelle.

DELETE : */gamelle?id_gamelle=<id_gamelle>*

Permet de supprimer une gamelle et son animal.

GET : */animal?id_gamelle=<id_gamelle>*

Permet de récupérer les paramètres d'un animal (un par gamelle).

POST : */animal*

{ nom, race, age, photo, obpoid, typeregime, id_gamelle }

Permet de créer un animal et de l'attribuer à une gamelle.

PUT : */animal?id_animal=<id_animal>*

{ nom, race, age, photo, objpoid, typeregime, id_gamelle }

Permet de modifier un animal.

DELETE : */animal?id_animal=<id_animal>*

Permet de supprimer un animal.

GET : */poids?id_animal=<id_animal>*

Permet de récupérer les différents poids d'un animal

POST : */poids*

{ kg, date, id_animal }

Permet d'ajouter un poids à la liste.

GET : */stats?id_gamelle=<id_gamelle>*

Permet de récupérer les stats d'une gamelle

POST : */stats*

{ date, poidsAvant, poidsApres, poidsConsomme, heureDebut, heureFin, id_gamelle }

Permet de créer une ligne de stats.

GET : */regime?id_gamelle=<id_gamelle>*

Permet de récupérer la fréquence de distributions de croquettes.

POST : */regime*

{ dose, frequence, id_gamelle }

Permet de créer un régime.

PUT : */regime?id_gamelle=<id_gamelle>*

{ dose, frequence}

Permet de modifier un régime.

PUT : */connection*

{ mail, pwd }

Permet de récupérer l'id_utilisateur si l'utilisateur éxiste.

GET : */home?id_user=<id_user>*

Permet de récupérer les infos nécessaire à l'affichage de la page d'accueil.

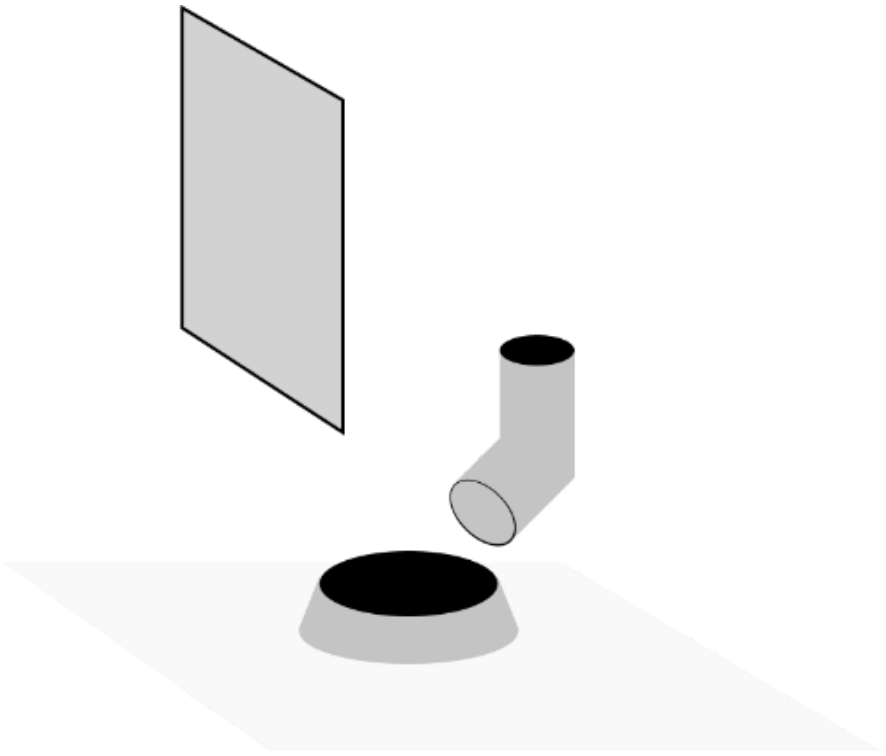
GET : */historique?id_gamelle=<id_gamelle>*

Permet de récupérer toutes les infos composant l'historique d'une gamelle.

C. La gamelle

1. Le prototype

Le prototype de la gamelle connecté



2. Les moteurs

Utilisation de deux moteurs qui permettent de gérer l'ouverture et la fermeture de la gamelle pour l'animal et du réservoir à croquettes. Ils sont gérés via une interface python pour chaque servomoteur. Les interfaces sont des classes et fonctionnent par état.

Pour le servomoteur de la trappe, un script est lancé dès le démarrage du linux. Pour celui du réservoir, un script de requête se lance à 1h du matin via Cron pour actualiser les configurations. Un autre Cron lance l'interface lorsque l'on atteint l'heure de délivrance d'une dose.

3. Les sonars

Le sonar permet de détecter la présence de l'animal, afin d'éviter que la trappe se referme sur lui pendant qu'il mange. Il s'active une fois la trappe ouverte et mesure la variation de distance.

4. Lecteur RFID

La carte permet d'identifier l'animal qui se présente à la gamelle afin d'éviter que d'autres animaux ne mangent. Il tourne en permanence jusqu'à ce qu'une puce autorisée soit reconnue, et se relance une fois la trappe fermée.