TP Fullstack “CENDRES ET VAPEUR”

Support pour le TP Fullstack “Cendres et vapeur” de Février 2026
Formateur : Dufrène Valérian
“Déclaration d’activité enregistrée sous le numéro 32800232680 auprès du préfet de région HAUTS-DE-FRANCE”

Directives de survie logicielle pour les ingénieurs de la zone franche

Écoutez bien, techniciens.

Le Grand Conseil ne vous accorde que **8 cycles de 24 heures** pour forger votre interface avant le jour du jugement final (Présentation).

Vos ressources sont limitées, votre code doit être d'acier.

Voici votre feuille de route détaillée.

Répartition des escouades

La Guilde a divisé vos forces en 4 secteurs d'intervention basés sur vos codes couleurs.

Chaque secteur est une unité autonome :

- ➔ 🔵 **Secteur Cobalt (11 membres) :** Sous la direction de **PINAULT Camille**. Vous formez l'épine dorsale du réseau et du flux de données.
- ➔ 🔴 **Secteur Rouille (8 membres) :** Composé de **Consolo Alexis**. Vous gérez la mécanique lourde, les moteurs de calcul et la persistance.
- ➔ 🟣 **Secteur Pourpre (11 membres) :** L'élite composée de **Boudegna Philippe**. Vous assurez la sécurité des transmissions et l'intégrité des communications cryptées.

Mise en oeuvre technique

La Forge Backend (Python & SQL)

- ➔ **Architecture de données :** Concevoir un schéma SQL normalisé.
  Vous devez gérer l'intégrité référentielle pour les transactions de troc, les logs d'accès et les sessions de double authentification.
  - ◆ **Hiérarchie des rôles :** Implémentation stricte de 4 niveaux d'accès : **Invité** (lecture seule), **Utilisateur** (achat/vote), **Éditeur** (gestion catalogue), **Administrateur** (contrôle total).
- ➔ **L'API de transmission :** Déployer une API REST avec Python (FastAPI ou Flask recommandés pour la vélocité).
  Elle doit inclure une gestion stricte des erreurs et des codes HTTP (404, 401, 500) formatés en JSON.
- ➔ **La sécurité Oauth/2FA :** Implémenter un module de vérification d'identité.
  Pas d'accès au Dashboard sans preuve de citoyenneté (code envoyé par email ou validation via service tiers).
  Techniciens, pour l'expédition de vos codes de sécurité et missives de contact, ne perdez pas de cycles précieux à tenter de restaurer les anciens relais SMTP dévastés.
  La guilde vous autorise l'usage de **services de télégraphie simulée** tels que **Mailtrap** ou **SendGrid** (via leurs paliers gratuits).
  Ces outils feront office de tunnels sécurisés pour tester vos envois sans que la pression ne fasse exploser vos serveurs locaux.
  Tout code qui resterait bloqué dans les tuyaux par manque de configuration sera considéré comme une défaillance de votre unité.

L'Interface de Commande (Angular ou React & CSS 3)

- ➔ **Esthétique post-apocalyptique :** Votre CSS doit être une œuvre d'art. Utilisez des bordures en cuivre brossé, des animations de rouages lors des chargements, et une typographie rappelant les anciennes machines à écrire.

- ➔ **Accessibilité (A11Y) :** La zone est sombre.
  Votre interface doit être lisible par tous : contrastes élevés, navigation au clavier impérative, et balises aria-label sur chaque composant interactif.

- ➔ **Dashboard d'administration :** Espace centralisé pour gérer les utilisateurs, les produits et les commandes.
  L'architecture doit être pensée pour rester fluide même si le nombre de modules augmente.
- ➔ **Vitrine dynamique :** Affichage des produits avec injection de données en temps réel depuis l'API.
- ➔ **Système de vote (Pression Populaire) :** Les produits doivent être triés dynamiquement selon les "Likes" des utilisateurs.
- ➔ **Tunnel d'achat :** Panier fonctionnel, validation de commande, simulation d'un module de paiement et **édition automatisée de factures** (format PDF ou HTML imprimable).
- ➔ **Gestion des remises :** Possibilité d'appliquer des codes de réduction ou des remises sur le prix final.

Modules de survie

En plus du socle de base (Panier, Planning, Contact), vous devez intégrer ces mécanismes :

- ➔ **Le moniteur de toxicité :** Développez un module de visualisation de données en temps réel.
  Il doit simuler l'état de l'air de la colonie (via une génération de données aléatoires côté Python).
  Si le taux de soufre dépasse un seuil, l'interface doit passer en "Alerte Rouge" (modification dynamique du DOM via CSS/JS).
- ➔ **La bourse du cuivre :** Les prix ne sont pas fixes.
  Implémentez un algorithme de fluctuation des prix.
  Chaque fois qu'un produit est consulté ou acheté, son prix varie en fonction de l'offre et de la demande simulée.
  L'affichage doit montrer des indicateurs de hausse ou de baisse (flèches de laiton).
- ➔ **Le chronomètre de la colonie :** Le temps est une ressource aussi précieuse que l'eau purifiée.
  Votre interface doit impérativement intégrer un **calendrier de rotation**, pilier de l'organisation de la zone franche :
  - ◆ **Affichage des éphémérides :** Développement d'un calendrier complet (vue mensuelle ou hebdomadaire) affichant les événements critiques de la colonie (ravitaillements, maintenances de la chaudière, couvre-feu).
  - ◆ **Notes de quart :** Chaque utilisateur accrédité doit pouvoir consigner des notes journalières ou **semi-journalières** (distinction stricte entre le quart du matin et le quart du soir).
  - ◆ **Persistance Mécanique :** Ces entrées ne doivent pas s'évaporer à la fermeture de la valve (le navigateur).
    Elles doivent être enregistrées via l'API dans vos tables SQL, liées à

l'identifiant du citoyen.

- **Interface synoptique :** Le design doit permettre de distinguer d'un seul coup d'œil les jours de haute pression (surchargés en événements) des jours de calme.

➔ **Le télégraphe de l'ombre :** Un chat interne réservé aux administrateurs et éditeurs.
Les messages doivent s'afficher instantanément.
Utilisation obligatoire des **WebSockets** (ou Long Polling en secours) pour un échange instantané.

➔ **Bureau de poste :** Formulaire de contact dont le contenu est formaté et envoyé par email via un service SMTP ou API dédiée.

➔ **Journal des survivants :** Un flux de logs public affichant les dernières actions de la colonie de manière immersive.

Contraintes techniques et discipline de travail

- ➔ **Design :** Création préalable d'une maquette et d'un prototype basique (Figma ou autre) avant tout codage.
- ➔ **Stack :**
  - ◆ Backend **Python** avec **API REST JSON**
  - ◆ Frontend **Angular ou React**
  - ◆ Base de données **SQL**.
- ➔ **Standard Web :**
  - ◆ CSS 3
  - ◆ HTML sémantique
  - ◆ point d'honneur sur **l'accessibilité** et **l'UX**.
- ➔ **Versioning :** Chaque commit doit être une brique solide.
  Suivez le schéma Major.minor.patch (0.0.1 > hiérarchie des montées de version).
  Pas de message de commit vague.
- ➔ **Kanban :** Un développeur sans tâche dans son Kanban est un ouvrier inutile.
  Le flux doit être constant.
- ➔ **Identité Individuelle :** Préparez vos preuves.
  Lors de la présentation individuelle, vous devrez isoler vos lignes de code et justifier vos choix techniques devant l'examinateur.

Conseils de planning

Techniciens, votre temps est compté, la “haute” ne vous laisse qu’un court délai pour accomplir cette mission de la meilleure des manières.

Voici les conseils que je peux vous donner pour répartir les tâches :

- ➔ **J1 :** Maquettage, schéma SQL et initialisation des dépôts.
- ➔ **J2-J4 :** Développement du noyau (auth 2FA, rôles, API de base).
- ➔ **J5-J6 :** Implémentation du e-commerce, du chat webSockets et du planning.
- ➔ **J7-8 :** Intégration des modules de survie, polissage CSS et tests finaux.
- ➔ **J10 :** Soutenance devant le Conseil (groupe + individuel).