😎 Plan de projet

Meneur	Équipe projet DIGICHEESE
Approbateur	Correcteur / formateur
Contributeurs	Équipe projet DIGICHEESE
Intervenants informés	Formation Diginamic – encadrement pédagogique
Objectif	Livrer une API FastAPI (Swagger `/docs`) + une base MySQL + tests Pytest, et piloter le tout via un backlog + board Agile avec un sprint fictif.
Date d'échéance	
Résultats clés	• API FastAPI opérationnelle + Swagger • DB MySQL : tables CRUD Admin obligatoires • Tests Pytest : nominal + erreur par ressource Admin (+ traçabilité SquashTM) • Dépôt Git structuré (branches, docs, scripts SQL) • Backlog complet + board à jour + sprint fictif + lien partagé
État	NON DÉMARRÉ / EN COURS / TERMINÉ

🤔 Énoncé du problème

Le système historique de gestion des cadeaux fidélité (Access + VBA, ~20 ans) présente des limites majeures :

- instabilité
- maintenance difficile
- interface obsolète
- incompatibilité avec les environnements modernes

Le besoin est de moderniser et sécuriser la gestion via :

- une API exposant des endpoints REST
- une base MySQL centralisée
- une démarche projet traçable (backlog, sprint fictif, tests, documentation)

🎯 Champ

Requis :	• **API FastAPI** avec endpoints REST + Swagger `/docs`
	• **Base MySQL** avec tables nécessaires au **CRUD Admin obligatoire** :
	  ◦ communes
	  ◦ objets
	  ◦ conditionnements
	  ◦ poids
	  ◦ poids-vignettes
	  ◦ utilisateurs
	  ◦ adresses
	• **Tests automatisés (Pytest)** :
	  ◦ minimum **1 test nominal + 1 test erreur** par ressource admin
	  ◦ tests **passants** + tests **non passants** (traçabilité SquashTM)
	• **Dépôt Git propre** :
	  ◦ branches `master`, `dev`, `test`, `prod` + branche par développeur
	  ◦ `requirements.txt`
	  ◦ scripts SQL
	  ◦ documentation
	• **Gestion de projet (rendu TP)** :
	  ◦ backlog complet (user stories + tâches + estimations)
	  ◦ board Agile à jour + sprint fictif (tickets déplacés / assignés / estimés)
	  ◦ lien vers l'outil (Jira ou GitHub Projects) + invitation correcteur si demandé

Souhaitable :	• Utiliser **Jira** si l'objectif est une gestion "pro" (stories, points, burndown) *(sinon GitHub Projects si priorité à la cohérence dépôt code)*

	• Structurer le backlog en **Épics** (SETUP / AUTH / DATABASE / TESTS / DOC…) • Compléter le macro-périmètre au-delà du CRUD Admin (colis / stocks) si le temps le permet
Hors périmètre :	• Front-end complet (IHM/SPA) et expérience utilisateur finalisée • Industrialisation “production” (hébergement, monitoring, sécurité avancée) • Couverture de test exhaustive (au-delà du minimum requis par ressource admin)

🗓️ Chronologie

Jour	Objectif
J1	Setup repo + branches + environnement + modélisation DB
J2–J3	Implémentation CRUD Admin + API + Swagger
J3–J4	Tests Pytest + traçabilité SquashTM
J4	Documentation + scripts SQL + nettoyage repo
J5	Finalisation board + sprint fictif + rétrospective

🚩 Étapes importantes et échéances

Étape importante	Propriétaire	Échéance	État

Initialisation dépôt (structure, branches, requirements, scripts)	À renseigner	J1	À renseigner
Modélisation DB + création tables CRUD Admin	À renseigner	J1	À renseigner
CRUD Admin : communes / objets / conditionnements / poids / poids-vignettes	À renseigner	J2–J3	À renseigner
CRUD Admin : utilisateurs / adresses	À renseigner	J2–J3	À renseigner
Tests Pytest (nominal + erreur par ressource admin)	À renseigner	J3–J4	À renseigner
Documentation + conformité livrables	À renseigner	J4	À renseigner
Backlog complet + board + sprint fictif (tickets déplacés)	À renseigner	J5	À renseigner
Rétrospective / bilan 5 jours Python	À renseigner	J5	À renseigner

👥 Gouvernance projet & RACI

1 Équipe & contributeurs

Contributeur	Rôle principal (projet)	Remarques
Stanislas DELANNOY	Référent technique / Data / Docker	Socle API, DB, reproductibilité
Imen KHAMMASSI	Développeuse Backend	Lots CRUD (Admin / OP-colis selon répartition)
Thi Thu Hien NGUYEN	Développeuse Backend	Lots CRUD (Admin / OP-colis selon répartition)
Xavier DEGUERCY	Référent fonctionnel & documentation	Cadrage, périmètre, backlog, doc Confluence

> Remarque : les rôles PO/SM ont été assurés de manière collégiale pendant le TP. Pour une lecture “entreprise”, Xavier porte la responsabilité fonctionnelle (PO) et Stanislas le pilotage technique (lead dev).

.2 Rôles projet (cadre)

- **PO (Product Owner)** : porte le besoin fonctionnel, arbitre le périmètre (In/Out), priorise le backlog (MoSCoW), valide l’adéquation au cahier des charges.
- **Scrum Master / Responsable projet** : sécurise le cadre Agile, anime le suivi, gère les blocages, garantit la discipline Jira.
- **Lead Dev / Référent technique** : fixe les conventions techniques (architecture, erreurs, conventions API), assure l’homogénéité et la qualité.
- **Développeurs** : implémentent les fonctionnalités (CRUD), tests, corrections.
- **Référent qualité** (si désigné) : garantit la cohérence DoD, la couverture minimale de tests, la non-régression.

---

3 Matrice RACI

Légende :

- **R** = Responsible (réalise)
- **A** = Accountable (porte la responsabilité / valide)
- **C** = Consulted (consulté)

- **I** = Informed (informé)

Activité / livrable	A	R	C	I
Cadrage fonctionnel & périmètre (In/Out)	Xavier	Xavier	Équipe	Tous
Backlog Jira (MoSCoW + critères d'acceptation)	Xavier	Xavier	Équipe	Tous
Conventions projet (M0) : structure, erreurs, endpoints	Stanislas	Stanislas	Équipe	Tous
Socle API (M1) : FastAPI + MySQL + .env	Stanislas	Stanislas	Équipe	Tous
Swagger & conventions endpoints (M2)	Stanislas	Stanislas	Équipe	Tous
CRUD Admin (M3 à M8) : implémentation lots	Stanislas	Imen / Hien / Xavier (selon lots)	Stanislas	Tous
Tests automatisés (pytest) : nominal + erreurs	Stanislas	Imen / Hien / Xavier	Stanislas	Tous
Documentation Confluence (FONC/TECH)	Xavier	Xavier	Équipe	Tous
Reproductibilité (Docker / compose)	Stanislas	Stanislas	Équipe	Tous
Qualité code (ruff / lint / conventions)	Stanislas	Stanislas	Équipe	Tous

---

4 Règles de fonctionnement (traçabilité)

- Chaque ticket Jira doit contenir :
  - un objectif clair,
  - des critères d'acceptation,
  - une priorité MoSCoW,
  - un lien vers la documentation Confluence impactée si applicable.
- Chaque livraison doit être traçable :
  - branche nommée selon convention,
  - commits associés,
  - relecture croisée minimale avant merge (si possible).

🎲 Gestion des risques (registre)

1 Méthode

- Échelle **Probabilité (P)** : 1 (faible) / 2 (moyenne) / 3 (élevée)
- Échelle **Impact (I)** : 1 (faible) / 2 (moyen) / 3 (élevé)
- **Criticité = P × I**
- Revue des risques à chaque fin d'itération (Sprint Review / Rétrospective).

---

2 Registre des risques

ID	Risque	P	I	Criticité	Indicateurs	Prévention / mitigation	Owner
R1	Traçabilité Jira ↔ code insuffisante	2	3	6	tickets sans lien, status non cohérent	DoD : lien ticket→branche/commit + mise à jour Jira systématique	Équipe

R2	Écarts de conventions entre CRUD	2	3	6	patterns différents, noms variables	conventions M0 + gabarits + revue croisée rapide	Stanislas
R3	Conflits Git / merges en parallèle	2	2	4	conflits fréquents, rebase long	branches par dev + PR petites + merge régulier	Équipe
R4	Dérive de périmètre (IHM / cyber / options)	2	3	6	nouvelles demandes non planifiées	In/Out clair + MoSCoW strict + timebox	Xavier
R5	Environnement non reproductible (DB, .env, deps)	2	3	6	“ça marche chez moi”	`.env.example` + docker-compose + guide d’installation	Stanislas
R6	Dette technique / qualité code hétérogène	2	2	4	style non uniforme, warnings	ruff + conventions + corrections par lot	Stanislas
R7	Tests ajoutés trop tard ⇒ debug coûteux	2	3	6	bugs découverts en fin de sprint	smoke tests early + min “nominal + 404” par CRUD	Équipe

R8	Contraintes délai TP	3	3	9	backlog trop ambitieux	MUST d'abord + report explicite des SHOULD/COULD/WOULD	Xavier

---

3 Plan d'action risques (sprint suivant)

1. Standardiser le DoD : "ticket complet + lien code + test minimal".
2. Démarrer chaque CRUD par : routes + smoke test + validation Swagger.
3. Exécuter ruff + pytest avant merge (local au minimum, CI si possible).
4. Geler le périmètre : toute demande OUT est taggée WOULD et planifiée hors sprint.

🗃️ Liens associés

- **Dépôt GitHub** : GitHub - xavier-deguercy/TP_digicheese
- **Board Jira / GitHub Projects** : https://xavespace.atlassian.net/?continue=https%3A%2F%2Fxavespace.atlassian.net%2Fwelcome%2Fsoftware%3FprojectId%3D10043&atlOrigin=eyJpIjoiMWZlZDYxNzllNTczNDBkMGI2NGQzY2Q5N2FjZDg1MjgiLCJwIjoiamlyYS1zb2Z0d2FyZSJ9
- **Swagger** : `<http://.../docs` > (URL à renseigner)
- **SquashTM (campagne de tests)** : À renseigner (URL / référence)
- **Documentation** :
  - README : TP_digicheese/README.md at main · xavier-deguercy/TP_digicheese
  - Doc Projet : TP_digicheese/DOC at main · xavier-deguercy/TP_digicheese
  - Doc technique : TP_digicheese/DOC/TECH at main · xavier-deguercy/TP_digicheese
  - Doc fonctionnelle : TP_digicheese/DOC/FONC at main · xavier-deguercy/TP_digicheese