Rapport d'Organisation — Projet JavaScript ESGI *(2025)*

---

1. Contexte & Objectifs

Le partiel « JavaScript Front-End débutant » visait à consommer une API locale (Mockoon) et à réaliser une suite d'exercices (ex 1 → ex 4) portant sur :

- les appels réseau (`fetch`) ;
- la manipulation du DOM ;
- la mise en forme CSS. L'ensemble devait être versionné sur GitHub et documenté.

---

2. Pile Technique & Logiciels

Domaine	Outil / version	Rôle principal
Contrôle de version	Git 2.44 + GitHub	Historisation, travail collaboratif
IDE	JetBrains WebStorm	Édition JS / HTML / CSS
Serveur mock	Mockoon 9.3.0	Endpoints REST locaux, réponses JSON
Runtime JS	Node.js 20 (LTS)	Scripts utilitaires, tests rapides `fetch`
Navigateurs	Safari & Brave	DevTools, tests multi-moteurs
Terminal	zsh (macOS 13)	Commandes Git / curl
Assistance IA	ChatGPT (o3)	Debug ciblé, refactoring CSS, regex

Dépôt GitHub : https://github.com/sneadz/esgi1-partiel_javascript.git

---

3. Méthodologie & Organisation

1. **Initialisation**\ • Création du dépôt GitHub ; clonage local.\ • Rédaction d'un *README* listant objectifs et todo-list.
2. **Mise en place de l'API**\ • Import de l'environnement Mockoon ; lancement sur `localhost:3004` .\ • Validation des données via curl et DevTools.
3. **Développement incrémental**\ • Implémentation séquentielle des ex 1 → ex 4 (DOM, CSS, events).\ • Commits atomiques avec messages explicites.
4. **Prise de notes continue**\ • Fichier `NOTES.md` mis à jour après chaque micro-tâche.\ • Problèmes & solutions consignés systématiquement.
5. **Tests et itérations**\ • Vérification live dans le navigateur ; rafraîchissement Mockoon.\ • Lint/Prettier exécutés en pré-commit pour garder un code homogène.

---

4. Extrait de Prise de Notes

```
- Création du dépôt GitHub
- Clonage local du répertoire
- Installation de l'environnement Mockoon
- Vérification des données (curl / DevTools)
- ex 1 : récupération du *waste-count*
- ex 2 : affichage des *waste-receipes* dans une UL
- ex 3 : gestion du clic sur les catégories
- ex 4 : style CSS + effet hover
- Problème : chemin « entrées » encodé (`%C3%A9`) → 404
  ▸ Analyse : le navigateur encode systématiquement l'URL ; Mockoon ne décode pas avant routage.
  ▸ Solution : path regex `^/entr(?:ées|%C3%A9es)$` pour accepter les deux formes.
```

---

5. Apports de l'IA

Étape concernée	Question posée	Valeur ajoutée par ChatGPT
Début ex 2	« Comment récupérer uniquement *waste-count* ? »	Snippet `fetch` minimal et clair (ES6 & ES5)
Début ex 3	« Boucle DOM plus propre »	Reco `innerHTML` vs `DocumentFragment`, optimisation reflow
Debug URL	404 sur `/entr%C3%A9es`	Explication encodage URL + proposition regex Express
CSS final	Transition hover	Transition fluide `transform` + `box-shadow`

L'IA a été utilisée comme **assistance ponctuelle** ; chaque suggestion a été relue, testée puis adaptés au contexte du projet.

---

6. Problème Majeur & Résolution

- **Symptôme** : appel à `/entrées` renvoyant 404 ; l'inspecteur réseau montre `/entr%C3%A9es`.
- **Diagnostic** : le navigateur encode l'URL ; Express (Mockoon) ne faisant pas de décodage, la route n'est pas reconnue.
- **Solution** : transformer la route en regex `^/entr(?:ées|%C3%A9es)$` via le switch « Regex » de Mockoon 9.3.\ ➜ Les deux formes sont désormais acceptées **sans changer le code front-end**.

---

7. Bilan & Perspectives

Points forts	Axes d'amélioration
Workflow Git rigoureux (commits atomiques)	Branches thématiques + PR pour revue
Documentation continue (`NOTES.md`)	Étendre la doc (diagrammes de flux, screenshots)
Résolution rapide du bug URL	Mettre en place des tests automatisés
Usage raisonné de l'IA	Passer le CSS en SCSS compilé pour imbrication

---

8. Conclusion

Le projet a été mené de façon itérative et transparente : contrôle de version, prise de notes structurée et outils adaptés ont permis d'atteindre les objectifs sans dérive. L'intégration ponctuelle de l'IA a accéléré le déblocage de points précis tout en laissant la prise de décision finale au développeur. Les pistes d'amélioration identifiées (branching avancé, tests automatisés, pipeline SCSS) serviront de base pour de prochains projets plus ambitieux.