Document Technique - Charlie the Creature

Table des Matières

---

1. Introduction

Charlie the Creature est une application web interactive full-stack conçue pour démontrer la maîtrise des technologies modernes et offrir une expérience utilisateur immersive.
L'application met en scène un pet virtuel (un petit chat) évoluant dans une chambre animée.
L'utilisateur interagit avec le pet via des boutons qui déclenchent des actions spécifiques (nourrir, hydrater, jouer, dormir), lesquelles entraînent une augmentation des jauges correspondant aux besoins du pet.

---

2. Architecture du Projet

Technologies Utilisées

- **Frontend :** Vue.js 3, Vite, Axios, CSS (pour transitions et animations)
- **Backend :** Node.js, Express, MongoDB, Mongoose
- **Hébergement :**
  - *Frontend* déployé sur **Vercel**
  - *API* déployée sur **Render**
  - Base de données hébergée sur **MongoDB Atlas**

Schéma de Communication

[Frontend (Vue.js)] │ ▼ [API (Node.js + Express)] ←→ [Base de données (MongoDB)]

L'API REST sert d'intermédiaire entre l'interface utilisateur et la base de données, permettant ainsi au pet de maintenir un état dynamique et persistant.

---

3. Backend - API et Base de Données

Le backend gère la persistance des données et la logique applicative. Il expose deux principaux endpoints :

- `GET /pet` : Récupère l'état actuel du pet, notamment sa position et ses jauges (faim, soif, énergie, bonheur), avec une décrémentation progressive dans le temps.
- `POST /pet` : Met à jour l'état du pet suite aux actions de l'utilisateur (ex. : nourrir, boire, jouer, dormir).

Exemple de Serveur Express (`server.js`)

```
const express = require('express');
const mongoose = require('mongoose');
const petRoutes = require('./routes/petRoutes');

const app = express();
app.use(express.json());
app.use('/api/pet', petRoutes);

mongoose.connect(process.env.MONGO_URI, { useNewUrlParser: true,
useUnifiedTopology: true })
  .then(() => console.log('MongoDB connecté'))
  .catch(err => console.error('Erreur de connexion MongoDB:', err));

app.listen(3000, () => console.log('Serveur lancé sur le port 3000'));

Modèle MongoDB (Mongoose) - Pet.js

const mongoose = require('mongoose');

const PetSchema = new mongoose.Schema({
  posX: { type: Number, required: true },
  posY: { type: Number, required: true },
  hunger: { type: Number, default: 100 },
  thirst: { type: Number, default: 100 },
  energy: { type: Number, default: 100 },
  happiness: { type: Number, default: 100 },
});

module.exports = mongoose.model('Pet', PetSchema);
```

4. Frontend - Interface Utilisateur et Composants

Le frontend est développé en **Vue.js 3** avec **Vite** pour assurer une interface réactive et une expérience de développement rapide.

Composant Pet.vue

Ce composant gère l'affichage et l'animation du pet. Ses principales parties sont :

- **.pet-container** : le conteneur qui positionne le pet.
- **.control-point** : un petit point de contrôle (utilisé pour le débogage ou la détection de hitbox), que vous pouvez masquer via CSS si nécessaire.

Exemple de Code de Pet.vue

```
<template>
  <div class="pet-container" :style="petStyle">
    <!-- Le point de contrôle indique la position centrale ou la hitbox -->
    <div class="control-point" :style="controlStyle"></div>
  </div>
</template>

<script>
import { defineComponent, ref, computed, onMounted } from "vue";
import { usePetMovement } from "../composables/usePetMovement";
import { usePetAnimation } from "../composables/usePetAnimation";
import { usePetConfig } from "../composables/usePetConfig";

export default defineComponent({
  name: "Pet",
  setup() {
    const movement = usePetMovement();
    const config = usePetConfig();
    const animation = usePetAnimation(movement);
    const { animateSprite, executeAction } = animation;
    const { scaleIdle, scaleMoving } = config;
    const { currentActionScale } = animation;

    const flipSpriteHorizontal = ref(false);
    const flipSpriteVertical = ref(false);

    const currentScale = computed(() => {
      if (movement.isPerformingAction.value) return currentActionScale.value;
      if (movement.isMoving.value) return scaleMoving.value;
      return scaleIdle.value;
    });

    const petStyle = computed(() => {
      let transformValue = `scale(${currentScale.value})`;
      if (flipSpriteHorizontal.value) transformValue += " scaleX(-1)";
      if (flipSpriteVertical.value) transformValue += " scaleY(-1)";
      return {
```

```
        position: "absolute",
        left: `${movement.petX.value}px`,
        top: `${movement.petY.value}px`,
        width: "64px",
        height: "64px",
        backgroundImage: "url('/assets/cats/cat_black.png')",
        backgroundPosition: `-${movement.spriteX.value}px
-${movement.spriteY.value}px`,
        backgroundRepeat: "no-repeat",
        transform: transformValue,
        transition: `left ${movement.moveDurationSeconds.value}s ease, top
${movement.moveDurationSeconds.value}s ease`,
      };
    });

    const controlStyle = movement.controlStyle;

    onMounted(() => {
      movement.getHitboxCoordinates();
      animateSprite();
      movement.movePetRandomly();
    });

    return {
      petStyle,
      controlStyle,
      executeAction,
      flipSpriteHorizontal,
      flipSpriteVertical,
    };
  },
});
</script>

<style scoped>
.pet-container {
  position: absolute;
  transform: scale(1);
}
</style>
```

Composant de Contrôle - Les Boutons

Le composant **UIControls.vue** (non présenté entièrement ici) regroupe les boutons d'interaction qui permettent à l'utilisateur d'interagir avec le pet. Chaque bouton déclenche une action spécifique :

- **Nourrir** : augmente la jauge de faim.
- **Hydrater** : augmente la jauge de soif.
- **Jouer** : augmente la jauge de bonheur.
- **Dormir** : augmente la jauge d'énergie.

Lorsqu'un bouton est cliqué, une fonction associée est exécutée pour :

- Augmenter la jauge correspondante.
- Effectuer un appel API afin de mettre à jour l'état du pet dans la base de données.
- Afficher l'évolution des jauges en temps réel dans le composant **Gauges.vue**.

Illustrations

Pour faciliter la compréhension de l'interface et du fonctionnement de l'application, le PDF comprendra trois images :

1. **Room (Chambre Animée)**

Illustration de la chambre dans laquelle évolue le pet.

2. **Boutons (Interface de Contrôle)**

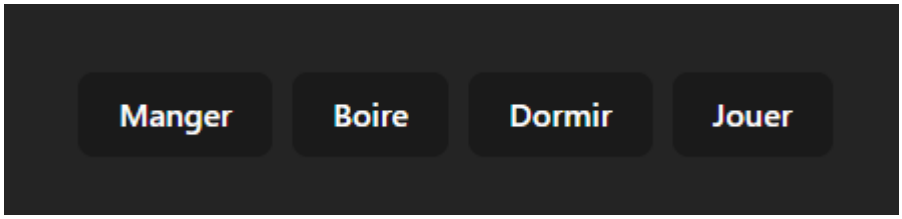
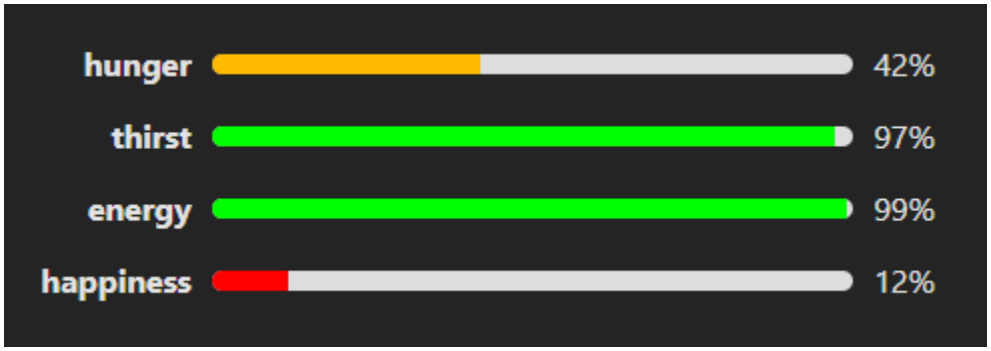


Illustration des boutons d'interaction. Chaque bouton (nourrir, hydrater, jouer, dormir) déclenche une action qui modifie la jauge correspondante.

3. **Gauges (Indicateurs de Besoins)**



Les jauges affichent en temps réel l'état des besoins du pet. Lorsqu'une action est effectuée, la jauge correspondante augmente pour refléter l'amélioration de l'état du pet.

Conclusion

Ce document technique détaille l'architecture, le design et la mise en œuvre de **Charlie the Creature**. Le projet démontre l'intégration efficace de technologies modernes pour créer une application web full-stack interactive. L'API REST connecte l'interface **Vue.js** au backend **Node.js** et à **MongoDB**, assurant ainsi une gestion dynamique et persistante des données du pet. Les interactions via les boutons influent directement sur les jauges, offrant une expérience utilisateur riche et interactive.

Document Technique généré par Charles Decheneux – 03 Juin 2025