Rapport Final du Projet

Chenil ChenYl-Tech

Réalisé par : Shahd, Kenza, Hadil

11/05/2025-Informatique II,MEF1

Sommaire

I-Présentation des Objectifs :

Ce document constitue le rapport final pour le projet de développement d'une application de gestion destinée au chenil "ChenYl-Tech". La mission principale, telle que définie dans la description initiale, était de gérer les pensionnaires du chenil, qui prend en charge divers types d'animaux proposés à l'adoption. Il était demandé de concevoir une application en langage C pour une gestion efficace du chenil, en assurant la persistance des informations dans des fichiers.

Le programme devait proposer une interface textuelle pour l'ajout, la recherche, la modification et la suppression des fiches animales, ainsi que pour la gestion d'autres aspects opérationnels du chenil. Ce travail a été conduit par notre équipe composée de Shahd, Kenza et Hadil. Ce rapport expose en détail les fonctionnalités qui ont été implémentées, la manière dont l'équipe s'est organisée, l'environnement de travail adopté, les défis techniques et organisationnels qui ont été surmontés, et enfin, évalue le projet par rapport aux critères généraux spécifiés, en se concentrant sur la fonctionnalité, la robustesse, la modularité et la qualité générale du code produit. Le sujet officiel portait donc sur la gestion du chenil ChenYl-Tech via une application développée en C, fonctionnant en mode console et assurant la persistance des données sur fichier.

II- Description de l'Application Développée :

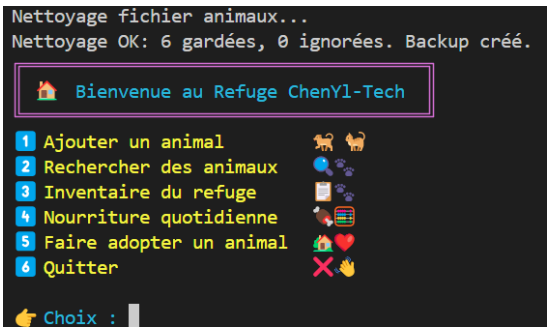
L'application "ChenYl-Tech" qui a été développée met à disposition une interface en ligne de commande permettant aux utilisateurs d'interagir avec les données relatives aux animaux du chenil. En évaluant les fonctionnalités implémentées au regard de l'exigence stipulant que le programme devait correspondre à la description et implémenter toutes les fonctionnalités listées, nous constatons que plusieurs opérations clés sont présentes :

-On a ajouter une fonction ajouterAnimal permettant d'enregistrer les détails d'un nouvel arrivant et de lui attribuer un identifiant unique.

-La recherche d'une fiche animale est également une fonctionnalité implémentée, répondant ainsi à l'exigence "rechercher" grâce à des options de recherche par identifiant, par nom (sans tenir compte de la casse) ou par catégorie d'âge.

-La suppression d'une fiche animale, correspondant à l'exigence "supprimer", est gérée via la fonction d'adoption qui retire l'animal de la liste active du refuge.

-D'autres aspects de la gestion du chenil sont aussi pris en compte : un inventaire complet peut être affiché, listant tous les animaux présents et triés par espèce, accompagné de statistiques globales ; de plus, l'application peut calculer les besoins alimentaires quotidiens estimés pour l'ensemble des pensionnaires.



Cependant, il est important de noter une différence par rapport aux exigences initiales : la fonctionnalité permettant de modifier les informations d'un animal déjà enregistré, bien que mentionnée dans la description du projet, n'a finalement pas été implémentée dans la version finale du code. Ainsi, pour corriger une information erronée, il faudrait actuellement supprimer l'enregistrement de l'animal puis le recréer avec les données correctes. Enfin, la persistance des données est assurée par leur stockage et leur lecture depuis un fichier texte, complétée par un mécanisme de sauvegarde et de nettoyage destiné à améliorer la robustesse du système.

En résumé, bien que l'application soit fonctionnelle pour les opérations essentielles d'ajout, de recherche et d'adoption, elle ne prend pas en compte l'intégralité du périmètre fonctionnel initialement demandé en raison de l'absence de la fonction de modification.

III-Méthodologie de Travail :

Pour mener à bien ce projet, l'équipe composée de Shahd, Kenza et Hadil a adopté une approche structurée, en tenant compte des bonnes pratiques de développement et des critères généraux définis pour le projet. La répartition des tâches a été pensée pour tirer parti des compétences de chacune : Shahd s'est concentrée les données, la persistance de ceux-ci via les fichiers, ainsi que sur la robustesse du système de sauvegarde et de nettoyage des données ; Kenza a pris en charge l'interface utilisateur, incluant le menu principal et l'affichage des informations, la gestion des entrées clavier et l'implémentation de la fonctionnalité de recherche ; enfin, Hadil s'est chargée de la logique, notamment les processus d'adoption, le calcul des besoins alimentaires, l'affichage de l'inventaire et la génération des identifiants uniques pour les animaux.

Concernant l'environnement de travail et les outils utilisés, l'équipe a suivi les recommandations pour favoriser la collaboration et l'efficacité. L'utilisation d'un système de contrôle de version comme Git était fortement conseillée et constituait même le format de livraison exigé pour le projet final. Cet outil a été essentiel pour éviter la perte de travail, faciliter la collaboration entre les membres de l'équipe en permettant de fusionner les contributions de manière organisée, et pour partager facilement l'avancement du projet. De plus, il était recommandé de mettre en place un environnement de travail en équipe s'appuyant sur divers outils collaboratifs. Ainsi, pour la communication et la coordination, on a pu utiliser des plateformes de messagerie instantanée comme Discord, et potentiellement un outil de gestion de tâches tel que Trello pour visualiser et suivre l'avancement des différentes fonctionnalités assignées à chaque membre. (On le fournira à la fin).

IV. Analyse des Défis Rencontrés et des Solutions Apportées :

Plusieurs critères de qualité du code et de conception étaient définis. Le code est effectivement réparti en plusieurs fichiers sources et en-têtes organisés dans une structure de répertoires distincts pour les sources, les en-têtes et les données, ce qui améliore grandement la lisibilité du projet. Le critère demandant que le code soit généreusement commenté a aussi été pris en compte ; des commentaires explicatifs accompagnent de nombreuses fonctions pour montrer leurs rôle. Conformément au critère de la langue, tous les éléments du code, qu'il s'agisse des noms de variables, des noms de fonctions ou des commentaires, sont rédigés de manière cohérente en Français, un choix permis par les consignes. Enfin, un critère particulièrement important concernait la stabilité et la gestion des erreurs : l'application ne devait jamais s'interrompre de manière intempestive et toutes les erreurs devaient être correctement gérées, la stabilité étant jugée préférable à une application complète mais instable. Pour répondre à cette exigence, on a mis en place plusieurs mécanismes visant à renforcer la robustesse de l'application :

-Le module de nettoyage des données gère les erreurs d'ouverture de fichier et effectue des sauvegardes pour prévenir la perte d'informations.

-Des boucles de validation ont été intégrées pour contrôler certaines saisies de l'utilisateur, évitant ainsi des plantages dus à des entrées invalides.

-Des techniques spécifiques comme l'utilisation de `fgets` et d'une fonction pour vider le buffer d'entrée ont été employées pour prévenir les erreurs courantes liées à la fonction `scanf`.

```
while (fgets(ligne, sizeof(ligne), f_in)) {
    int idCourant;
    if (sscanf(ligne, "%d;", &idCourant) == 1) {
        if (idCourant == id_a_adopter) {
            trouve_final = 1;
            sscanf(ligne, "%*d;%49[^;];", nomTrouveFinal);
```

Néanmoins, il faut reconnaître qu'assurer une absence totale de crash en langage C est une tâche difficile qui nécessiterait une gestion de toutes les erreurs potentielles, comme les échecs d'allocation mémoire ou les retours d'erreur de fonctions système. Même si l'application se montre stable lors d'une utilisation normale et que des efforts aient été faits en ce sens, garantir un code parfait avec le critère d'absence totale d'interruption intempestive reste difficile. La priorité donnée à la stabilité, même au détriment d'une fonctionnalité comme la modification, semble cependant avoir été globalement respectée. Le flux de travail adopté a suivi les étapes classiques :

-Conception initiale, développement modulaire avec utilisation de Git, restructuration de l'arborescence, intégration continue manuelle via des points réguliers et Git, tests unitaires et d'intégration, tests réguliers sur des sites tels que gdb online, revues de code informelles, et enfin finalisation incluant la documentation et la préparation du rapport.

Au cours du développement, on a dû surmonter plusieurs défis techniques et organisationnels :

-Un premier point de discussion a concerné le choix du format de fichier pour la sauvegarde des données, opposant les avantages et inconvénients d'un fichier binaire à ceux d'un fichier textuel.

Finalement, on a opté pour un fichier texte utilisant le point-virgule comme délimiteur, privilégiant la facilité de lecture et de débogage pendant la phase de développement.

```
81551;JDN;Chat;2014;12.00;sympa
```

-Un autre problème majeur est survenu du fait que le projet avait démarré sans une structure de répertoires bien définie. Cette absence initiale d'arborescence a rapidement rendu la gestion des fichiers et des dépendances compliqué. Pour y remédier, on a décidé en cours de route d'adopter une structure plus standard séparant les sources, les en-têtes et les données. Cette restructuration, bien que bénéfique à long terme pour la clarté du projet, a nécessité un effort de ré arrangement non négligeable, notamment pour mettre à jour tous les chemins d'inclusion dans les fichiers sources et les en-têtes, ce qui a été géré à l'aide du système de contrôle de version.

-Des difficultés classiques liées à la gestion des entrées/sorties en C ont également été rencontrées, en particulier les problèmes de buffer résiduel avec la fonction `scanf`. La solution adoptée a été de standardiser l'utilisation de `fgets` pour lire les lignes complètes, suivie d'une analyse de la chaîne lue, et de créer une fonction utilitaire pour nettoyer explicitement le buffer d'entrée lorsque nécessaire.

```c
void nettoyerBuffer() {
    int c;
    while ((c = getchar()) != '\n' && c != EOF);
}
```

-Pour améliorer l'ergonomie de la recherche par nom, il est apparu nécessaire de rendre cette recherche insensible à la casse. Une fonction de comparaison spécifique a donc été créer pour ignorer les différences entre majuscules et minuscules lors de la recherche de noms.

-Lors de l'implémentation de la fonction d'adoption, le cas où plusieurs animaux pouvaient porter le même nom a posé problème. Pour résoudre cette ambiguïté, une étape supplémentaire a été ajoutée : si plusieurs animaux sont trouvées pour un nom, le système liste les animaux concernés avec leur identifiant unique et demande à l'utilisateur de spécifier l'identifiant exact de l'animal à adopter.

-Enfin, maintenir un style de codage cohérent entre les trois membres de l'équipe a représenté un défi organisationnel mineur, relevé par la définition de quelques règles de base en début de projet et par des revues de code informelles facilitées par le partage du code via Git.

V-Bilan Final :

En conclusion, le projet "ChenYl-Tech", qu’on a réaliser en équipe a abouti à la création d'une application console en langage C qui répond de manière fonctionnelle aux principales exigences de gestion d'un chenil, incluant l'ajout, la recherche, la suppression (via l'adoption), la consultation de l'inventaire et le calcul des besoins alimentaires. Le projet respecte la plupart des critères généraux définis : il présente une structure claire, utilise une langue cohérente (le Français) dans tout le code, inclut des commentaires explicatifs, et démontre des efforts significatifs pour assurer la stabilité de l'application et la robustesse de la gestion des données stockées dans un

fichier texte. L'utilisation attendue de Git pour la gestion de version et comme format de livraison final est également conforme aux exigences. Le principal écart par rapport au cahier des charges initial reste l'absence de la fonctionnalité permettant de modifier directement les informations d'une fiche animale existante. Bien que l'application se comporte de manière stable pour les fonctionnalités présentes, la garantie d'une robustesse totale face à tous les types d'erreurs potentielles est un objectif assez difficile. Finalement, on a réussi à livré un projet solide qui témoigne d'une bonne compréhension et application du langage C et des pratiques de travail collaboratif, tout en reconnaissant les compromis effectués et les pistes d'amélioration possibles.