Secured WebShop



Dario Chasi – CID2A

Vennes

32 périodes

M.Sonney

Table des matières

[1 Spécifications 3](#_Toc167577064)

[Web Store 3](#_Toc167577065)

[Description 3](#_Toc167577066)

[Matériel et logiciels à disposition 3](#_Toc167577067)

[Prérequis 3](#_Toc167577068)

[Cahier des charges 3](#_Toc167577069)

[1.1.1 Dockerisation 3](#_Toc167577070)

[1.1.2 Profil du client 3](#_Toc167577071)

[1.1.3 HTTPS Il doit être possible d’accéder à votre site de e-commerce de manière sécurisée (https://localhost). Le port utilisé sera le 443.Le certificat sera auto-signé par OpenSSL. 3](#_Toc167577072)

[1.1.4 Authentification par mot de passe 3](#_Toc167577073)

[1.1.5 Administration 3](#_Toc167577074)

[1.1.6 Protection contre les injections SQL 3](#_Toc167577075)

[1.1.7 Versioning 4](#_Toc167577076)

[Les points suivants seront évalués 4](#_Toc167577077)

[Validation et conditions de réussite 4](#_Toc167577078)

[2 Analyse 4](#_Toc167577079)

[Document d’analyse et conception 4](#_Toc167577080)

[3 Réalisation 4](#_Toc167577081)

[Structure du code 4](#_Toc167577082)

[Installation de packages 5](#_Toc167577083)

[Authentification 5](#_Toc167577084)

[3.1.1 Vérification du token 5](#_Toc167577085)

[3.1.2 Authentification Admin 6](#_Toc167577086)

[Https 6](#_Toc167577087)

[Modèle 7](#_Toc167577088)

[3.1.3 Table users 7](#_Toc167577089)

[Insertion base de données 8](#_Toc167577090)

[3.1.4 Connexion à la base de données 8](#_Toc167577091)

[3.1.5 Hachage de mot de passe 8](#_Toc167577092)

[3.1.6 Fonction insertUsers 9](#_Toc167577093)

[Contrôleurs 10](#_Toc167577094)

[3.1.7 Login 10](#_Toc167577095)

[3.1.8 Users All 11](#_Toc167577096)

[3.1.9 Users nickname 11](#_Toc167577097)

[3.1.10 AddUser 12](#_Toc167577098)

[Routes 12](#_Toc167577099)

[FrontEnd 13](#_Toc167577100)

[4 Conclusion 14](#_Toc167577101)

[Bilan des fonctionnalités demandées 14](#_Toc167577102)

[Bilan personnel 14](#_Toc167577103)

[5 Divers 14](#_Toc167577104)

[Webographie 14](#_Toc167577105)

[6 Annexes 14](#_Toc167577106)

[Code 14](#_Toc167577107)

[Journal de travail 14](#_Toc167577108)

[Présentation 14](#_Toc167577109)

# Spécifications

## Web Store

Création d’un site d’e-commerce sécurisé

## Description

Au terme du projet, l’apprenti sera capable de construire une application node.js offrant un accès sécurisé et une gestion des rôles.

## Matériel et logiciels à disposition

• Un ordinateur standard de la section informatique avec Docker Desktop

## Prérequis

Modules 294, 335

## Cahier des charges

### Dockerisation

L’ensemble des services web sera conteneurisé.

### Profil du client

Le client peut accéder à son propre profil en utilisant un lien tel que : https://localhost/users/john

Seul son profil lui sera rendu visible.

### HTTPSIl doit être possible d’accéder à votre site de e-commerce de manière sécurisée (https://localhost). Le port utilisé sera le 443.Le certificat sera auto-signé par OpenSSL.

### Authentification par mot de passe

L’utilisateur devra s’authentifier par la page https://localhost/login

Le mot de passe sera hashé et salé avant d’être stocké dans la base de données (tablet\_users)

### Administration

Une page d’administration devra avoir un champ de recherche (Nom du visiteur) et permettre d’afficher tous les utilisateurs ayant tout ou partie de ce nom.

### Protection contre les injections SQL

Votre page d’administration devra être protégée contre les injections SQL.

Sans utiliser sequelize ou tout autre ORM, votre site devra être robuste face au

### Versioning

Votre code sera versionné sur Github et un .gitignore empêchera de versionner les binaires npm (dossiers node\_modules). Votre dépôt sera partagé avec votre chef de projet.

## Les points suivants seront évalués

* Le rapport
* Le journal de travail
* Une conclusion sur le travail fourni et sur l’attitude face au projet

## Validation et conditions de réussite

* Compréhension du travail
* Possibilité de transmettre le travail à une personne extérieure pour le terminer, le corriger ou le compléter
* Etat de fonctionnement du produit livré

# Analyse

## Document d’analyse et conception

Pour ce projet, un dossier zip nous est fourni avec des conteneurs Docker, ainsi que la restriction de ne pas utiliser Sequelize ni bcrypt. Notre site doit être accessible en utilisant HTTPS. Nous devons créer un site web protégé contre les injections SQL et gérer la création des rôles.

# Réalisation

## Une image contenant texte, capture d’écran, logiciel Description générée automatiquementStructure du code

Voici la structure de mon code pour le bon fonctionnement de mon application, en séparant les fichiers concernant l’authentification, les certificats, la base de données, les contrôleurs, les modèles et les routes dans différents répertoires :

## Installation de packages

Le zip de base reçu par l’enseignant manqué de certain packages, j’ai commencé par installer :

* jsonwebtoken
* crypto
* cors
* mysql2
* http-server

## Authentification

### Vérification du token

Pour cette première authentification, j’ai créé un système qui vérifie l’authenticité du token à l’aide d’une clé privée, qui, dans un cas réel, devrait être stockée ailleurs.

Une image contenant texte, Appareils électroniques, capture d’écran, affichage

Description générée automatiquement

### Authentification Admin

Pour la deuxième authentification l’utilisateur doit être admin pour y accéder au différents ressources.

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

## Https

Avec OpenSSL préinstallé sur le PC de l’école, j’ai utilisé la commande suivante pour obtenir les certificats autosignés :

openssl req -nodes -new -x509 -keyout server.key -out server.cert

Une fois cette étape complétée, j’ai ajouté les certificats à mon projet et configuré mon server.js pour son bon fonctionnement :

Une image contenant texte, Appareils électroniques, capture d’écran, affichage

Description générée automatiquement

Comme https et fs sont déjà présents sur node, j’ai ne pas eu besoin d’installer aucun package supplémentaire.

## Modèle

Une image contenant texte, capture d’écran

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquementJ’ai ainsi créé un modèle pour mes utilisateurs, qui empêche l’utilisation de caractères spéciaux, l’insertion de valeurs nulles, impose une longueur maximale de caractères et une longueur minimale pour le mot de passe :

### Table users

Table appartenant à ce modèle

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

## Insertion base de données

### Connexion à la base de données

Pour faire la connexion à ma base des données j’ai créé un pool avec les différentes informations de ma base des données.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

### Hachage de mot de passe

Pour hasher le mot de passé, j’ai créé un salt avec le nickname pour le rendre unique, puis j’ai le rajouté au hachage de mon mot de passe.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

### Fonction insertUsers

Ensuite pour l’insertion correcte des utilisateurs présents dans mon mock des utilisateurs, je me connecte à ma base de données, puis je vérifie si les données insérées correspondent à mon modèle. Je m'assure également de hasher le mot de passe avant de l’insérer dans ma base de données avec le salt correspondant.

Une image contenant texte, capture d’écran

Description générée automatiquement

Ensuite, j’utilise cette fonctionne pour insérer mes utilisateurs.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

## Contrôleurs

### Login

Pour mon contrôleur de login, l’utilisateur doit entrer son pseudo et son mot de passe. Une fois cela fait, j’utilise le salt sauvegardé au préalable et je compare les hashes.

Une image contenant texte, capture d’écran

Description générée automatiquement

### Users All

Contrôleur pour l’obtention de tous les utilisateurs avec des requêtes préparées, un message affiché aux utilisateurs et un autre message affiché dans la console.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

### Users nickname

Contrôleur pour l’obtention des utilisateurs par rapport à son nickname avec des requêtes préparées, un message affiché aux utilisateurs et un autre message affiché dans la console.

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

### AddUser

La route addUser est la même que pour l’insertion automatique des utilisateur présents dans le mock

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

## Routes

Ensuite l’utilisation des différents moyens d’authentification dépendant des routes

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

## FrontEnd

Création du frontend avec vue.js. (npm run dev pour l’activer)

L’utilisateur doit entrer ses identifiants pour effectuer une recherche des utilisateurs et doit être administrateur pour afficher tous les utilisateurs.

Une image contenant capture d’écran, texte, diagramme, ligne

Description générée automatiquement

# Conclusion

## Bilan des fonctionnalités demandées

L’utilisateur peut accéder à son compte avec son pseudo. Le rôle administrateur a été créé ainsi que les certificats pour le HTTPS. Pour les contrôleurs, des requêtes préparées ont été mises en place, et l’authentification a été ajoutée aux routes. Des limitations de caractères et des restrictions sur les caractères spéciaux ont été implémentées pour éviter les injections SQL. Le mot de passe est hashé avec un salt. Des vues en Vue.js ont été créées pour la barre de recherche, la connexion et l’affichage de tous les utilisateurs.

## Bilan personnel

Pour ce projet, j'aurais souhaité recevoir un backend plus avancé afin de pouvoir expérimenter plus sur les autres moyens de sécurité. J'ai apprécié les restrictions, telles que l'interdiction d'utiliser Sequelize ou bcrypt, car elles m'ont permis d'apprendre de nouvelles méthodes de création d'un backend, avec les contrôleurs et les routes séparés chose que nous n’avons pas vu durant le module backend. Cependant, ce choix ne m'a pas permis de progresser à la vitesse souhaitée.

# Divers

## Webographie

[OpenSSL - CrypTool](https://www.cryptool.org/en/cto/openssl/)

[Form Data Validation in Node.js with express-validator (stackabuse.com)](https://stackabuse.com/form-data-validation-in-nodejs-with-express-validator/)

[Using express-validator for data validation in NodeJS | by Chau Nguyen | Medium](https://medium.com/@hcach90/using-express-validator-for-data-validation-in-nodejs-6946afd9d67e)

<https://www.cockroachlabs.com/docs/stable/create-security-certificates-openssl>

# Annexes

## Code

## Journal de travail

## Présentation