|  |
| --- |
| **TPI**  **Gestion des activités d’un apiculteur**  **Kevin Avdylaj – CID4B ETML** |

Chef de projet : M. Gaël Sonney

Experts : Nicolas Borboën et Pascal Benzonana

Lieu : ETML Vennes, Av. de Valmont 30, 1014 Lausanne

Date : Du jeudi 02 mai au lundi 03 juin 2024

Durée : 88 heures

# Analyse préliminaire

## Introduction

Ce rapport contient la réalisation du début à la fin de mon travail pratique individuel (TPI) effectué à l’ETML.

Le sujet de TPI est un site web qui servira à la gestion de ruches et rucher pour un apiculteur.

Le choix du sujet du TPI n’a peu d’importance et j’ai laissé celui-ci à mon chef de projet, ce qui va nous intéresser c’est la conception d’un site web full-stack

## Objectifs

Le but est de fournir une application web permettant la journalisation des activités pour un apiculteur.

L’application doit pouvoir :

* Gérer l’authentification d’un apiculteur
* Fournir les opérations CRUD sur un rucher, une ruche et les activités qui y sont lier
* Consulter la liste de toutes les activités par années
* Consulter la liste des ruchers et ruche d’un apiculteur

L’application doit également avoir une base de données conçue selon ce qui a été vue lors des modules ICT (104, 105, 1153).

Le code source doit être lisible et respecte les conventions de nommage standards pour le langage de programmation utilisé.

## Planification initiale

La planification initiale est fournie en annexe

# Analyse / Conception

## Concept

Lors de la saison des abeilles, un apiculteur doit régulièrement réaliser des inspections et relever les détails propres à chaque ruche. Il doit aussi exécuter des travaux ou des activités spécifiques.

Cette application est destinée à un apiculteur qui s’occupe de plusieurs ruches et réalise les travaux nécessaires à la bonne conduite de son rucher. Un rucher est composé de plusieurs ruches. Il possède un numéro de rucher, un nom et une localisation.

Une ruche possède un numéro, une description, une couleur, l’année de naissance de la reine associée à une couleur. Un apiculteur peut posséder plusieurs ruchers.

Les activités possèdent une catégorie, une description, une durée et une date. Les catégories d’activités sont inspection, mise des hausse, extraction, traitement et nourrissement. Une activité peut être réalisée sur une ruche ou un rucher (toutes les ruches du rucher).

## Méthode de projet

Pour ce projet je vais utiliser la méthode des 6 pas

* ***INFORMER***

Ici il va falloir s’informer sur le projet, prendre connaissance des objectifs, des outils à utiliser, etc.

Pour cette étape j’ai pris connaissance du cahier des charges et eu une discussion avec le premier expert sur le déroulement du TPI. J’ai aussi obtenu des clarifications sur le cahier des charges après avoir fait par de mes questionnements à mon chef de projet.

* ***PLANIFIER***

La phase de planification consiste simplement en la réalisation de ma planification initiale

* ***DÉCIDER***

Pour la partie « décider » je dois choisir la façon dont laquelle je vais réaliser se projet. Les technologies utilisées pour ce projet ont déjà été décidée au préalable lors du P\_APPRO 1 et 2. Il ne manque plus qu’à établir le modèle de base de données, la maquette du site ainsi que la stratégie de test. Une fois tout ça fait, on peut passer à la phase de réalisation

* ***RÉALISER***

C’est ici qu’on commence à coder ! Il faut implémenter le backend (BDD, API, CRUD) et le frontend (Intégration de la maquette)

* ***CONTRÔLER***

Pour le contrôle, je vais effectuer les testes prévus sur l’application, relire et finaliser le rapport

* ***ÉVALUER***

L’évaluation consiste à la rédaction de la conclusion de ce rapport. Conclusion qui contient tous les bilans du projet, le problème rencontrer, l’état finale de l’application, etc.

## Technologies du projet

### Typescript

Typescript est le langage de programmation utilisé dans ce projet. Il permet de réaliser du code en frontend ainsi qu’en backend, me permettant d’utiliser qu’un seul langage pour tout le projet. Typescript apporte des éléments supplémentaires à javascript, notamment les types. Cela me permet de typer mes variables et de débugger plus simplement

### Node JS

Node et un environnement d’exécution pour javascript qui permet de faire du code javascript en dehors du navigateur. Il sera utilisé pour la réalisation du backend.

### Express JS

Express est un Framework javascript qui aide à la réalisation d’API en fournissant les éléments de base pour leurs création (route, middlewares, etc.).

### MySQL

MySQL est le SGBD que je vais utiliser dans ce projet, il permet gérer des base de données relationnel, ce qui est nécessaire pour la BDD de ce projet.

### Prisma (ORM)

Prisma est un ORM qui supporte le Typescript. Celui-ci va me permettre de communiquer avec la base de données en ayant les types des données directement traduit en Typescript et ainsi me donné plusieurs avantages comme l’autocomplétions dans mon IDE, requêtes simplifiée, migrations, etc.

### Vue JS

Vue JS et le framework frontend que j’ai choisi, il va me simplifier l’intégration du site en me permettant de créer des composants qui contiennent leur propres template (HTML), style (CSS) et script (Typescript), facilitant la création d’interface réactive

### Figma

Figma est un éditeur graphique, permettant la réalisation de maquette pour site web principalement. C’est avec lui que je vais designer mon site

### Draw.io

Draw.io est un site qui permet de créer plusieurs type diagrammes. Je l’ai utilisé pour schématisé ma base de données (MCD, MLD, MPD)

### VS code

VS code est l’IDE que j’ai utilisé pour ce projet

## Fonctionnalités

### Authentification

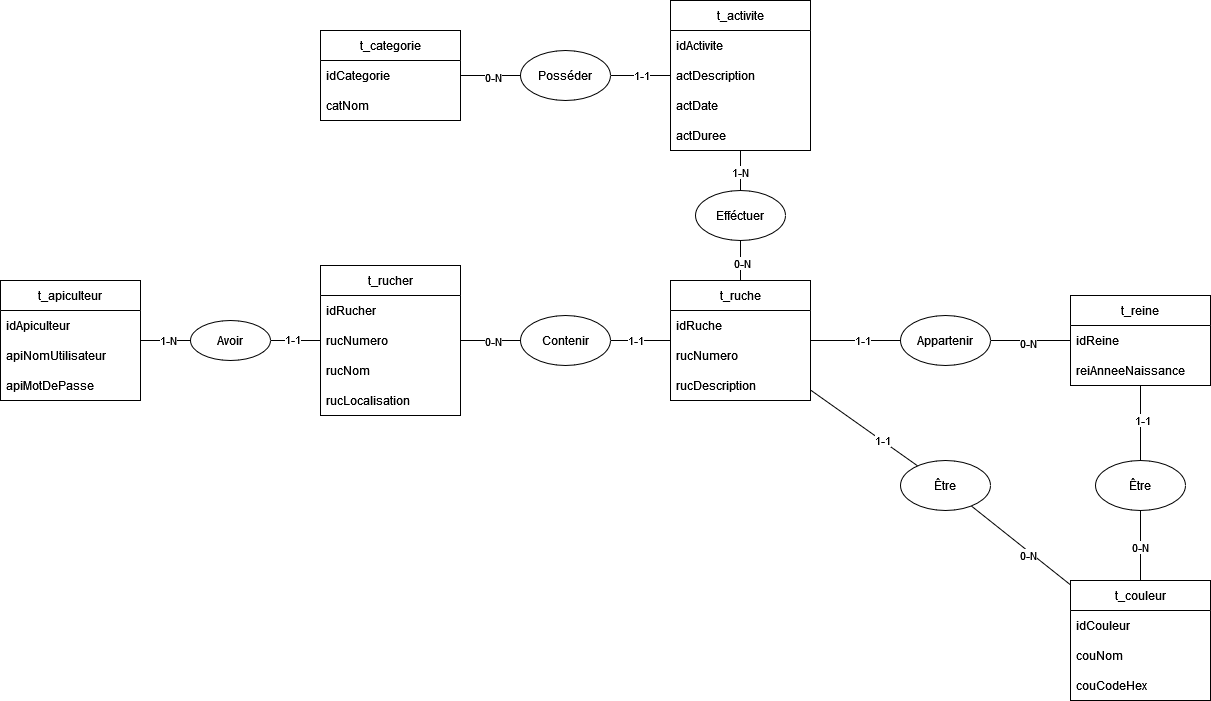
L’authentification s’effectue avec un nom d’utilisateur et un mot de passe, une fois authentifié l’utilisateur possède tous les droits sur l’application. Un utilisateur non-authentifié n’a accès à aucune fonctionnalité.

## Maquette

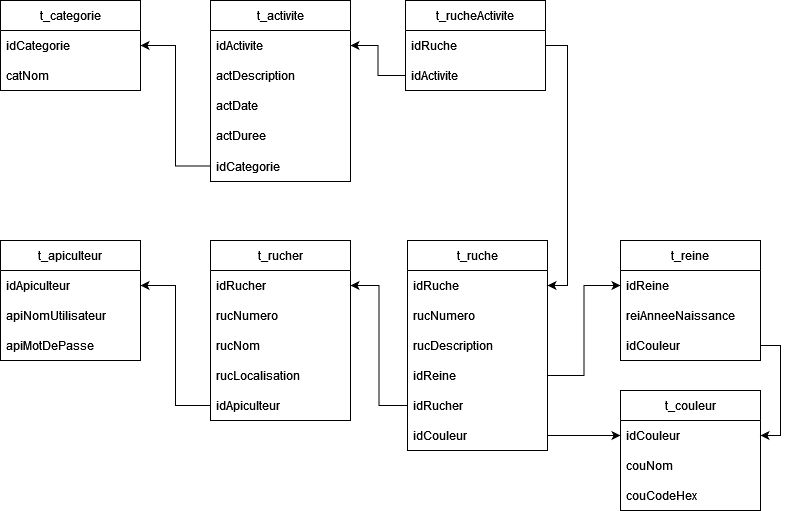
La maquette du site est fournie en annexe

## Base de données

### MCD



### MLD



### MPD

## Stratégie de test

Dans les points techniques évalué dans le TPI (point A14 à A20) il y a :

* L’apiculteur peut se loguer dans l’application et afficher ses ruchers et ruches.
* Les opérations CRUD sur un rucher et une ruche.
* Les opérations CRUD sur une activité.
* L’utilisateur peut afficher la liste des activités pour une année spécifique.
* Les activités concernant un rucher ou une ruche sont affichées dans les détails du rucher ou de la ruche

Je vais donc concentrer mes tests sur tous ces points.

En premier lieu, je vais tester l’API en testant chaque Endpoint et en vérifiant que l’on obtient le résultat attendu. Le teste de tous les Endpoint de l’API va me permettre de déterminer la validité des CRUD et de l’authentification.

Il va également falloir tester l’interface, pour cela je vais décrire les étapes que l’utilisateur doit effectuer et décrire le résultat attendu, c’est-à-dire, décrire les informations qu’il devrait voir à l’écran

# Réalisation

## Tests

### API

Pour tester l’api, je vais définir l’Endpoint testé, la forme que doit avoir la requête et le résultat attendu.

**Tests**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test N° | Endpoint | Header (Requête) | Body (Requête) | Résultat attendu |

**Résultat**

|  |  |  |
| --- | --- | --- |
| Test N° | Date | Résultat du test |

### Interface

Pour tester l’interface, je vais définir les actions que l’utilisateur doits effectuer et les informations qui devrait être affichées une fois les actions réalisées

**Tests**

|  |  |  |
| --- | --- | --- |
| Test N° | Actions | Résultat attendu |

**Résultats**

|  |  |  |
| --- | --- | --- |
| Test N° | Date | Résultat du test |

# Conclusion

## État finale de l’application

## Objectifs atteints

## Objectifs non-atteints

## Problèmes rencontrés

## Amélioration possible

## Bilan de la planification

## Bilan personnel

# Annexe

## Sources – Bibliographie

## Glossaire

## Annexes physiques

## Annexes numériques

# Résume