|  |
| --- |
| ExceptionTea |

Table des matières

[1 Analyse préliminaire 5](#_Toc197332455)

[1.1 Introduction 5](#_Toc197332456)

[1.2 Objectifs 5](#_Toc197332457)

[*3.2.1.* Respect du modèle MVC avec Laravel 5](#_Toc197332458)

[*3.2.2.* Qualité et lisibilité du code 5](#_Toc197332459)

[*3.2.3.* Processus d’inscription et d’authentification complet 5](#_Toc197332460)

[*3.2.4.* Sécurité des accès et des données 5](#_Toc197332461)

[*3.2.5.* Validation serveur et retours utilisateurs clairs 5](#_Toc197332462)

[*3.2.6.* Système de gestion des thés 5](#_Toc197332463)

[*3.2.7.* Listes et exportation PDF 5](#_Toc197332464)

[1.3 Planification initiale 6](#_Toc197332465)

[*3.2.8.* Méthodologie et planification 6](#_Toc197332466)

[2 Analyse / Conception 7](#_Toc197332467)

[2.1 Concept 7](#_Toc197332468)

[2.2 Stratégie de test 7](#_Toc197332469)

[*3.2.9.* Données de test 8](#_Toc197332470)

[2.3 Risques techniques 8](#_Toc197332471)

[*3.2.10.* Risques identifiés 8](#_Toc197332472)

[*3.2.11.* Solutions appliquées 8](#_Toc197332473)

[2.4 Planification 8](#_Toc197332474)

[2.5 Dossier de conception 9](#_Toc197332475)

[*3.2.12.* Configuration de la machine virtuelle 9](#_Toc197332476)

[*3.2.13.* Liste des logiciels testés 10](#_Toc197332477)

[3 Réalisation 11](#_Toc197332478)

[3.1 Dossier de réalisation 11](#_Toc197332479)

[3.1.1. Problèmes de versions (Truffle, Ganache, NPM) 11](#_Toc197332480)

[3.1.2. Problèmes de await sur VirtualBox 11](#_Toc197332481)

[3.1.3. Problème de version Solidity 11](#_Toc197332482)

[3.1.4. Problème d’installation de Remix IDE 12](#_Toc197332483)

[3.1.5. Problème de permissions entre root & eleve sur les fichiers systèmes 12](#_Toc197332484)

[3.1.6. Problème de l’explorateur de blockchain (Blockscout / Etherscan + Vue.js) avec Docker 12](#_Toc197332485)

[*3.2.14.* Description physique du projet 13](#_Toc197332486)

[3.2. Description des tests effectués 13](#_Toc197332487)

[*3.2.15.* Test 1 : Analyse des contracts avec Mythril 13](#_Toc197332488)

[*3.2.16.* Test 2 : Exécution des tests Hardhat 14](#_Toc197332489)

[3.3. Erreur restante 15](#_Toc197332490)

[3.4. Liste des documents fournis 15](#_Toc197332491)

[*4.* Conclusions 15](#_Toc197332492)

[4.2. Objectifs et Résultats 15](#_Toc197332493)

[4.3. Suites possibles pour le projet (évolutions & améliorations) 16](#_Toc197332494)

[*4.3.1.* Développement d’une interface graphique dédiée 16](#_Toc197332495)

[*4.3.2.* Création de dApps et de smart contracts plus complexes 16](#_Toc197332496)

[*4.3.3.* Déploiement sur un réseau local (ETML) 16](#_Toc197332497)

[5. Annexes 17](#_Toc197332498)

[5.2. Résumé du rapport du TPI / version succincte de la documentation 17](#_Toc197332499)

[5.3. Sources – Bibliographie 17](#_Toc197332500)

[5.4. Journaux de travail 18](#_Toc197332501)

[5.5. Manuel d'Installation 18](#_Toc197332502)

[5.4.1. Installation de Remix IDE sur Ubuntu 12. 18](#_Toc197332503)

[5.4.1.1. Introduction 18](#_Toc197332504)

[5.4.1.2. Prérequis 18](#_Toc197332505)

[5.4.1.2.1. Mise à jour du système 18](#_Toc197332506)

[Avant d'installer Remix IDE, il est recommandé de mettre à jour votre système. Ouvrez un terminal et exécutez la commande suivante pour mettre à jour les paquets : sudo apt-get update && sudo apt-get upgrade -y 18](#_Toc197332507)

[5.4.1.2.2. Installation de curl 18](#_Toc197332508)

[Si curl n'est pas installé sur votre système, vous pouvez l'installer en utilisant la commande suivante : 18](#_Toc197332509)

[5.4.1.3. Téléchargement et installation de Remix IDE 19](#_Toc197332510)

[5.4.1.4. Explication des commandes 19](#_Toc197332511)

[5.4.2. Résolution des problèmes 19](#_Toc197332512)

[5.4.2.1. Erreur de dépendances manquantes 19](#_Toc197332513)

[5.4.2.2. Vérification de l'installation 19](#_Toc197332514)

[5.4.3. Conclusion 19](#_Toc197332515)

[5.6. Manuel d'Utilisation 19](#_Toc197332516)

[*5.6.1.* Interaction avec les smart contracts en utilisant truffle 19](#_Toc197332517)

[Étape 1 : Déploiement du smart contract 19](#_Toc197332518)

[Étape 2 : Ouverture de la console interactive 19](#_Toc197332519)

[Étape 3 : Interagir avec le smart contract 20](#_Toc197332520)

[*5.6.2.* Utilisation du script de mise en place de l’environnement 21](#_Toc197332521)

[5.7. On voit aussi plus bas que notre contrat s’est bien compilé avec quelques informations comme l’adresse et le Gas utilisé pour cette transaction. Manuel d'Utilisation 22](#_Toc197332522)

[*5.7.1.* Interaction avec les smart contracts en utilisant Remix IDE 22](#_Toc197332523)

[5.8. Archives du projet 22](#_Toc197332524)

# Analyse préliminaire

Dans cette partie du rapport nous verrons le cadre ainsi que le but du projet.

## Introduction

Ce projet s’inscrit dans le cadre du travail pratique individuel (TPI) réalisé en fin de formation d’informaticien à l’ETML. Il répond à la demande d’ExceptionTea SA, une entreprise lausannoise spécialisée dans les thés rares, qui souhaite disposer d’une solution personnalisée pour référencer, organiser et gérer sa collection.

L’application web sera développée avec le framework PHP Laravel, en suivant une architecture MVC rigoureuse. Elle proposera une interface moderne, responsive et intuitive, conçue avec HTML5, CSS3 et Tailwind CSS.

## Objectifs

## Respect du modèle MVC avec Laravel

Mettre en œuvre rigoureusement l’architecture Modèle-Vue-Contrôleur à chaque étape du développement, en séparant clairement la logique métier (modèles), la gestion des requêtes et règles (contrôleurs) et la présentation (vues Blade).

## Qualité et lisibilité du code

Produire un code clair et bien structuré, en respectant les conventions officielles Laravel, ainsi que les standards HTML5 et Tailwind CSS pour le front-end. Les noms de classes, méthodes et fichiers doivent être explicites et cohérents.

## Processus d’inscription et d’authentification complet

Implémenter l’ensemble des fonctionnalités utilisateurs : création de compte, connexion, réinitialisation de mot de passe. L’ensemble doit être utilisable sans bug, avec des redirections et messages d’état appropriés.

## Sécurité des accès et des données

Appliquer les normes de sécurité Laravel : hachage des mots de passe, protection CSRF, gestion des autorisations via middleware et policies, et chiffrement des sessions. Les rôles et permissions doivent garantir que seuls les utilisateurs authentifiés accèdent aux pages protégées.

## Validation serveur et retours utilisateurs clairs

Mettre en place des règles de validation sur tous les formulaires côté serveur. En cas d’erreur, l’utilisateur reçoit un message précis indiquant le champ concerné et la nature du problème.

## Système de gestion des thés

Une interface complète de gestion des produits (CRUD) sera intégrée à l’application. Les utilisateurs pourront créer, modifier, supprimer ou consulter les informations liées aux thés (nom, variété, type, provenance, prix, etc.).

## Listes et exportation PDF

Les utilisateurs auront la possibilité de créer des listes de thés, en fonction de leurs préférences. Une fonctionnalité d’export au format PDF permettra de générer et sauvegarder ces listes de manière pratique.

## Planification initiale

## Méthodologie et planification

Pour ce projet, mon choix de méthodologie s’est tourné vers la méthode des six pas, celle-ci répond particulièrement bien aux exigences d’un TPI sur le développement web. Elle permet aussi une approche structurée avec une phase d’analyse approfondie qui permet de bien cercer les points importants et les objectifs du travail. La phase suivante se concrétise par la planification claire des tâches, suivie d’une conception méthodique des solutions. Le développement lui se fait de manière itérative, ce qui donne l’avantage de la flexibilité du projet en cours de route. Enfin, des phases de contrôle qualité rigoureuses assurent la fiabilité du produit final. Contrairement au modèle en cascade, cette méthode permet une progression plus souple, chaque étape pouvant évoluer indépendamment. Les dernières phases garantissent une évaluation complète avant la livraison, identifiant les points d'amélioration pour les projets futurs.

#### Décomposition en tâches selon les six pas

##### **S’informer**

* + Lecture et compréhension du cahier des charges pour cadrer l’application ExceptionTea.
  + Entretiens avec l’expert ou l’enseignant pour valider les objectifs.
  + Lecture d’informations sur Laravel, MySQL et Tailwind CSS.

##### **Planifier**

* + Découpage du projet en jalons (maquettes, prototype, développement, tests, documentation).
  + Estimation des durées pour chaque phase et chaque tâche.
  + Mise en place du dépôt GitHub et configuration du suivi de version.

##### **Décider**

* + Validation des wireframes basse fidélité pour chaque page clé.
  + Choix des librairies front-end (Tailwind CSS, plugin Vite) et des outils de génération de PDF (Dompdf ou Snappy).
  + Élaboration du MCD, du MLD et du MPD pour modéliser la structure de la base de données.
  + Conception des maquettes des différentes pages.

##### **Réaliser**

* + **Développement back-end** : création des modèles, migrations, contrôleurs et routes.
  + **Développement front-end** : intégration des maquettes en Blade, application des styles Tailwind et interactivité (pop-ups, filtres).
  + Configuration de la base de données MySQL et exécution des migrations Laravel.
  + Rédaction de la documentation technique et du journal de bord.

##### **Contrôler**

* + Exécution des tests unitaires et fonctionnels (vérification des flux CRUD, sécurité, validation des formulaires).
  + Revue de code et corrections éventuelles (refactoring au normes, ajustement des performances).
  + Préparation de la démonstration finale et génération des livrables.

##### **Evaluer**

* + Recueil des retours venant de l’évaluation du chef de projet et des experts.
  + Analyse des écarts entre le cahier des charges et le résultat livré.
  + Identification des améliorations et des bonnes pratiques à retenir pour un futur projet.

A présent voici à quoi ressemble la planification initiale sous forme de graphique :

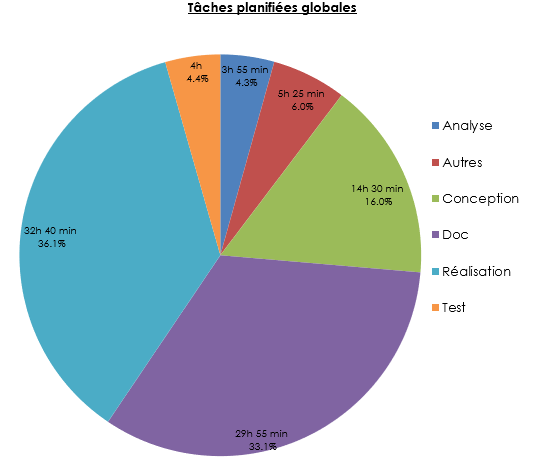


Figure 1 - Graphique des tâches planifiées

On observe tout d’abord qu’avec près de 33 % du temps consacré à la documentation, celle-ci occupe une place majeure dans le projet, soulignant l’importance accordée à la traçabilité et à la qualité du livrable écrit.

L’analyse (4,3 %) et la conception (16 %) totalisent environ 20 %, exactement ce qui était préconisé dans le cahier des charges pour la phase d’étude.

La réalisation atteint 36 %, très proche des 35 % réservés à l’implémentation, ce qui confirme un bon respect de la répartition initiale.

Dans la catégorie ‘’autres’’ nous pourrons retrouver notamment les absences et imprévus potentiels, ces deux choses n’étant pas planifiables ces 6 % représente actuellement une légère marge de sécurité, du temps prévu en plus dans le cas ou une fonctionnalité ne seraient pas terminé ou que la doc devrait être finalisée.

Enfin, les tests représentent 4,4 % du planning : ce temps dédié à la validation des fonctionnalités garantit la fiabilité du logiciel.

# Analyse / Conception

## Maquettes

J’ai choisi de réaliser des maquettes plus haute fidélité plutôt que de simples wireframes, afin de faciliter la phase front-end et d’éviter toute hésitation lors de l’intégration.  
Les voici :   


Figure 2 - Maquette - S'enregistrer

Cette page permet de s’enregistrer la première fois, lorsqu’on n’a pas encore de compte.  
Pour ce faire, il faut entrer son nom, son e-mail et un mot de passe. Puis la validation se fait avec le bouton ‘’S’enregistrer’’.

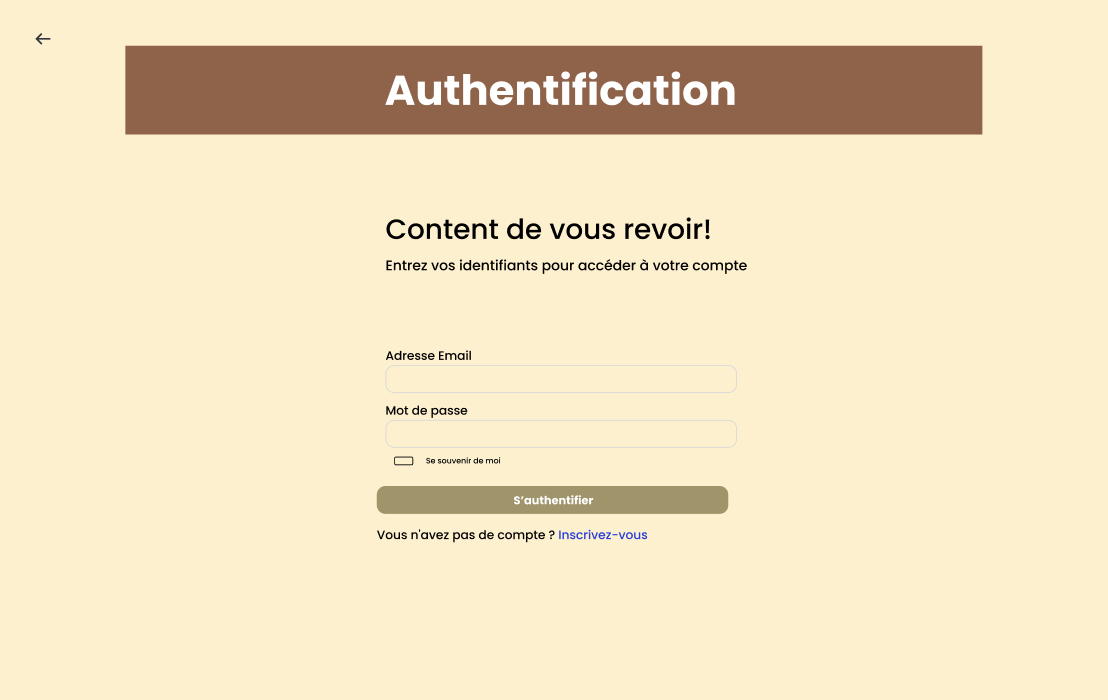
**

Figure 3 - Maquette - S’authentifier

Ici nous avons la page qui permet de se connecter si on a déjà un compte, il suffit de rentrer son adresse mail et son mot de passe dans les champs prévus à cet effet. Puis la validation se fait avec le bouton ‘’S’authentifier’’.

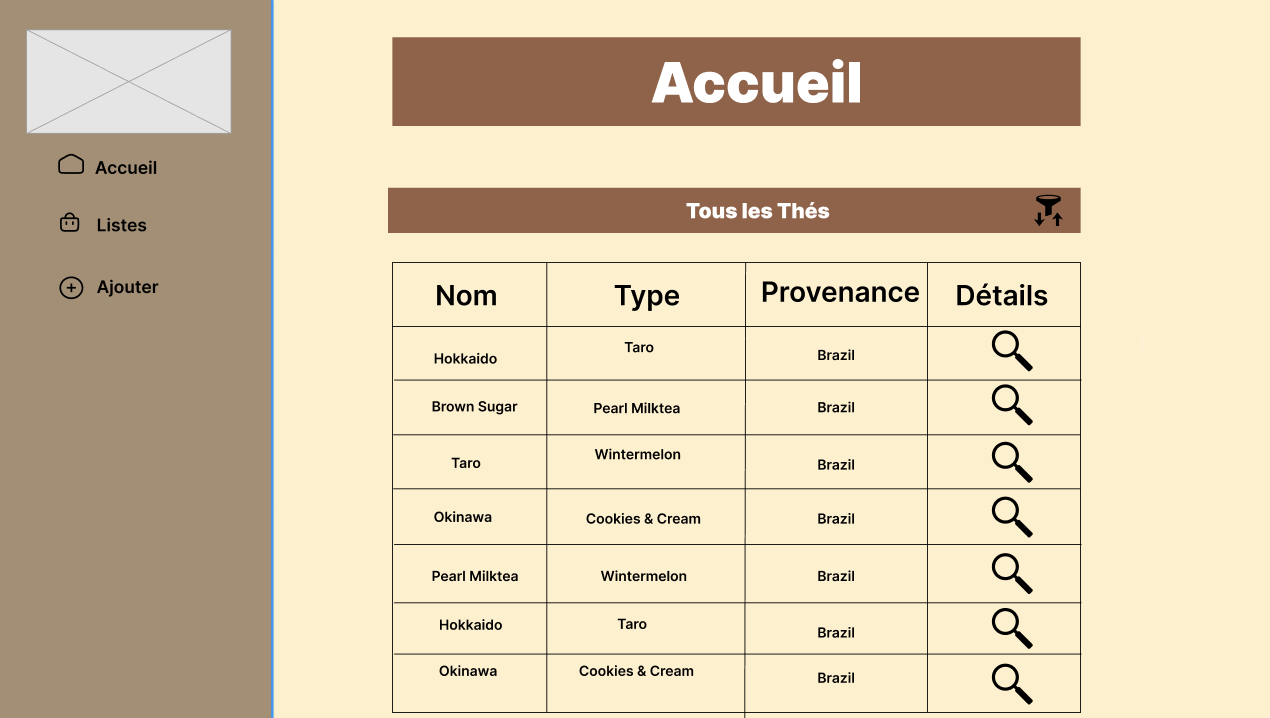


Figure 4 - Maquette - Accueil

Voici la page d’accueil. On y trouve une liste de thés disponibles dans la boutique, avec des options de tri et de filtrage. Il est possible d’accéder à la page de détails d’un thé en cliquant sur l’icône en forme de loupe située à droite de chaque élément. À gauche, des boutons de navigation permettent de parcourir les différentes pages du site.

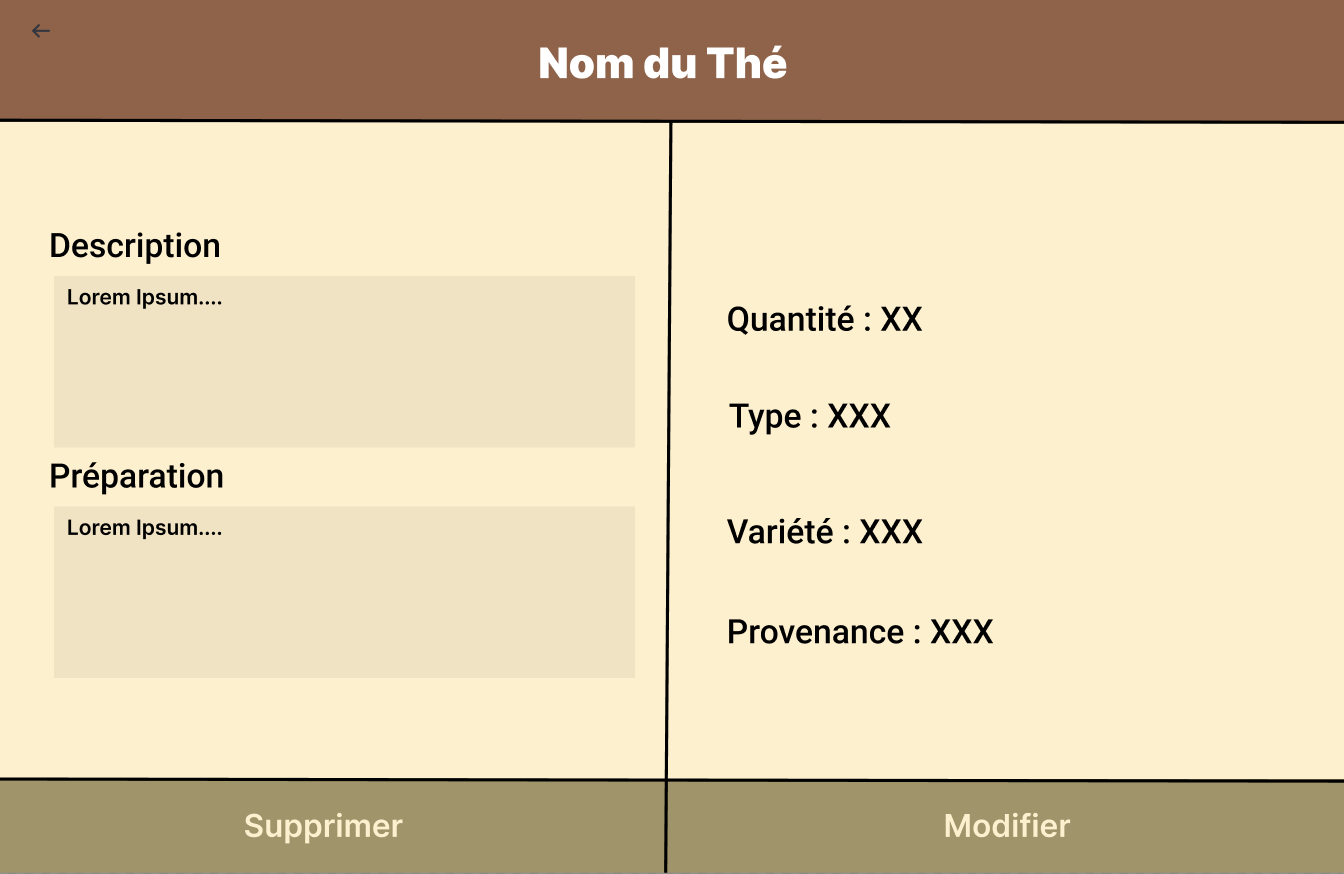
**

Figure 5 - Maquette - Détails du Thé

Ici, la page qui permet d’obtenir plus d’informations sur le thé sélectionné. On peut y consulter des éléments tels que la description, les instructions de préparation, la quantité en stock, la variété et la provenance du thé. Deux boutons situés dans le pied de page permettent de modifier ou de supprimer les informations liées à ce thé.



Figure 6 - Maquette - Ajouter un thé

Sur cette page, nous avons un formulaire permettant d’ajouter un nouveau thé dans la base de données. Nous devons renseigner toutes ces informations importantes dans des champs prévus à cet effet. Les cinq premiers champs sont essentiellement des champs texte. Le champ de la date permettra d'ouvrir un petit calendrier en cliquant sur l’icône à droite, ce qui facilite la sélection d’une date. Les trois dernières informations à renseigner sont la variété, la provenance et le type. Ces trois champs sont sous forme de listes déroulantes, il suffit de sélectionner l’information correcte. Ces listes déroulantes permettent également de supprimer des options grâce à la croix à droite de l’élément, de modifier des éléments avec l’icône de crayon et d’ajouter un élément avec le bouton "Ajouter un élément" en fin de liste.

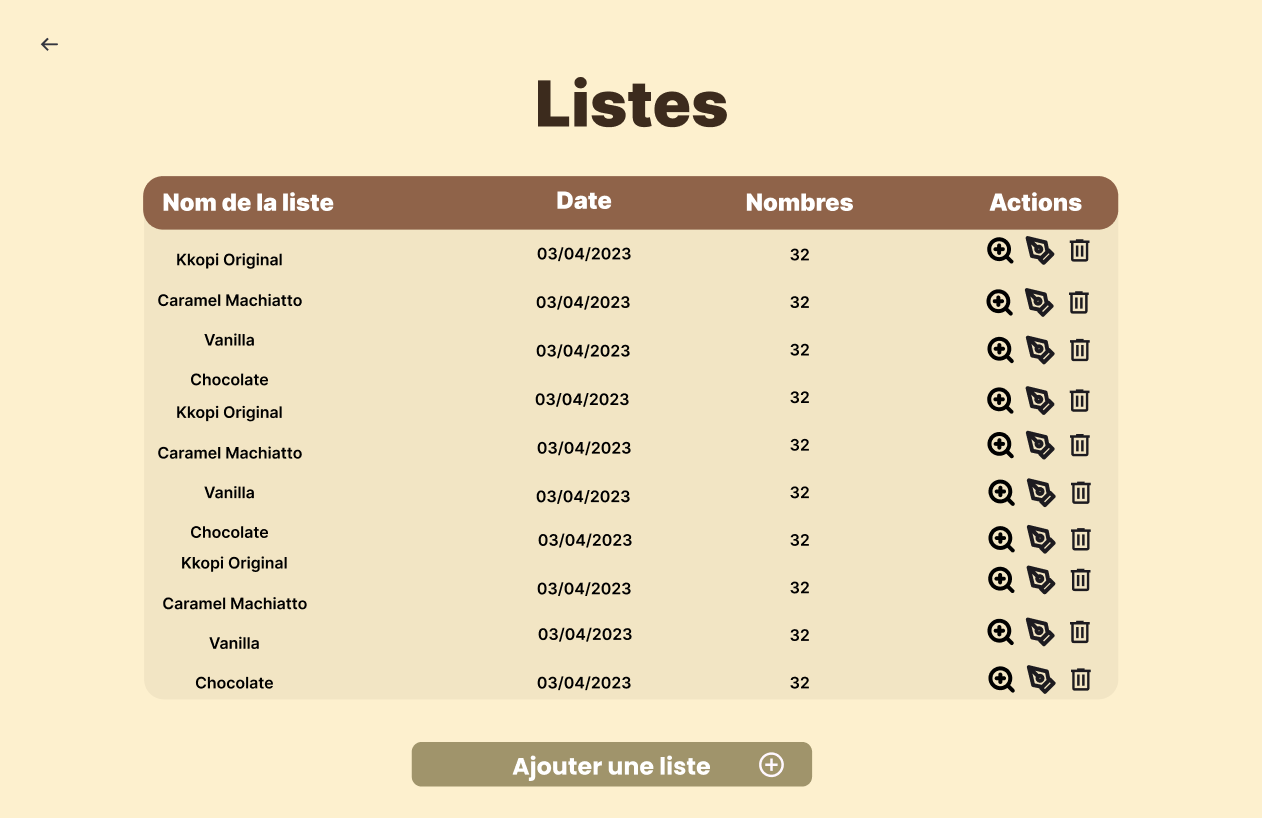


Figure 7 - Maquette – Listes

Ci-dessus, la page contenant toutes les listes de thés disponibles, répertoriées par leur nom, la date de création et le nombre de thés présents dans chaque liste. Dans la colonne "Actions", nous avons trois boutons :

1. La loupe qui affiche les détails de la liste,
2. Le stylo qui modifie les éléments de la liste,
3. La poubelle qui supprime entièrement la liste.

**

Figure 8 - Maquette - Détails d'une liste

Voici la page de détails d’une liste, elle permet à l’utilisateur de voir le contenu de toute la liste ainsi que de l’exporter en PDF avec le bouton vert.

## Modélisation de la base de données

## MCD

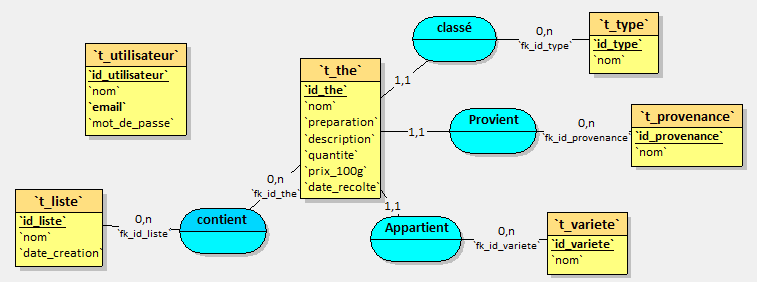


Figure 9 - BDD – MCD

Le MCD présente, de manière visuelle et abstraite, les **entités** clés du domaine, ici Thé, Provenance, Type, Variété et Liste et leurs **relations**. Chaque entité est représentée par un rectangle listant ses principales propriétés (nom, description, etc.), et chaque association par un ovale verbal (par exemple ‘’contient’’ pour relier Liste et Thé, ‘’Provient’’ pour relier Thé et Provenance). Ce schéma garantit que tous les éléments métier et leurs interactions sont correctement identifiés avant toute implémentation technique.

## MLD

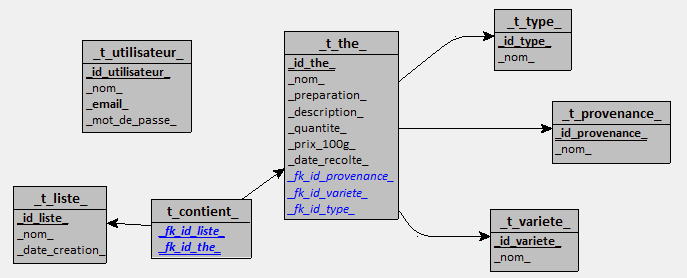
******

Figure 10 - BDD – MLD

Le MLD reprend le MCD pour en faire un schéma relationnel exploitable sous MySQL : chaque entité du modèle conceptuel devient une table préfixée t\_qqch (par exemple t\_the, t\_provenance), dotée de ses colonnes et d’une clé primaire id\_... tandis que les relations sont matérialisées par des clés étrangères fk\_… dans les tables concernées. La relation multiple entre les listes et les thés est, quant à elle, gérée par une table intermédiaire t\_contient dont la clé primaire composite (fk\_id\_liste, fk\_id\_the) garantit l’unicité de chaque association.

## Stratégie de test

La stratégie de test consiste à concevoir au début du projet des tests de fonctionnalités basé sur les objectifs du cahier des charges. Puis de les effectuer dans la partie ‘’test’’ prévue pour. De plus j’ajouterai des tests unitaires

*Tout au long du projet, nous avons mis en place différentes méthodes de test afin de valider nos solutions et garantir leur bon fonctionnement.*

*Dès les premières étapes, nous avons réalisé des tests fonctionnels pour vérifier que nos choix technologiques et nos implémentations répondaient aux attentes. Ces tests portaient notamment sur l’exécution des transactions, la visualisation des blocs et l’interaction avec les smart contracts.*

*Par la suite, pour tester notre infrastructure finale, comprenant l’installation automatisée des outils via notre script ainsi que l’exécution d’une blockchain locale avec Hardhat, nous avons utilisé des tests unitaires intégrés à Hardhat. Ceux-ci sont exécutés automatiquement à la fin du script, avec un retour affiché dans la console et enregistré dans un fichier dédié.*

*Enfin, nous avons renforcé nos vérifications en menant une analyse de sécurité avec Mythril, un outil permettant de détecter d’éventuelles vulnérabilités dans les smart contracts Ethereum. Nous détaillons plus bas son utilisation.*

### Données de test

*Les tests ont été effectués avec des données simulées, pas de données réelles pour des raisons de sécurité.*

## Risques techniques

### Risques identifiés

#### Complexité des smart contracts :

Solidity contient un des risques d'erreurs complexes et difficiles à détecter.

#### Environnement stable :

Il est très complexe d’avoir un environnement stable avec toutes les versions différentes des applications utilisées.

#### Manque de compétences :

La maîtrise de certaines technologies, telles que la blockchain, Hardhat et Solidity, a représenté un défi en raison de leur spécificité. L'absence de cours sur ces sujets à l’ETML a renforcé la difficulté d'adaptation.

#### Immuabilité du code déployé :

Une fois un smart contract déployé sur la blockchain, il ne peut plus être modifié. Cela signifie que toute erreur, faille de sécurité ou mise à jour nécessaire ne peut être corrigée directement. Il faut alors redéployer un nouveau contrat et migrer les données, ce qui peut être complexe et entraîner des coûts supplémentaires en gas sur un réseau réel.

### Solutions appliquées

#### Tests de sécurité priorisés :

Audits des smart contracts avec Mythril.

#### Formation :

Formation avec des articles et des sites web sur les technologies spécifiques du projet.

#### Modularité des contrats :

Réduction de la complexité avec des contrats plus simples et modulaires afin de comprendre vraiment ce qu’on fait et ce qu’il se passe en conséquence.

#### Snapshots :

Mise en place d’un système de snapshot à chaque opération conséquente sur la machine virtuelle.

## Planification

Date de début :

*Révision de la planification initiale du projet :*

* *planning indiquant les dates de début et de fin du projet ainsi que le découpage connu des diverses phases.*
* *partage des tâches en cas de travail à plusieurs.*

*Il s’agit en principe de la planification définitive du projet. Elle peut être ensuite affinée (découpage des tâches). Si les délais doivent être ensuite modifiés, le responsable de projet doit être avisé, et les raisons doivent être expliquées dans l’historique.*

## Dossier de conception

Dans le cadre de ce projet, nous avons testé de nombreux logiciels et outils pour mettre en place un environnement de développement fonctionnel. Certains d'entre eux se sont révélés compatibles et efficaces, tandis que d'autres ont présenté des problèmes d'installation ou de compatibilité entre eux.

Nous avons au départ tenté d'installer une machine virtuelle avec **VirtualBox**, mais celle-ci a rencontré des crashs répétés. Nous avons ensuite téléchargé **VMware Workstation Pro** avec une machine virtuelle sous **Ubuntu**, sans succès. Finalement, nous avons opté pour une machine virtuelle sous **Debian 12.0**, qui a fonctionné correctement.

### Configuration de la machine virtuelle

* **Hôte** : Station de travail de l’école sous Windows 10
* **Logiciel de virtualisation** : **VMware Workstation PRO** (version **17.6.2**)
* **Système d’exploitation invité** : **Debian 12.0 LTS**
* **Utilisateur** : **eleve**
* **Mot de passe** : **etml-**
* **Nombre de processeurs** : **2**
* **Nombre de cœurs** : **8**
* **Mémoire RAM** : **16GB**
* **Contrôleur SCSI** : **LSI LOGIC frea**
* **Type de réseau** : **NAT**
* **Type de disque** : **SCSI dynamiquement alloué (100GB)**

### Liste des logiciels testés

| Outil | Version | Utilité | Fonctionnel (Oui/Non) |
| --- | --- | --- | --- |
| NPM | 10.9.2 | Gestion et installation des packages | Oui |
| Node.js | 22.13.1 | Environnement d'exécution JavaScript | Oui |
| Truffle | 5.11.5 | Framework Ethereum pour dApps | Oui |
| RemixIDE | 0.12.0 | IDE pour smart contracts Ethereum | Oui |
| Ganache | 7.9.1 | Blockchain personnelle pour tester | Oui |
| Goeth | 1.15.3 | Langage de programmation Ethereum | Non |
| Visual Studio Code | 1.97 | IDE pour développement logiciel | Oui |
| Hardhat | 2.22.19 | Framework Ethereum pour développement | Oui |
| JSON-RPC | 2.0 | Protocole de communication pour Ethereum | Oui |
| GitHub (web) | - | Plateforme de gestion de code source | Oui |
| Docker | 28.0.0 | Conteneurisation des applications | Non |
| Fuse | 7.13 | Système de fichiers virtuel pour Ethereum | Non |
| Etherscan API | 10.3.0 | API pour interroger la blockchain | Non |
| Blockscout | 7.0.0 | Explorateur de blockchain Ethereum | Non |
| Curl | 7.88.1 | Outil de ligne de commande pour les requêtes HTTP | Oui |
| VirtualBox | 6.1 | Virtualisation de machines | Non |
| VMware Workstation | 17.6.2 | Virtualisation de machines | Oui |
| Solidity | Plusieurs | Langage de programmation des contrats Ethereum | Oui |
| Web3.js | 1.10.0 | Bibliothèque pour interagir avec la blockchain Ethereum | Oui |

# Réalisation

## Dossier de réalisation

### Problèmes de versions (Truffle, Ganache, NPM)

* + - 1. *Problème*

Lors des installations des dépendances de Truffle, Ganache, NPM, nous avons rencontrés des soucis de comptabilités de certaines versions de ces dépendances, certaines versions récentes de Ganache ne fonctionnaient pas avec des versions plus récentes que Truffle

* + - 1. *Cause*

Truffle et Ganache ont des dépendances qui évoluent avec des versions de Node.js, et certaines versions plus récentes de NPM modifient la gestion de packages, provoquant des soucis de comptabilité.

* + - 1. *Solution*

Nous avons testé plusieurs combinaisons et avons retenu une combinaison stable et compatible :

Node.js : 22.13.1

NPM : 10.9.2

Truffle : 5.11.5

Ganache : 7.9.1 Cela a permis d’assurer une compatibilité avec les smart contracts développés en Solidity 0.8.x.

### Problèmes de await sur VirtualBox

* + - 1. *Problème*

Les commandes asynchrones utilisant await provoquaient des erreurs d’exécution sous VirtualBox (Ubuntu).

* + - 1. *Cause*

VirtualBox gérait mal certains appels réseau ou process asynchrones (surtout dans Ganache)

* + - 1. *Solution*

Migration vers **VMware Workstation Pro 17.6.2**, qui est une version de machine virtuelle beaucoup plus stable. Avec VMware, les commandes await dans les scripts de déploiement de smart contracts fonctionnent correctement.

### Problème de version Solidity

* + - 1. *Problème*

Certains fichiers utilisaient la version 0.5.x de Solidity, d’autres 0.7.x, ce qui provoquait des erreurs de compilation avec Hardhat et/ou Ganache.

* + - 1. *Cause*

Incohérence entre les versions des fichiers et les outils (Hardhat, Truffle, etc.), en particulier dans le hardhat.config.js ou truffle.config.js

* + - 1. *Solution*

Uniformisation de tous les contrats en version **Solidity 0.8.28**, en précisant au début du code « pragma solidity ^0.8.28; » version compatible avec les plugins Hardhat, Ganache.

### Problème d’installation de Remix IDE

* + - 1. *Problème*

Installation de Remix Desktop non fonctionnelle via les méthodes classiques ou les gestionnaires de paquets.

* + - 1. *Cause*

La version packagée pour Debian 12 n’est pas maintenue. Le fichier AppImage ou Snap ne fonctionnait pas correctement

* + - 1. *Solution*

Utilisation d’une commande terminale qui va installer via GitHub pour récupérer la version fonctionnelle :

curl -sLO https://github.com/remix-project-org/remix-desktop-insiders/releases/download/v1.0.8-insiders/remixdesktop\_1.0.8-insiders\_amd64.deb

sudo dpkg -i remixdesktop\_1.0.8-insiders\_amd64.deb

Cette commande est spécifique à une version Debian 12

### Problème de permissions entre root & eleve sur les fichiers systèmes

* + - 1. *Problème*

Le script n’autorisait pas à être lancé par un sudoer root, sauf qu’il y avait des installations de dépendance tel que Hardhat et il y a des modifications dans les fichiers systèmes nécessaires, donc il fallait des permissions root pour pouvoir faire ces modifications

* + - 1. *Cause*

Le script d’installation Hardhat et déploiement de DAPP essayait de tout exécuter avec un seul utilisateur, ce qui posait des conflits de permissions.

* + - 1. *Solution*

Utilisation de sudo uniquement pour les étapes nécessitant des droits root pour modifier les fichiers système

Exécution de Hardhat, déploiement de la DAPP et des tests directement sous l’utilisateur eleve.

### Problème de l’explorateur de blockchain (Blockscout / Etherscan + Vue.js) avec Docker

* + - 1. *Problème*

Impossible de connecter correctement un explorateur comme Blockscout ou etherscan utilisant Vue.js à la blockchain locale

* + - 1. *Cause*

Inconnue

* + - 1. *Solution*

Aucune solution complète trouvée.

*Décrire la réalisation "physique" de votre projet*

* *les répertoires où le logiciel est installé*
* *la liste de tous les fichiers et une rapide description de leur contenu (des noms qui parlent !)*
* *les versions des systèmes d'exploitation et des outils logiciels*
* *la description exacte du matériel*
* *le numéro de version de votre produit !*
* *programmation et scripts: librairies externes, dictionnaire des données, reconstruction du logiciel - cible à partir des sources.*

*NOTE : Evitez d’inclure les listings des sources, à moins que vous ne désiriez en expliquer une partie vous paraissant importante. Dans ce cas n’incluez que cette partie…techniquement les problèmes rencontrés et comment les résoudre*

### Description physique du projet

#### Répertoires où le logiciel est installé :

*/home/eleve/blockchain\_calculator # Répertoire principal du projet*

*│── contracts/*

*│ └── Calculator.sol # Contient les contrats Solidity*

*│*

*│── scripts/*

*│ └── deploy.js # Script de déploiement du contrat*

*│*

*│── tests/*

*│ └── test\_results.txt # Fichier contenant les résultats des tests*

#### Liste des fichiers et description :

* *Calculator.sol : Contrat Solidity avec des fonctions pour l'addition, la soustraction, la multiplication et la division.*
* *deploy.js : Script JavaScript pour déployer le contrat Calculator sur un réseau local via Hardhat.*
* *hardhat.config.js : Configuration de Hardhat, spécifiant le compilateur Solidity et le réseau local.*
* *package.json : Dépendances et scripts définis pour le projet, généré par npm init.*

## Description des tests effectués

### Test 1 : Analyse des contracts avec Mythril

*Conditions : sécurité et aucune erreur de code.*

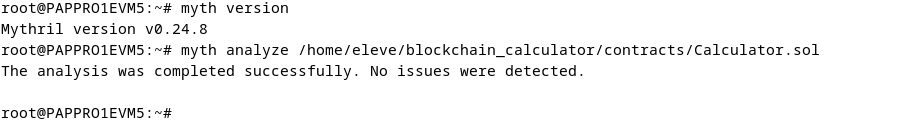
*Test effectué :*

Installation de Mythril :   
pipx install mythril

Vérification de la version :

myth version  
Analyse d’un smart contract

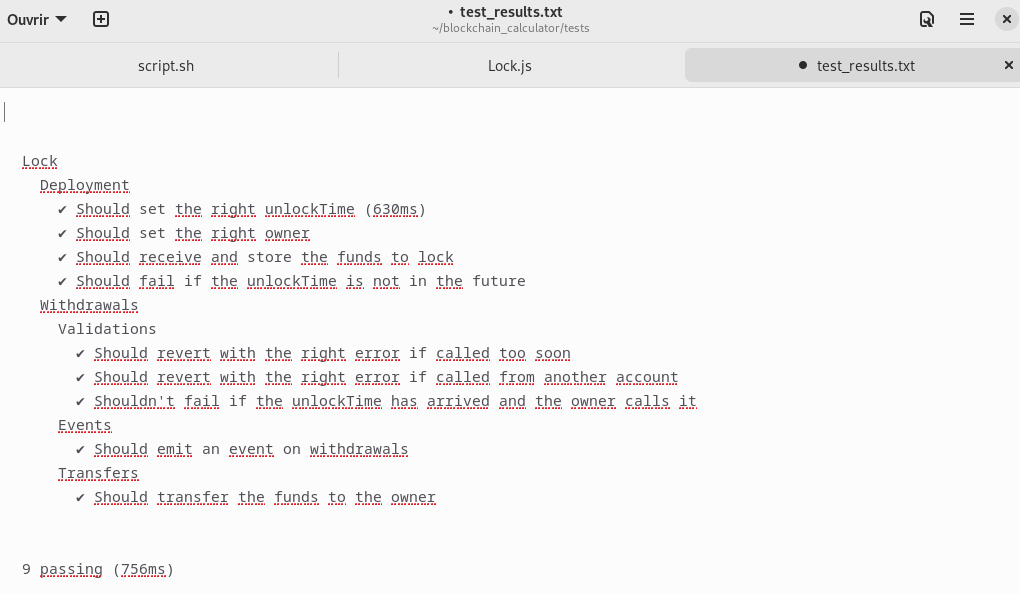
myth analyze /home/eleve/blockchain\_calculator/contracts/Calculator.sol

*Preuve : Mythril nous renvoi un message pour dire qu’il n’y a aucun problème avec le contrat.*  


### Test 2 : Exécution des tests Hardhat

* *Conditions : Les tests sont définis dans le répertoire tests.*
* *Tests effectués : Lancement de npx hardhat test.*
* *Preuve : Les résultats des tests sont enregistrés dans le fichier test\_results.txt.  
  Résultat graphique :*

**

*Résultat des tests dans le fichier test\_result.txt : *

Pour certains tests, comme l'interaction via Web3.js ou l'intégration avec Ganache, il n'y a pas de preuve formelle (aucun fichier spécifique), mais ils ont été validés en observant l'exécution des scripts en temps réel.

## Erreur restante

Installation d’un explorateur de blockchain d’ethereum

Actuellement, nous avons une blockchain, une DAPP pouvant interagir dans cette blockchain mais nous n’avons pas d’explorateur de blockchain d’ethereum, ce n’est pas essentiel en terme technique, c’est surtout une question d’esthétique, nous pouvons visualiser tout de même par ligne de commande malgré qu’on n’ait pas de GUI.

## Liste des documents fournis

*Lister les documents fournis au client avec votre produit, en indiquant les numéros de versions*

* *le rapport de projet*
* *le manuel d'Installation (en annexe)*
* *le manuel d'Utilisation avec des exemples graphiques (en annexe)*
* *autres…*

# Conclusions

## Objectifs et Résultats

|  |  |  |
| --- | --- | --- |
| *Objectif* | *Description* | *Statut* |
| *Mettre en place un environnement de développement Ethereum complet* | Installation et configuration d’un environnement Ethereum avec Hardhat ou Truffle et un réseau Ethereum local sur une machine virtuelle Debian 12. | *✅ Atteint* |
| *Exploration et Compréhension de la Blockchain* | Apprentissage des bases de la blockchain via la pratique et l'étude théorique. Mise en place d'une première dApp en suivant un tutoriel. | *✅ Atteint* |
| *Utiliser une interface utilisateur simple (Objectif optionnel)* | Intégration d'une interface graphique permettant l’interaction avec la blockchain, la visualisation des transactions et l’exécution d’opérations sur les smart contracts. | *✅ Atteint* |
| *Déployer une application décentralisée (dApp) fonctionnelle sur la blockchain Ethereum (Objectif optionnel)* | Développement et déploiement d'une dApp sur une blockchain Ethereum interne pour interagir avec les smart contracts. | *✅ Atteint* |

*Points positifs / négatifs*

* *Difficultés particulières*

## Suites possibles pour le projet (évolutions & améliorations)

### Développement d’une interface graphique dédiée

Une évolution majeure serait la création d’une interface graphique à Hardhat permettant d’interagir avec les smart contracts sans passer par Remix IDE. Cette interface offrirait une visualisation en temps réel des transactions et des blocs générés sur le réseau Ethereum, rendant l’utilisation plus accessible.

### Création de dApps et de smart contracts plus complexes

Le projet pourrait être poussé plus loin en concevant une application décentralisée (dApp) mais avec des fonctionnalités plus avancées. Cela inclurait la gestion d’identité décentralisée, des contrats intelligents interconnectés ou encore l’intégration avec des API externes pour enrichir les interactions.

### Déploiement sur un réseau local (ETML)

Une autre amélioration possible serait de déployer la blockchain sur plusieurs postes en l’hébergeant sur le réseau local de l’ETML, notamment sur les serveurs de l’établissement. Cela permettrait d’expérimenter un environnement distribué, d’analyser les performances d’un réseau Ethereum multi-nœuds et de simuler des cas d’usage réels en conditions locales.

# Annexes

## Résumé du rapport du TPI / version succincte de la documentation

## Sources – Bibliographie

1. <https://mythril-classic.readthedocs.io/en/master/about.html>
2. <https://github.com/curl/curl/releases>
3. <https://github.com/ethereum/remix-ide/releases>
4. <https://stackoverflow.com/questions/32426601/how-can-i-completely-uninstall-nodejs-npm-and-node-in-ubuntu>
5. <https://www.reddit.com/r/solidity/comments/r0k95k/best_gui_for_hardhat/?rdt=50906>
6. <https://hardhat.org/>
7. <https://scaffold-eth-2-docs.vercel.app/>
8. <https://remix.ethereum.org/>
9. <https://formationdigitalmarketing.ch/guide-ultime-pour-la-creation-dune-blockchain/>
10. <https://appstronaute.com/creer-une-blockchain/>
11. <https://ibmcloud.developpez.com/tutoriel/creation-reseau-de-base-de-chaines-de-blocs-blockchain/#LIII-B>
12. <https://www.malekal.com/ubuntu-ouvrir-installer-fichier-deb-paquet/>
13. <https://www.malekal.com/comment-creer-un-fichier-sur-linux-terminal-ligne-de-commandes/>
14. <https://mcflypartages.fr/blog/linux_installer_deb/>
15. <https://askubuntu.com/questions/1193484/how-to-display-version-only-of-curl>
16. <https://www.malekal.com/comment-ajouter-utilisateur-sudo-sudoers-ubuntu-debian/>
17. <https://www.reddit.com/r/debian/comments/1cv6b2b/create_user_with_root_permissions/?rdt=36663>
18. <https://stackoverflow.com/questions/10227590/change-user-in-linux-script>
19. <https://superuser.com/questions/93385/run-part-of-a-bash-script-as-a-different-user>
20. <https://www.pluralsight.com/resources/blog/software-development/linux-add-user-command>
21. <https://geth.ethereum.org/>
22. <https://medium.com/coinmonks/testing-solidity-with-truffle-and-async-await-396e81c54f93#:~:text=To%20use%20async%2Fawait%20you,%2Fmethod%2C%20not%20a%20callback>!

*Liste des livres utilisés (Titre, auteur, date), des sites Internet (URL) consultés, des articles (Revue, date, titre, auteur) … Et de toutes les aides externes (noms)*

Au vu de la difficulté de prise en main du projet nous avons coopérer avec Monsieur Dylan Artero ayant le même cahier des charges.

## Journaux de travail

## Manuel d'Installation

## Installation de Remix IDE sur Ubuntu 12.



## Introduction

Remix IDE est un environnement de développement intégré (IDE) populaire pour le développement de contrats intelligents sur Ethereum. Cette documentation vous guidera à travers les étapes nécessaires pour installer Remix IDE sur Ubuntu 12.0.

## Prérequis

## Mise à jour du système

## Avant d'installer Remix IDE, il est recommandé de mettre à jour votre système. Ouvrez un terminal et exécutez la commande suivante pour mettre à jour les paquets : sudo apt-get update && sudo apt-get upgrade -y

## Installation de curl

## Si curl n'est pas installé sur votre système, vous pouvez l'installer en utilisant la commande suivante :

sudo apt-get install curl -y

## Téléchargement et installation de Remix IDE

Une fois curl installé, vous pouvez télécharger et installer Remix IDE avec la commande suivante :  
curl -sLO https://github.com/remix-project-org/remix-desktop-insiders/releases/download/v1.0.8-insiders/remixdesktop\_1.0.8-insiders\_amd64.deb && sudo dpkg -i remixdesktop\_1.0.8-insiders\_amd64.deb

## Explication des commandes

* curl -sLO <URL> : Cette commande télécharge le fichier .deb depuis le dépôt officiel sans afficher de détails supplémentaires (-s pour silencieux) et conserve le nom du fichier d'origine (-O).
* sudo dpkg -i <fichier.deb> : Cette commande installe le fichier .deb téléchargé.

## Résolution des problèmes

## Erreur de dépendances manquantes

Si l'installation échoue en raison de dépendances manquantes, vous pouvez résoudre ce problème en exécutant la commande suivante :  
sudo apt-get install -f  
Ensuite, relancez l'installation avec :  
sudo dpkg -i remixdesktop\_1.0.8-insiders\_amd64.deb

## Vérification de l'installation

Pour vérifier si Remix IDE a été correctement installé, lancez la commande suivante dans un terminal :  
remix-ide  
Si Remix IDE ne s'ouvre pas, essayez de redémarrer votre système.

## Conclusion

Félicitations ! Vous avez installé Remix IDE sur Ubuntu 12.0 avec succès. Vous pouvez maintenant commencer à développer vos contrats intelligents Ethereum.

## Manuel d'Utilisation

### Interaction avec les smart contracts en utilisant truffle

### Étape 1 : Déploiement du smart contract

**Commande :**

truffle migrate

**Description :** Cette commande compile vos smart contracts (si nécessaire) et les déploie sur la blockchain spécifiée dans votre fichier de configuration (truffle-config.js).

### Étape 2 : Ouverture de la console interactive

**Commande :**

truffle console

**Description :** Cette commande ouvre une console interactive JavaScript connectée à votre réseau Ethereum local ou distant, permettant ainsi d'interagir directement avec les smart contracts.

### Étape 3 : Interagir avec le smart contract

**Commande :**

let Add = await Addition.deployed()

**Description :** Cette commande permet de récupérer une instance du smart contract nommé Addition que vous avez déployé préalablement. Elle sera utilisée pour appeler ses fonctions.

**Commande :**

await Add.addx(1, 3)

**Description :** Cette commande appelle la fonction addx du smart contract Addition avec les paramètres fournis (ici : 1 et 3).

**Commande :**

await Add.retrievex()

**Description :** Cette commande appelle la fonction retrievex du smart contract Addition, récupérant ainsi la valeur stockée après l'appel à la fonction précédente.

Notes supplémentaires :

* Assurez-vous que Ganache (ou un autre environnement Ethereum) est lancé avant d'utiliser ces commandes.
* Vérifiez que votre smart contract Addition dispose bien des fonctions addx et retrievex correctement définies. Les fonctions doivent être dans le smart contract

### Utilisation du script de mise en place de l’environnement

Aller dans le dossier puis ouvrir un terminal et entrer :

touch nom\_du\_script.sh

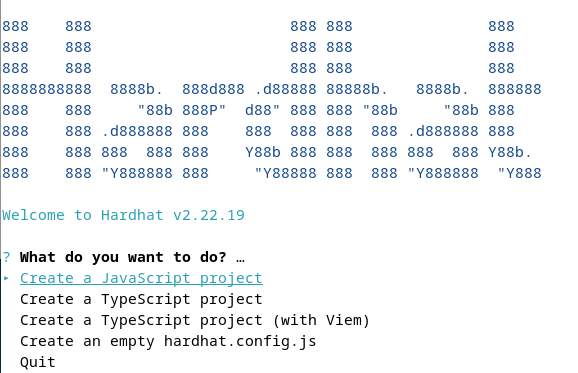
Aller dedans et éditer avec un éditeur puis coller le code, si vous ne pouvez pas, importer notre script sur la VM.

Puis il faut mettre votre utilisateur non-root (chez nous eleve) dans le dossier des sudoers.

Pour ça il faut se mettre en sudo avec "su" puis : sudo usermod -aG sudo nom\_utilisateur

Ensuite il faut bien être dans le dossier du script et l'exécuter avec : sudo su nom\_utilisateur ./nom\_du\_script.sh

Puis on arrive sur le menu console d'Hardhat qui ressemble à ça :



Ici il faut choisir ‘’Create a JavaScript project’’ avec enter.   
  
Ensuite il nous propose un dossier de base pour le projet, dans notre cas celui par défaut est juste. Donc ‘’Enter’’.  
  
Dans la suite il nous demande si on veut un gitignore et si on veut partager nos données pour qu’ils améliorent Hardhat, là c’est à vous de choisir ce qui vous convient

Puis il demande d’installer des dépendances d’Hardhat, nous n’avons pas besoin car elles sont installées avec le script => n.  
  
Le script continue et avant la fin on peut apercevoir les différents comptes créés pour utiliser cette blockchain.

## On voit aussi plus bas que notre contrat s’est bien compilé avec quelques informations comme l’adresse et le Gas utilisé pour cette transaction. Manuel d'Utilisation

### Interaction avec les smart contracts en utilisant Remix IDE

Après avoir installé correctement Remix IDE et avoir configuré votre smart contract, vous devez sélectionner un smart contract et appuyer sur le bouton « Compile » qui est le bouton play en vert

## Archives du projet

*Media, … dans une fourre en plastique*