Rapport de stage Année 2020-2021

DUT Informatique

2 Novembre 2020 – 29 Janvier 2021

SAINT-SORNY Evan

Mme Fort

Développeur junior web/javascript

Conception et développement d'une sous application web, liée au projet principal

(Application multimédia embarquée dans des systèmes propriétaires)

Deepidoo
Addicted to retail

22 Quai du docteur Gailleton 69002

Lyon 2ème France

Mr Michel

Université Claude Bernard - Lyon 1 Institut Universitaire de Technologie

Département informatique

43 Boulevard du 11 Novembre 1918 69622 VILLEURBANNE Cedex

Tél : 04-72-69-21-90

Fiche technique

Nom de l'entreprise : Deepidoo - Addicted to retail

Activité de l'entreprise : Expert en animation et digitalisation des points de ventes.

- Solution web de *marketing sensoriel* (voir définition page 3) pour piloter à distances des ***animations sonores (Deepisound), visuelles (Deepiscreen) et olfactives (Deepiscent),*** afin de créer une expérience client.
- Présent à l'internationale.

Intitulé du sujet : Développeur junior web/javascript : Conception et développement d'une sous application web embarquée, liée au projet principal (Application multimédia embarquée dans des systèmes propriétaires).

L'application web va permettre à l'utilisateur/client de modifier certains paramètres de la solution physique (**Deepibox :** voir Annexe 2&3), sans avoir à renvoyer le matériel à l'équipe informatique, cela évite des coûts/frais supplémentaires et aussi servir de couche d'abstraction entre le client et la solution.

L'utilisateur doit pouvoir modifier des paramètres tels que la configuration réseau (Ethernet, Wi-Fi), la résolution de la/les sortie vidéo (écran), mais encore le Bluetooth ou le volume du Player audio.

Cette application web embarquée dans la **Deepibox** sera utilisée pour d'éventuelles modifications de paramétrages.
Pas de contraintes de temps forts pour la réalisation de l'application web.

L'entreprise a une équipe informatique constitué de 2 personnes.
Je travaille seul sur un projet en autonomie sous la directive de Mr Michel.

Matériel utilisé : **Deepibox** (mini-pc : système d'exploitation Ubuntu Gnome une distribution de Linux), écran, clavier et souris.

Langages et outils utilisés : **Gnome-terminal** (interpréteur/terminal/console de commande Linux de Ubuntu Gnome), **Javascript** (langage de programmation), **Xrandr** (outil en ligne de commande de gestion de paramètres d'affichage) et **Network Manager** (l'outil de gestion des connexions réseau d'Ubuntu).

Environnement d'exécution (logiciel responsable de l'exécution des programmes) : **Node.js**
Le choix du logiciel IDE (environnement de développement : ensemble d'outils qui permet d'augmenter la productivité) est libre. Il doit être compatible avec **Node.js et le langage Javascript**.
J'ai choisi d'utiliser **Visual studio code** par habitude et connaissance de celui-ci (logiciel gratuit et simple d'utilisation, debugger inclus, coloration syntaxique, mise en forme du code automatique, nombreux plugins).

Les outils **Xrandr** et **Network Manager** étant nécessaires au projet de l'application web, j'ai acquis les connaissances et compétences par le biais de documentations, forums, sites webs et tests des différentes fonctions.

Travail à terminer avant la fin du stage : Initialement prévu les modules configuration réseau et résolution d'écran doivent être fonctionnels, et si le temps le permet l'ajout de deux modules pertinents supplémentaires (Bluetooth, volume audio) envisagés lors d'une discussion avec l'équipe le midi.

Temps de rédaction pour le rapport : Dans l'intervalle des deux mois de stage, planning souple

Remerciements

Je tiens tout d'abord à remercier **Mr Feminier et Mr Assous** de m'avoir accueilli au sein de leur entreprise dans le cadre de ce stage, qui m'a permis de développer des compétences et d'avoir une expérience professionnelle enrichissante.

Je veux également remercier Marc Michel, mon maître de stage, pour sa disponibilité, et pour m'avoir encadré, suivi, formé et conseillé tout au long du stage.

Je tiens aussi à remercier toute l'équipe de Deepidoo pour l'accueil chaleureux, leur sympathie et les bons moments passés ensemble qui ont fait de ce stage un moment très profitable.

Je veux aussi adresser mes remerciements à **Mme Deboute** pour m'avoir aidé et accompagné dans ma recherche de stage.

Enfin je tiens aussi à remercier **Mme Fort et** l'ensemble de mes professeurs Pour m'avoir apporté les connaissances nécessaires au bon déroulement de ce stage et à la poursuite de mes études.

Sommaire :

Introduction

J'ai choisi cette entreprise dans l'optique de poursuivre mes études, dans le domaine du développement logiciel, d'application ainsi que la sécurité informatique à l'université Lyon 1 Claude Bernard (licence pro DevOps). En entrant dans la vie professionnelle, j'ai eu l'opportunité d'intégrer l'entreprise **Deepidoo** par une connaissance familiale pour réaliser mon stage de fin d'études d'une durée de 2 mois.

Je suis passionné depuis longtemps par le domaine de la sécurité informatique et du web avec son explosion ces dernières années et ses applications informatiques. Pour cela, le sujet qui m'a été proposé, de la part de cette entreprise, est une occasion de mettre en application mes capacités.

Deepidoo est une société experte et unique en son genre, en animation et digitalisation des points de ventes (marketing sensoriel voir page 3) avec une solution unique **:** une plateforme web afin de gérer et piloter à distances ***des animations sonores (Deepisound), visuelles (Deepiscreen) et olfactives (Deepiscent)***, pour créer une expérience client.

Deepidoo est présent en France comme à l'international avec 30% de son chiffre d'affaire réalisé à l'export.

La société est leader de son marché.

Du 2 Novembre 2020 au 29 Janvier 2021 j'ai effectué mon stage malgré la situation sanitaire dans l'entreprise **Deepidoo** (située au 22 Quai Gailleton à Lyon 2ème). Une partie de l'équipe était en télétravail mais se réunissait 2 jours par semaine afin de faire un point. La plupart du temps j'ai effectué mon stage en entreprise pour des raisons pratiques afin de communiquer avec mon maître de stage plus facilement.

On m'a intégré dans cette entreprise dans le pôle développement (équipe informatique) pour réaliser un projet qui consiste à concevoir et développer une application web embarqué dans le matériel physique **: Deepibox** (mini-pc pré-configuré). Et ce afin de permettre à l'utilisateur/client de modifier des paramètres de configuration du matériel physique (**Deepibox**) tels que la configuration réseau (Ethernet, Wi-Fi), la résolution de l'écran, le Bluetooth, ainsi que le volume du lecteur(player) audio.

Les objectifs de ce stage sont les suivants :

-Découvrir le monde professionnel.

-Utiliser mes connaissances et compétences en web et les enrichir.

-Présenter et promouvoir les réalisations effectuées.

Dans un premier temps, nous présenterons de façon générale l'entreprise **Deepidoo** avec un résumé en quelques chiffres, la définition du marketing sensoriel, les activités et objectifs de **Deepidoo**, et un historique de la société.

Ensuite, nous décrirons l'organisation de la société et de l'environnement technologique.

Après nous verrons le travail effectué lors du stage au sein de **Deepidoo**.

Et enfin, nous ferons un bilan de l'expérience de stage.

A. Présentation générale de l'entreprise Deepidoo

I. Résumé de Deepidoo en quelques chiffres :

, fondée en 2013 par Yann Féminier et Alexandre Assous, est une société spécialisée dans l'animation et la digitalisation des points de vente.

Figure 1 logo Deepidoo

L'équipe est organisée en pôles (artistique et technologique, développement, commercial) et est constituée de 15 personnes (sans stagiaire ni intervenant, ni partenaire).

La société se situe actuellement au 22 Quai du Dr Gailleton Bellecour, Lyon 2eme arrondissement, France.

Elle a développé la première solution 100% web de **marketing sensoriel** dédiée aux marché B2B (services, commerce de détail « retail », restauration, entertainment…) pour piloter à distance et en temps réel des ***animations sonores, visuelles et olfactives.***

DEEPIDOO est une entreprise novatrice, ambitieuse et performante, ayant 15 collaborateurs et partenariats (dont Samsung depuis 2014 et LG depuis 2019 mais encore GMZ&Sons). Son chiffre d'affaire en 2018 s'élevait à 1 million €, dont 30% en export dans 20 pays différents mais (pas encore implantée en Chine à ce moment-là).

Figure 2 Deepidoo à l'international map

Figure 3 Deepidoo en quelques chiffres

II. Qu'est-ce que le marketing sensoriel ?

Le marketing sensoriel est une technique qui fait intervenir un ou plusieurs des cinq sens du consommateur pour créer une expérience client positive et différenciante basée sur le bien-être, et ainsi stimuler son comportement vis-à-vis d'un service ou d'un produit.

Les entreprises doivent trouver un moyen d'apporter une valeur ajoutée à leur offre pour se distinguer de la concurrence et fidéliser leur clientèle. Le marketing sensoriel représente ici un réel atout pour les secteurs de la vente, de l'animation et du service. Moyen efficace pour stimuler l'intérêt du consommateur, il incarne également une vision novatrice du commerce en plaçant l'expérience client au centre de la démarche. À l'heure où la publicité est omniprésente, la valorisation de l'expérience client prend le contrepied de la stratégie de vente usuelle et intervient comme un levier majeur de valorisation. Créer des expériences authentiques, uniques et mémorables, pour vivre le marketing différemment.

III. Objectifs et activités de l'entreprise Deepidoo :

Mettre le digital au service de l'expérience client.

C'est l'ambition principale de Yann Féminier et Alexandre Assous qui – en fondant **DEEPIDOO** – ont fait le pari de proposer tous les aspects du marketing sensoriel sur une seule et unique plateforme web ***: PLAY by Deepidoo***.

2 visées principales au marketing sensoriel de **DEEPIDOO** :

- *Améliorer le confort et bien-être du consommateur pour garantir une expérience client positive*
- *Stimuler l'acte d'achat*

DEEPIDOO déploie son expertise sur trois types d'animations sensorielles :

Deepisound

Figure 4 logo Deepisound

1. ***Des animations sonores*** : Création de design sonore (playlists personnalisées, sons d'ambiance, messages sonores, expériences uniques et attractives, …)

Deepiscreen

Figure 5 logo Deepiscreen

2. ***Des animations visuelles*** : En partenariat avec SAMSUNG (Affichage dynamique sur écrans, social wall, informations et services, annonces publicitaires, interactivité flash code)

Deepiscent

Figure 6 logo Deepiscent

3. ***Des animations olfactives*** : Diffusions de parfum d'ambiance (Catalogue de 3000 fragrances fabriquées à Grasse ou création d'identité olfactive sur-mesure)

Elle exerce son activité auprès des marchés B2B et en retail dans des secteurs hétérogènes : Bureaux, Gares, Hôpitaux, Magasins, Équipements Publics, EHPAD, Aéroports, Complexes de loisirs, Centres Commerciaux, Cinémas, Stades

PLAY by Deepidoo est une plateforme web innovante en mode SaaS (Software as a Service) unique sur le marché pour la gestion et la diffusion sonore, visuelle et olfactive. La plateforme imaginée par **DEEPIDOO** offre la possibilité – sans nécessiter l'installation d'un logiciel – de programmer et diffuser en temps réel et à distance tous type de contenus multimédias : vidéos, musique, parfums d'ambiance etc.

Ainsi, chaque professionnel, même sans notions techniques, peut imaginer, concevoir et animer un univers sur-mesure pour son établissement en pleine autonomie.

Un concept innovant ***: il n'y a qu'une seule plateforme web pour tout gérer, aucun logiciel est nécessaire, simple d'utilisation, solution autonome ne nécessitant pas de compétences techniques ni de prestataire, la capacité de stockage est illimitée, une diffusion en locale n'utilisant presque pas la bande passante, avec des box autonomes permettant un système de diffusion automatique (Deepibox) et du Plug&Play***

Afin de simplifier au maximum l'utilisation de sa solution, **DEEPIDOO** a mis en place « **Deepibox** », une box qui matériel et pré-configurée, un simple branchement lui permet d'établir sa connexion avec son serveur et sa mise en route automatique.

Figure 0 Deeepibox

Deux modes de diffusion :

• On Line : Si la bande passante le permet, la lecture audio et/ou vidéo est réalisée sur le web. En cas de coupure, la diffusion est assurée grâce au programme de secours stocké sur la **Deepibox**

• Off Line : Les contenus audios et vidéos sont synchronisés et stockés sur la **Deepibox** selon un rythme défini pas le professionnel. La diffusion est réalisée depuis la **Deepibox**, sans utiliser la bande passante de l'espace équipé.

Depuis plusieurs années, le marché de l'affichage dynamique évolue vers des solutions sans player audio : **DEEPIDOO** se veut pionnier sur le marché. La technologie de **DEEPIDOO** est intégrée aux écrans connectés SAMSUNG et LG. **Deepidoo** et ses partenaires proposent ensemble une solution globale d'intégration, de programmation et de diffusion digitale en temps réel et en toute autonomie.

DEEPIDOO fait régulièrement appel à des labels et des DJ pour la partie créative de son offre.

Réalisations & références :

Figure 2 Réalisations & références de Deepidoo

IV. Historique de Deepidoo :

2013 - 2016 : Lancement de la société

2017 :

700 000 CA (mutliplié par 3 par rapport à 2016) 10 personnes 3,4 collaborateur hypercroissance de 0,6
Objectif internationale (Chine,Etat-Unis, ..) + et 2 millions de CA dans 2ans.

2018 :

CA 2018 : 1million€
Effectif 10 personnes
Réalisation pour NAF NAF, Nocibé, Alice délice, Lacoste, Courtepaille 3.000 à ,4.000 point de ventes équipés présent dans une vingtaine de pays.
Signature de contrat avec ID Group/IDKIDS en fin d'année (enseignes Jacadi, Okaidi et Oxybul)

Figure 3 logo Pass French Tech

L'entrée au Pass French Tech, véritable atout business pour **DEEPIDOO**

DEEPIDOO intègre le très sélect Pass French Tech en Mai 2018, programme national ayant pour objectif d'amplifier le développement d'entreprises en hyper-croissance à très fort potentiel. Ce nouveau soutien représente pour l'entreprise une opportunité de développer de nouveaux leviers distributeurs et financiers, ainsi qu'une ouverture possible à l'export. L'entreprise souhaite devenir l'un des leaders reconnus sur le secteur du marketing sensoriel et de l'optimisation des lieux de vente. Prochaine étape : l'international. : « Nous avons vocation à suivre nos clients français dans leur développement à l'international », poursuit Yann Féminier qui réfléchit par ailleurs à quels marchés cibles s'adresser : « Les Etats-Unis et la Chine nous intéressent ».

Pour financer cette internationalisation, **Deepidoo** ne prévoit pas à court terme de levée de fonds : « Nous sommes à l'équilibre et prévoyons d'atteindre 1,2 million d'euros de chiffre d'affaires cette année contre 700.000 euros en 2017 »

Participation & sélection au FRENCH TECH TOUR CHINA 2018 programme de développement business pour les start-ups françaises. Ce programme d'immersion permet aux start-ups de la French Tech de saisir les opportunités du marché chinois (deux semaines d'immersion dans les villes-écosystèmes de l'Innovation chinoise : Shanghai, Shenzhen, Hong-Kong ou Chengdu et Pékin). Rencontre des prospects et partenaires potentiels lors des deux semaines d'immersion (en 2019).

2019 :
Refonte de la plateforme web et embauche de personnels supplémentaires.
Objectif : implantation à Paris CA double par rapport à 2018 (grâce piste marché chinois)
1 M€ de CA en Asie dès 2021.
Nouveau client et réalisation : Porsche et e réseau mondial des 1800 magasins Okaïdi

Figure 4 Club Digital Media

Trophée de l'Expérience Phygitale "Hub Retail (association « Cross-Canal et Omni-Logistique » d'entreprise de la région Auvergne-Rhône-Alpes) 2019" Membre certifié OLED par le constructeur LG

Rachat de l'enseigne NAF NAF par un groupe chinois qui projette d'ouvrir 500 points de vente en deux ans dans leur pays. Ils font un premier test avec 10 ouvertures entre mars et septembre 2019 et **Deepidoo** devrais être leur fournisseur pour créer l'ambiance musicale de ces magasins.

Intégration du marché chinois par un premier séjour en Chine où les enseignes locaux ont été séduites par la nouveauté tri-sensorielle proposée par Deepidoo et le principe de hub par lequel tout est connecté, sur-mesure et à usage simplifié.
Deepidoo développe sa solution pour un réseau de 70 magasins Pricerite, enseigne chinoise spécialisée dans l'ameublement intérieur et l'électroménager basée à Hong-Kong.
Investissement pour le marché chinois (web différent) : achat d'une structure et de serveurs car web chinois pas accessible depuis la France.

2020 :
Propose des playlists grand public
Participe à l'effort collectif du confinement
Podcasts, playlists gratuites et grand public
Participation à l'Euroshop 2020 à Dusseldorf l'un des plus grands salons mondiaux dédiés au retail. Organisé tous les 3 ans à Düsseldorf, capitale industrielle allemande.

V. Environnement de stage au sein de Deepidoo

1. L'entreprise Deepidoo comme organisation :

Pôle commercial :
-Développement commercial
-Marketing produit
-Gestion de projets
-Avant-vente

Figure 5 Yann Feminier co-fondateur

Figure 6 Alexandre Assous co-fondateur

Nurh (directeur administratif et financier), Anne-Laure Poyard(responsable administrative) ,

Figure 7 Arthur Gagneux responsable opérationnel & accompagnent et conseil

Figure 13 Alexis Muntaner Business développer, accompagnent et conseil

Figure 14 Emmanuelle Dehaudt Assistante commerciale, accompagnent et conseil

Alban (chargé de communication à Paris), Marion Arfelis (business développer)

Pôle artistique et technologique :
-Accompagnent et conseil
-Stratégie digitale
-Création de contenu

Figure 15 Kévin Mussard Responsable artistique, accompagnent et conseil , chef de projet

Figure 16 Mathieu Chiron technicien accompagnent et conseil, développeur web

Luca (responsable artistique), Anthony Lux (technicien support Hotline)

Pôle développement :
-R&D / Solutions sur-mesure
-Administration système
-Veille innovation

Figure 17 Guillaume Barillot CTO

Figure 18 Marc Michel développeur web

Place dans l'organisation :

En tant que stagiaire je fais partie du pôle développement (service informatique).
Je travaille en autonomie, sous la directive de Mr Michel.
Mes horaires sont variables, mais généralement 10h- 18h, sous l'encadrement de Mr Michel et avec des mises au point quotidiennes de l'avancement du projet et du travail effectué.
Espace de travail : bureau en open space avec l'équipe de développeurs (pôle développement), avec tout le matériel informatique nécessaire.

2. Présentation de l'environnement technologique :

Langages et outils utilisés pour l'application web :

-Matériels utilisés : Deepibox (ou Shuttle voir Annexe 2 et 3). Il s'agit d'un mini pc préconfiguré avec comme **système d'exploitation (voir glossaire)** une distribution de **Linux : Ubuntu Gnome.** Clavier,écran,souris.

-Gnome-terminal (interpréteur/terminal/console de commande Linux de Ubuntu Gnome système d'exploitation de le **Deepibox**),
-**Javascript** (langage de programmation),
-**Xrandr** (outil en ligne de commande de gestion de paramètres d'affichage),
- **Network Manager** (l'outil de gestion des connexions réseau d'Ubuntu),
- Environnement d'exécution (logiciel responsable de l'exécution des programmes) **: Node.js**
-Logiciel IDE (ensemble d'outils qui permet d'augmenter la productivité) **: Visual studio code**

Le choix du logiciel IDE (environnement de développement) est libre.
Il doit être compatible avec **Node.js et le langage Javascript.**
J'ai choisi d'utiliser **Visual studio code** par habitude et connaissance de celui-ci (logiciel gratuit et simple d'utilisation, debugger inclus, coloration syntaxique, indentation du code automatique, nombreux plugins).

Aucunes connaissances de base sur les outils **Xrandr** et **Network Manager** (en ligne de commande) étant nécessaires au projet de l'application web. J'ai acquis les connaissances et compétences par le biais de documentations, forums, sites webs et tests des différentes fonctions.

Maîtrise de diverses connaissances et compétences de base acquises à l'iut informatique sur l'interpréteur de commande Linux, sur Node.js, et sur Javascript. Approfondissement des connaissances et compétences grâce aux documentations, différents sites webs (e-learnig tels que openClassroom, w3school, developpeur Mozilla…) et forums. Découverte des promesses et callback Javascript (voir glossaire)

Maîtrises des connaissances et compétences acquises notamment à l'IUT sur les notions de réseau et de sa configuration.

Les outils utilisés et les technologies au sein de l'entreprise :

slack

Est une plateforme de communication collaborative organisé en canaux de discussion

WhatsApp est une application mobile multiplateforme qui fournit un système de messagerie instantanée chiffrée de bout en bout aussi bien via les réseaux de téléphonie mobiles que par Internet

AirCall est une solution de téléphonie d'entreprise cloud connectés aux différents outils CRM (gestion de la relation client), permet à Deepidoo d'accompagner ces clients et de dépanner les solutions mises en place.

Technologies et langages : Serveur interne avec une base de données dans la société Deepidoo.Langage de programmation du système de gestion de base de données (SGBD) SQL.

Electron est un environnement permettant de développer des applications multi-plateformes de bureau avec des technologies web (JS, HTML, CSS). L'infrastructure est codée en node.js, et l'interface est bâtie sur les outils Chromium, la partie open source de Google Chrome

Ruby un langage de programmation orienté objet et open source

Ruby on rails framework de Ruby structure des fichiers en pattern MVC (modèle vue contrôleur).

JavaScript et VueJS un framework de JavaScript

B. Travail effectué au sein de Deepidoo

Réalisation d'une application web (single page, et embarqué) proposée par Mr Marc Michel de l'équipe informatique afin de permettre à l'utilisateur/client de modifier certains paramètres de la **Deepibox** sans avoir à la renvoyer à l'équipe informatique, cela évite des coûts/frais supplémentaires de transports et de maintenance et aussi servir de couche d'abstraction entre le client et la solution.

Explication : Quand la **Deepibox** est remis au client celui-ci a juste à la brancher car elle est préconfigurée et fonctionne en autonomie. Cependant si le client change par exemple de box internet il faudra donc modifier la configuration réseau de la/les **Deepibox** qui n'a plus la vocation d'ordinateur à ce moment-là et ne comporte plus de bureau, ni de clavier, ni de souris afin que le client ne puisse pas effectuer de modifications. Le client doit donc renvoyer le matériel à l'équipe informatique ce qui n'est pas pratique. Cette application web embarqué dans la **Deepibox** doit donc permettre au client de modifier ces paramètres depuis un autre ordinateur.

Besoins fonctionnels : L'utilisateur doit pouvoir modifier à distance des paramètres tels que la configuration réseau (Ethernet, wifi), la résolution de la/les sortie vidéo(écran), mais aussi le Bluetooth ou le volume du lecteur audio de la **Deepibox** afin d'éviter des coûts et frais de transport ainsi que de maintenance.

Besoins non fonctionnels : L'application doit être extensible, c'est-à-dire qu'il pourra y avoir une possibilité d'ajouter ou de modifier de nouvelles fonctionnalités ; l'application doit adopter une interface de paramétrage conviviale et simple d'utilisation. L'application doit être facile à maintenir et à être repris par un informaticien (Factorisation et découpage du code en plusieurs fichiers qui suit une logique ou pattern).

Cette application web sera utilisée pour d'éventuelles modifications de paramétrages.

Résultats attendus : Initialement prévu les modules configuration réseau et résolution d'écran doivent être fonctionnels, et si le temps le permet ajout de deux modules pertinents supplémentaires (Bluetooth, volume audio) obtenus lors d'une discussion avec l'équipe.

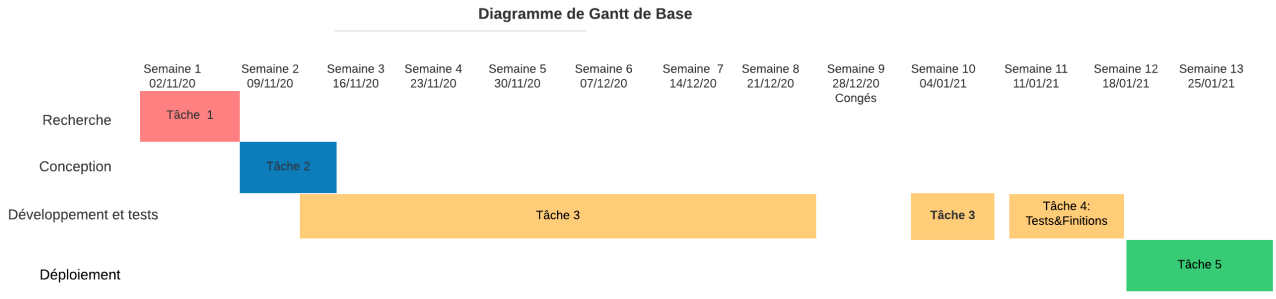


Figure 19 Diagramme de Gantt de la réalisation de l'application web

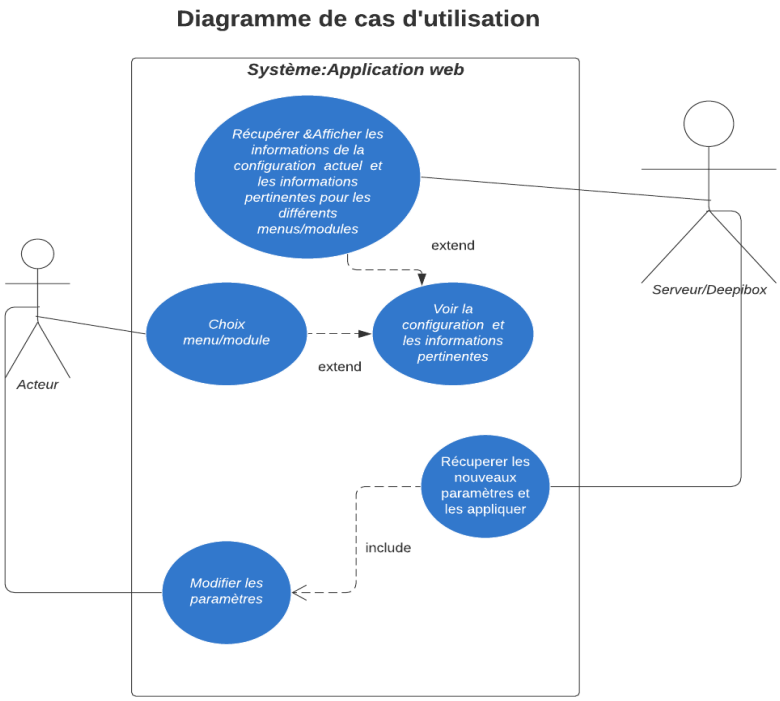


Figure 20 Diagramme de cas d'utilisation de l'application web

I. Phase de recherche : Les outils, et les langages mis à ma disposition

Récapitulatif :

Langages et outils utilisés : ***Gnome-terminal*** (interpréteur/terminal/console de commande Linux de Ubuntu Gnome système d'exploitation de le **Deepibox**), ***Javascript*** (langage de programmation), ***Xrandr*** (outil en ligne de commande de gestion de paramètres d'affichage) et ***Network Manager*** (l'outil de gestion des connexions réseau d'Ubuntu).

Environnement d'exécution (logiciel responsable de l'exécution des programmes) **:** ***Node.js du côté serveur***

Un navigateur web du côté client

Le choix du logiciel IDE (environnement de développement : ensemble d'outils qui permet d'augmenter la productivité) est libre. Il doit être compatible avec **Node.js et le langage Javascript.**
J'ai choisi d'utiliser ***Visual studio code*** par habitude et connaissance de celui-ci (logiciel gratuit et simple d'utilisation, debugger inclus, coloration syntaxique, mise en forme du code automatique, nombreux plugins).

Aucunes connaissances de base sur les outils ***Xrandr*** et ***Network Manager*** (en ligne de commande) étant nécessaires au projet de l'application web. J'ai acquis les connaissances et compétences par le biais de documentations, forums, sites webs et tests des différentes fonctions.

Maîtrise de diverses connaissances et compétences de base acquises à l0'IUT informatique sur l'interpréteur de commande Linux, sur Node.js, et sur Javascript. Approfondissement des connaissances et compétences grâce aux documentations, différents sites webs (e-learnig tels que openClassroom, w3school, developpeur Mozilla…) et forums. Découverte des promesses et callback Javascript (voir glossaire)

Maîtrises des connaissances et compétences acquises notamment à l'IUT sur les notions de réseau et de sa configuration.

1. Gnome-terminal

Il s’agit d’un programme qui émule une console (interface textuelle) dans une interface graphique. Il permet de lancer des commandes et de communiquer avec la machine en langage Shell (Linux et non Windows)

Une commande est une instruction qu'un utilisateur envoie au système d'exploitation de son ordinateur pour lui faire exécuter une tâche.

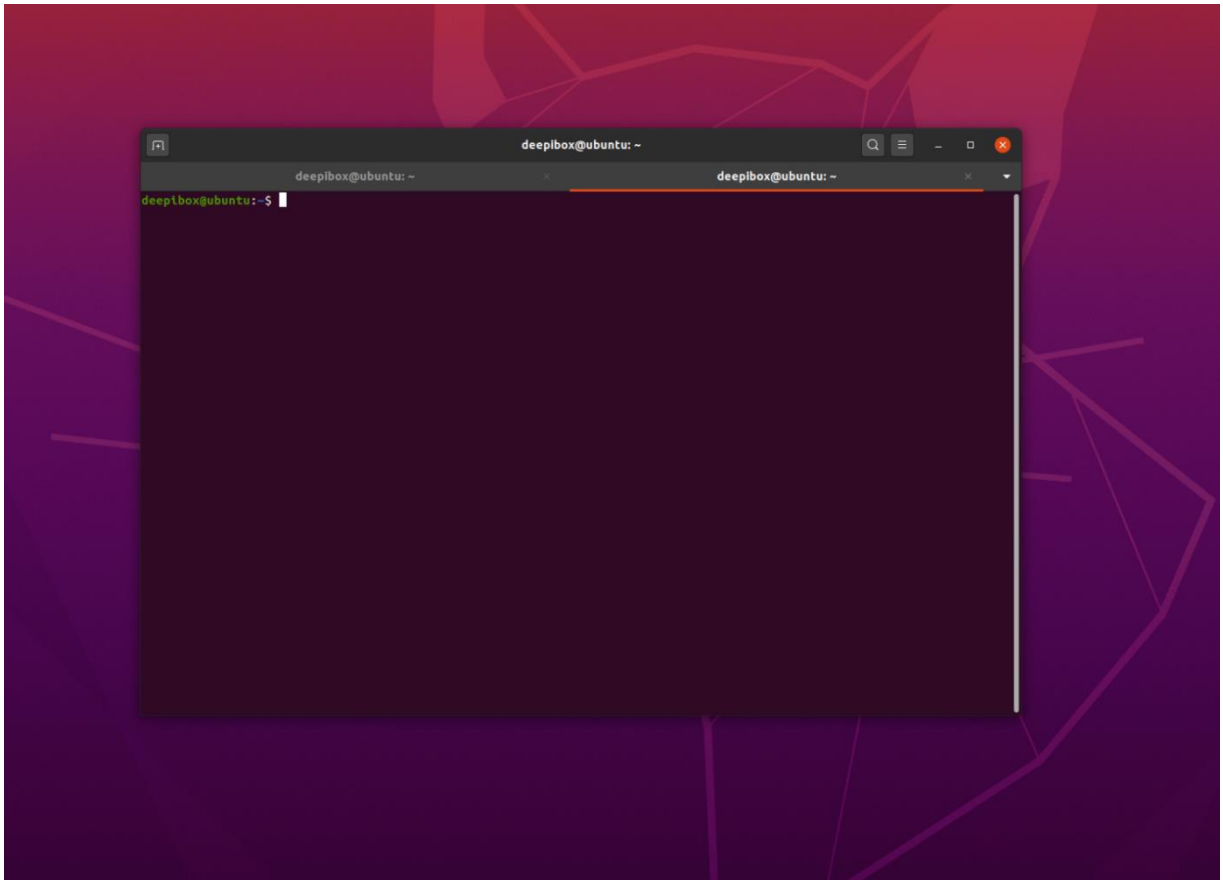


Figure 21 Gnome-terminal

2. Network Manager

Network-Manager est l'outil de gestion des connexions réseau d'Ubuntu. Son utilité est la création et la configuration des accès à divers types de réseaux et de nombreux types de connexions.

Network-Manager est installé par défaut dans Ubuntu. Il prend graphiquement la forme d'une applet, une petite icône située dans le tableau de bord :

Aucun réseau connecté			
Réseau filaire connecté		Ou	
Réseau sans fil connecté			

Figure 22 interface graphique menu applet Network Manager

(Les différentes notions Réseau sont expliquées dans le glossaire technique).

Pour se connecter à un réseau, il faut configurer une interface réseau autrement dit une carte réseau (matériel physique dans un ordinateur). Pour cela Il faut renseigner pour une connexion à un réseau Wi-Fi (sans fil) son identifiant (SSID) et son mot de passe s'il y en a un. Pour une connexion Ethernet (avec fil) on n'en a pas besoin.
Cependant parfois cela ne suffit pas cela dépend du protocole d'adressage (de configuration).
Le protocole d'adressage ou de configuration peut être statique (qui veut dire manuel) ou dynamique (DHCP).
Si le protocole de configuration est manuel il faut renseigner en plus la configuration IP (adresse IP, masque du réseau, passerelle du réseau, un/des serveurs DNS).
Le protocole DHCP permet d'obtenir dynamiquement sa configuration réseau (c'est-à-dire sans intervention particulière) par un serveur du réseau

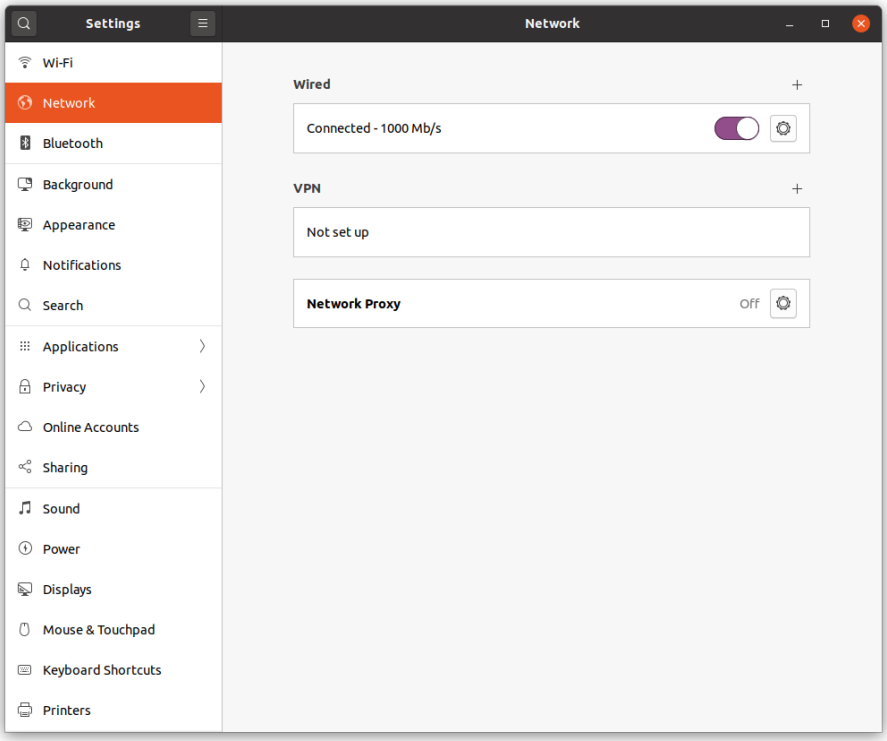


Figure 23 Ubuntu interface de Paramètres Ethernet

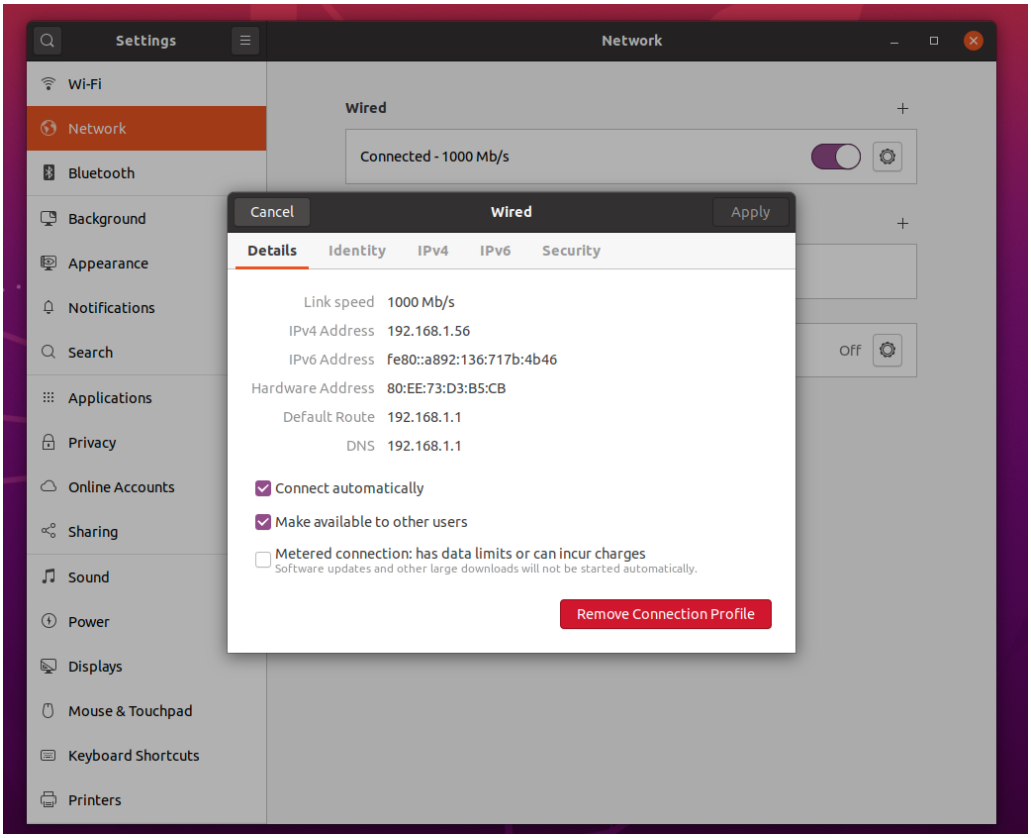


Figure 24 Ubuntu interface générique de paramètres détaillés d'une connexion Ethernet ou Wi-Fi

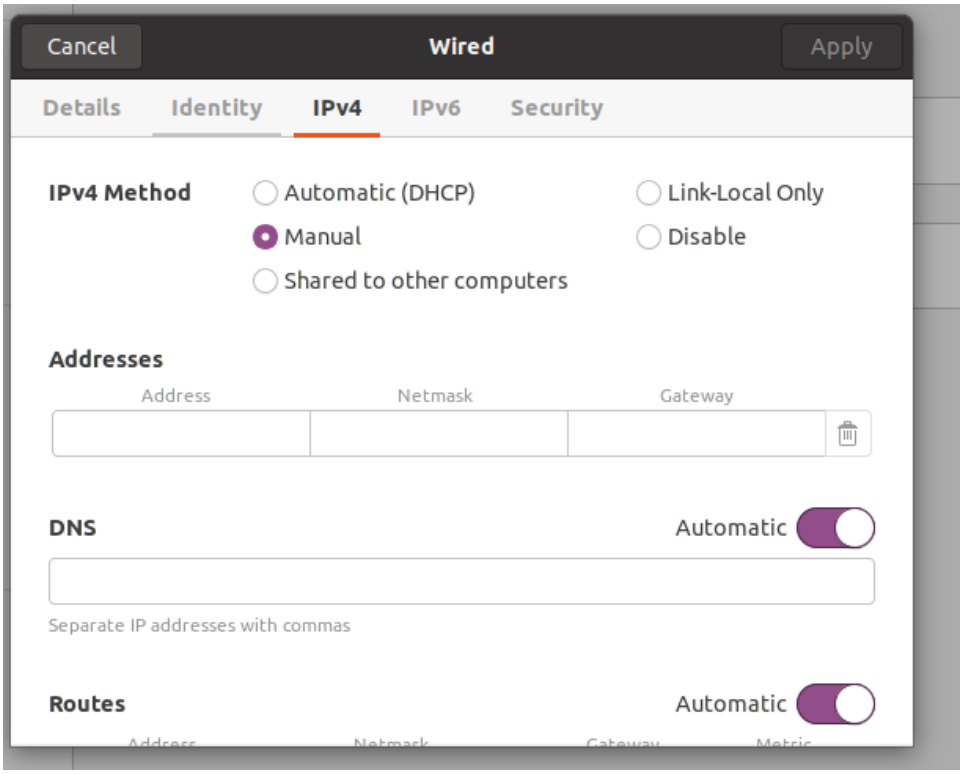


Figure 25 Ubuntu interface générique d'une configuration réseau Ethernet ou Wi-Fi

Network-Manager est fourni avec un petit utilitaire (**nmtui**) permettant de gérer les connexions dans un environnement pseudo graphique. Il permet d'effectuer les mêmes actions que via les différentes interfaces graphiques.

Lancement de l'utilitaire à partir d'une console avec la commande : **nmtui** (Text User Interface for controlling NetworkManager)

Naviguer dans l'application (au clavier car pseudo graphique la souris ne marche donc pas) :

Les touches de direction permettent de naviguer dans les options.

La touche « entrée » permet de valider.

La touche « echap » permet de revenir en arrière sur les écrans.

La touche « espace » permet de cocher ou décocher les cases à choix.

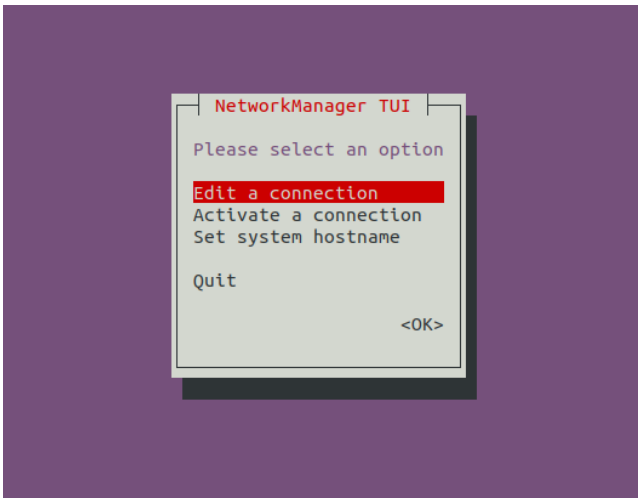


Figure NetworkManager TUI

Les captures d'écran suivantes vous donnent un aperçu des principaux écrans disponibles. Pour configurer les différentes connexions disponibles les options à renseigner sont les mêmes, seule la disposition change.

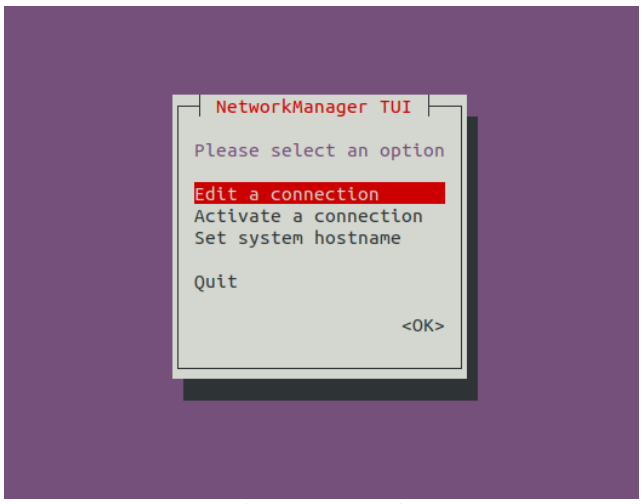


Figure 26 NetworkManager TUI choix option edit connection

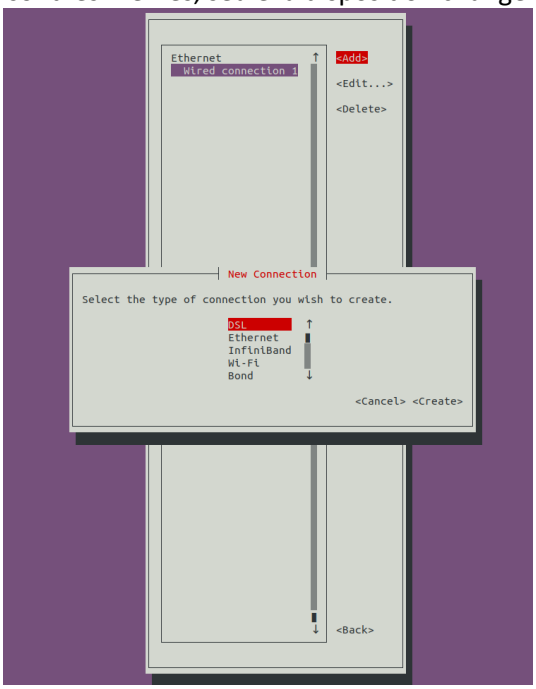


Figure 27 Network Manager TUI add connection

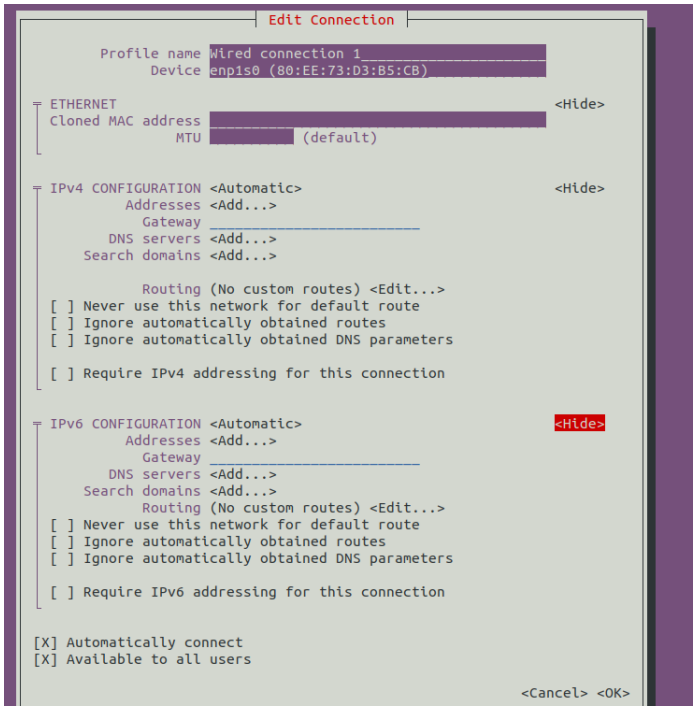


Figure 28 Network Manager TUI edit connection interface.

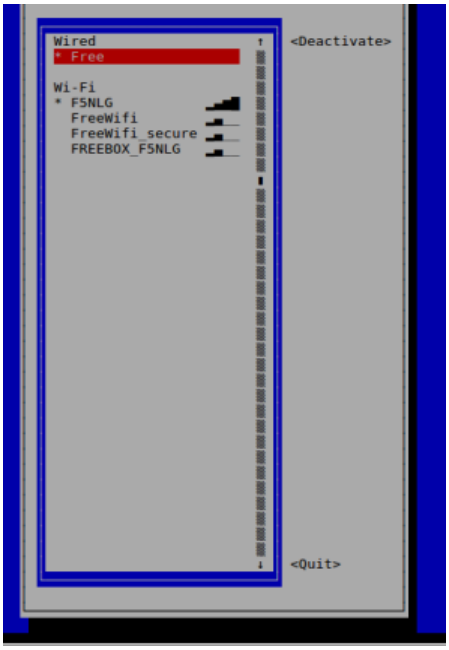


Figure 29 NetworkManager TUI interface activate or desactivate connection

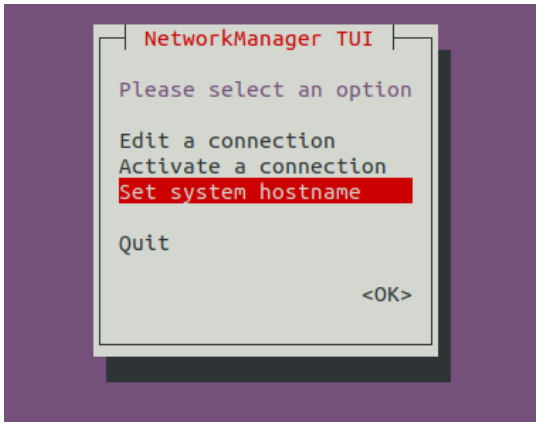


Figure 30 NetworkManager TUI choix option set hostname

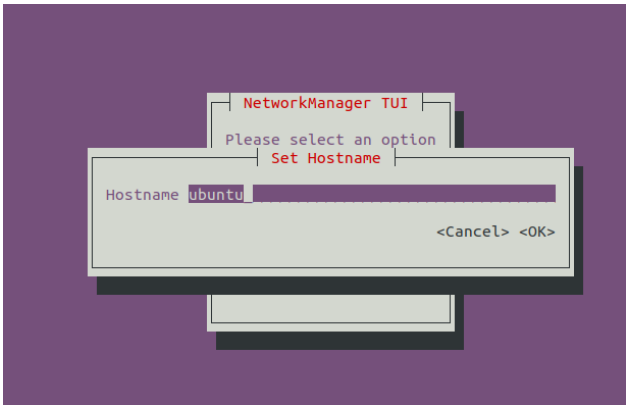


Figure 31 NetworkManager TUI interface set system hostname.

L'outil de ligne de commandes du NetworkManager, **nmcli**, est l'outil de ligne de commande sans interface graphique. Il est installé avec le NetworkManager par défaut et permet de contrôler celui-ci.

Il s'utilise depuis une console/terminal avec la commande : **nmcli** (command-line tool for controlling NetworkManager)

Il permet d'effectuer les mêmes actions que via les différentes interfaces graphiques.

Voici quelques options utiles :

-f permet de spécifier les champs d'une commande qui nous intéresse afin de limiter la sortie (réponse sur le terminal)

-t modifie l'affichage de la sortie (de la console/terminal) d'une commande donnée afin que celui-ci soit adapter pour le traitement informatique (script)

-a qui permet de voir que la connexion active

Voici quelques commandes utiles :

- nmcli con show liste de toutes les connections configurées via NetworkManager.
- nmcli dev show liste de toutes les interfaces ainsi que les détails liés aux interfaces.
- nmcli dev wifi list permet d'obtenir la liste des réseaux Wi-Fi par un scan de l'interface wifi
- nmcli dev wifi connect SSID ''identifiant de la connexion '' password ''mot de passe de la connexion'' (hidden yes/no) permet d'attribuer à une interface Wi-Fi un réseau Wi-Fi et de si connecter. hidden permet de spécifier si il s'agit d'un réseau Wi-Fi caché ou non.
- nmcli con show uuid "UUID de la connexion" ou nmcli con show id "ID de la connexion" permettent d'obtenir tous les détails liés à une connexion en particulier.
- nmcli con down id "nom de la connexion" désactive la connexion.
- nmcli con up id "nom de la connexion" active la connexion.
- nmcli con delete id "nom de la connexion" supprime la configuration et la connexion.

Voici quelques exemples d'illustration :

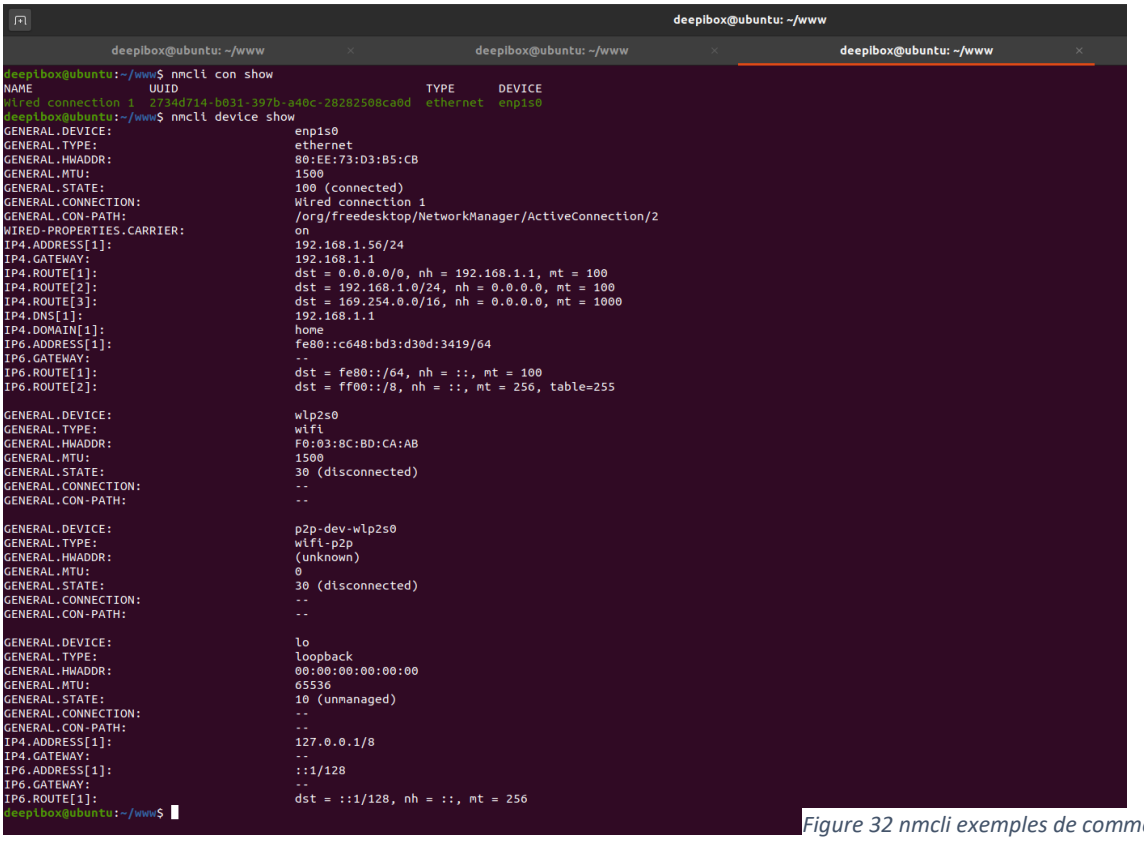


Figure 32 nmcli exemples de commandes

3. Xrandr

Xrandr est un outil en ligne de commande (sur un terminal) afin de gérer les paramètres d'affichage des écrans comme la taille, le redimensionnement, la rotation ou le multi-écran.

Il ne nécessite pas de redémarrage lorsque l'on souhaite par exemple changer la résolution, brancher un deuxième écran ou en utiliser plusieurs. Cependant au redémarrage, ces modifications seront perdues. Afin de rendre persistantes ces modifications, il faut créer un script et le placer dans les fichiers de démarrage afin que celui-ci se lance après un redémarrage.

Pour lister les résolutions disponibles depuis un terminal, il suffit de lancer la commande suivante : **xrandr**

Vous obtenez une réponse, variable suivant votre matériel et la configuration de votre serveur mais la réponse sera du type :

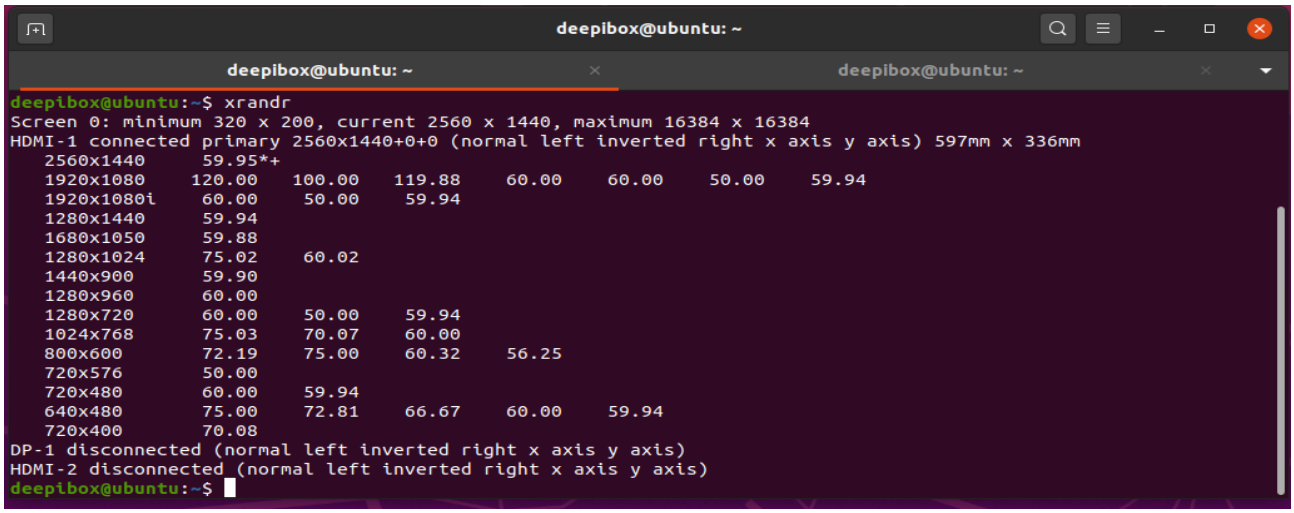


Figure 33 Commande Xrandr

La première ligne indique la résolution minimal, courante, et maximum que votre écran peut supporter.
Les noms des sorties vidéos ou des écrans apparaissent en majuscule et leur statut suit. Il existe 6 sorties vidéo possibles : sortie VGA, sortie DVI, sortie HDMI, sortie DP (display port), sortie LVDS pour la sortie principale d'un portable et sortie TV pour S-video.
Vous voyez apparaître les noms des modes écrans (résolution) disponibles en première colonne et les différentes fréquences de rafraîchissement disponibles en second colonne. En fonction des écrans et pour certaines résolutions, il peut y avoir plusieurs fréquences de rafraichissement disponibles se sont les autres colonnes.
Les modes activés sont directement suivis du signe *.
Les sorties vidéos ou les écrans utilisables sont marquées « connected », leurs noms seront utilisés avec l'option « --output » dans la commande « xrandr ». Exemple : xrandr --output « nom sortie » --mode « choix résolution » --rate « choix rafraichissement »
Si une seule sortie vidéo ou écran est utilisée alors pour changer la résolution il existe une commande plus simple : xrandr -s « choix résolution » -r (équivalent à --rate) « choix rafraichissement » (optionnel)

Ainsi, nous pouvons en déduire l'effet des paramètres :
--output détermine l'écran à configurer
--mode détermine le mode utilisé (optionnel) (abréviation -m)
--rate la fréquence de l'écran, optionnel : par défaut c'est la plus grande valeur qui est appliquée.
(Abréviation -r peut être remplacer par _ « fréquence choisie »)

Il existe d'autres paramètres ou mots clés afin d'obtenir une liste de propriétés plus complète**,** de gérer plusieurs écrans en même temps ou de créer et d'ajouter des modes non existants, de pivoter un écran mais n'est pas utile dans ce projet pour le moment, il s'agit de fonctionnalités supplémentaires à implémenter plus tard.

4. JavaScript

Figure 34 logo JavaScript

Définition d'un script : Un script est un ensemble d'instructions données sous forme de codes. Les instructions sont conçues soit pour le navigateur Web (script côté client), soit pour le serveur (script côté serveur). Les scripts fournissent des modifications à une page Web.

Histoire de JavaScript Annexe1

Javascript est la troisième couche des technologies standards du web, les deux premières sont l'HTML et le CSS.

JavaScript peut interagir avec les éléments HTML DOM et contrôler dynamiquement la page Web.

Exemple de HTML et de CSS avec JavaScript :

```
document.getElementById("demo").style.fontSize = "35px";
var btn = document.createElement("BUTTON");
```

Le DOM (Document Object Model) qui est une interface de programmation ou environnement d'exécution pour les documents HTML, XML, et SVG. LE DOM fournit une représentation structurée du document sous forme d'un arbre et définit la façon dont la structure peut être manipulée par les programmes, en termes de style et de contenu. Le DOM représente le document comme un ensemble de nœuds et d'objets possédant des propriétés et des méthodes. Les nœuds peuvent également avoir des gestionnaires d'événements qui se déclenchent lorsqu'un événement se produit. Cela permet de manipuler des pages web grâce à des scripts et/ou des langages de programmation.

JavaScript est un langage de script côté client basé sur des objets, très populaire et utilisé pour créer des pages Web dynamiques et interactives. JavaScript peut être utilisé pour charger des données asynchrones (faire plusieurs requêtes en même temps sans attendre la réponse de la requête précédente) sans actualiser la page Web.

Les langages orientés objet (OO) sont généralement reconnus grâce à leur utilisation de classes pour créer divers objets qui ont des propriétés et des méthodes similaires.

Il est à noter que, JavaScript au démarrage avant la version ES6(ECMAScript est un ensemble de normes concernant les langages de programmation de type script) n'a pas de concept de classes, et donc les objets sont différents de ceux des langages basés sur les classes.
La notion d'objet en JavaScript est assimilable à une structure de données appelée Dictionnaire ou Hashmap dans d'autres langages. (Tableau avec des clés et des valeurs)

```
var person = {
name: "Karlos",
age: 23,
job: "Network Engineer",
say_Name: function(){
alert(this.name);
}
};
```

Figure 35 Exemple langage orienté objet

JavaScript est aussi un langage orienté objet, un langage de programmation peut être appelé orienté objet lorsqu'il fournit aux programmeurs au moins quatre capacités de base pour développer :

- **Encapsulation** : C'est la capacité de stocker des informations connexes, qu'il s'agisse de données ou de méthodes, mutuellement dans un seul objet.
- **Agrégation** : c'est la capacité de stocker un objet dans un autre.
- **Héritage** : Une classe peut dépendre d'une autre classe ou d'un autre nombre de classes et hériter de leurs variables et méthodes pour une utilisation spécifique.
- **Polymorphisme** : C'est le potentiel du concept de POO (programmation orienté objet) pour écrire une même fonction ou une méthode qui fonctionne de différentes manières.

Les objets sont composés d'attributs, et lorsqu'un attribut contient une fonction, il est considéré comme une méthode de l'objet autrement ; l'attribut est considéré comme une propriété.

Les propriétés d'objet sont des variables qui sont utilisées dans les méthodes de l'objet, mais peuvent également être des variables globalement visibles qui sont utilisées dans toute la page.

Avantages de JavaScript :

Rapidité : étant un langage de script côté client, JavaScript est très rapide car toutes ses fonctions de code s'exécutent immédiatement sur la machine cliente au lieu de contacter le serveur et d'attendre une réponse.

Simplicité : JavaScript est relativement facile à apprendre et à coder.

Polyvalence : JavaScript fonctionne bien avec d'autres langages et est également utilisé dans une grande variété d'applications.

Charge du serveur : étant du côté client, cela réduit l'exigence sur le serveur du site Web.

Le client est la structure ou le système sur lequel s'exécute le navigateur Web. JavaScript est le principal langage de script côté client pour le Web. Les scripts côté client sont interprétés par le navigateur. Le serveur est l'endroit où résident la page Web et les autres contenus. Le serveur envoie des pages à l'utilisateur / client sur demande. Ce langage de programmation unique facilite cette combinaison de gestion client-serveur.

JavaScript est à la fois un langage très simple mais en même temps très complexe car il prend des minutes à apprendre mais des années à maîtriser.

Le Format JSON et AJAX (voir glossaire) :

Les promesses, callback Javascript (voir glossaire) :

5. NodeJS

Est un environnement d'exécution pour le langage Javascript asynchrone et orienté évènement, permet du faire du web sans navigateur, et permet donc des fonctionnalités supplémentaires que ceux-ci restreignent tels que l'accès aux processus système (voir glossaire) et la modification de la configuration de celui-ci.

Figure 36 logo NodeJS

NodeJs utile le moteur V8 de chrome. Le V8 est le moteur JavaScript et WebAssembly open source haute performance de Google. Il s'agit d'un interpréteur du langage Javascript c'est ce qui va interpréter le code et le transformer en bytecode,lui-même compilé en langage machine.

Différence entre langage compilé et interprété : Un langage est dit compilé quand son implémentation exécutable requiert un compilateur. De la même manière, un langage interprété requiert un interprète.

La différence centrale entre compilé et interprété est comme suit : là où le compilateur traduit une bonne fois pour toute un code source en un fichier indépendant exécutable (donc utilisant du langage/code machine), l'interprète est nécessaire à chaque lancement du programme interprété, pour traduire au fur et à mesure le code source en code machine.

NodeJs est extensible par des modules qui sont des librairies de fonction.

NodeJS permet donc de faire du web côté serveur grâce au module natifs de NodeJS : http.

Le module http permet le développement de serveur HTTP ou web (voir glossaire). Il est donc possible de se passer de serveurs web tels Apache lors du déploiement de sites et d'applications web développés avec Node.js.

NodeJS va nous permettre de créer un serveur web embarqué dans la Deepibox qui va stocker notre application web de configuration.

Afin de communiquer et d'envoyer des données avec Internet à un réseau plus précisément à un ordinateur, il faut renseigner le port de service qui correspond aux différents protocoles de services d'Internet (Port 80 : service HTTP) afin que l'ordinateur recevant, peut savoir à quel programme les données qui arrivent sont destinées.

Les numéros de Port possibles vont de 0 à 65535. Cependant, ce secteur est subdivisé en trois catégories :

-0 jusqu´à 1023 sont des numéros réservés pour des services spéciaux comme FTP (transfert de fichier) (21), SMTP (25) (mails), HTTP (80) (web), POP3 (110), DNS (53) …

-1024 jusqu´à 49151 sont enregistrés pour certains services.

-49152 jusqu´à 65535 sont des numéros que chacun peut utiliser.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Figure 37 Exemple de serveur Node utilisant l'adresse de loopback

Adresse de loopback (127.0.0.1 ou localhost) : l'adresse réseau réservée et utilisée par le système local pour permettre les communications entre processus.

Ici notre application ne va pas utiliser l'adresse de loopback sinon elle ne pourra pas communiquer l'extérieur et donc avec d'autres ordinateurs, il faudra renseigner dans le hostname l'adresse IP actuel de la Deepibox.

Un serveur Apache du côté client écoutant sur le port : 80 est ici utilisé comme proxy (passerelle entre Internet et le réseau local privé) pour notre application laquelle écoute sur le port : 8080

Plusieurs connexions peuvent être gérées de manière concurrente. À chaque connexion, la fonction de rappel (callback function voir glossaire) est déclenchée, mais s'il n'y a rien à faire, Node.js (le serveur Node) restera inactif asynchrone car le fichier n'attend pas que le processus de création d'entrée/sortie de création du serveur soit terminer. Il s'agit de la Capacité de la boucle d'événements à exécuter des fonctions de rappel JavaScript après avoir terminé d'autres travaux. Tout code censé s'exécuter de manière simultanée doit permettre à la boucle d'événements de continuer à s'exécuter pendant que des opérations non JavaScript, telles que les Entrée / Sortie, se produisent.

II. La conception de l'application web en général

Lors de la phase de conception de l'application, il faut penser qui va utiliser l'application surtout si l'entreprise dans laquelle on se trouve est présente à l'international afin de définir la langue de l'application aussi bien dans le code que sur le visuel de l'application Deepidoo, présent à l'international, a donc choisi l'anglais.

Il faut aussi penser à respecter la charte graphique de l'entreprise si elle en possède une.

Rappel : la charte graphique présente l'ensemble des éléments contenus dans l'identité visuelle d'une entreprise ou d'une marque. Elle est l'un des éléments fondamentaux de la communication de l'entreprise puisqu'elle a pour objectif de permettre l'identification immédiate de l'entreprise dans tous les supports de diffusion qu'elle utilise, tout en assurant une cohérence dans ses supports de communication.
Quatre éléments sont détaillés dans la charte graphique : le logo, la typographie (police d'écriture), les codes couleurs, et les éléments graphiques liée à l'entreprise et au logo.

J'ai dû créer une application web permettant de modifier divers paramètres tels que la configuration réseau, la résolution de la/les sortie vidéo (écran), mais encore le Bluetooth ou le volume audio que je décompose en sous parties que j'appelle menus ou modules car il s'agit de choses distinctes se retrouvant dans la même application Pour le moment les 2 menus configurationRéseau et configurationRésolutionEcran sont réaliser et presque totalement fonctionnels.

Le choix d'utiliser qu'un seul langage de programmation le Javascript pour l'application web était obligatoire à la réalisation de l'application et permet aussi de faire du HTML et du CSS (qui sont les langages de programmation du fond et de la forme du web). J'ai découvert la possibilité de faire du HTML et CSS en JavaScript ainsi que l'utilisation du DOM.

Nous avons un côté serveur Node où nous lançons les commandes systèmes. Nous récupérons les informations sur le terminal, que nous mettons sous forme d'objet ou tableau javascript (parsage) pour les transmettre au serveur (par la méthode des promesses javascript découvert). Celui-ci les convertis en fichier avec une fonction existantes (JSON.stringify) au format qui nous intéresse (ici JSON : JavaScriptObjectNotation) afin de les transmettre au côté client, qui va demander au serveur les informations/paramètres de la configuration actuelle et les afficher dans un formulaire. Cela lui permettra de saisir des nouvelles données afin de changer la configuration en envoyant ses données au serveur au format JSON que celui-ci va retransformer en valeur afin d'effectuer les requêtes de modifications avec la fonction existante (JSON.parse).

J'ai choisi la création d'un formulaire maison pour saisir les différentes informations nécessaires à la configuration et de ne pas d'utiliser la balise du langage HTML <form> en Javascript. J'ai créé mes propres boutons avec des évènements plus pertinents et correspondant plus au besoin de l'application (plusieurs boutons « submit » pour le module de la configuration réseau et autres ; ainsi que d'autres boutons non spécifiques à un formulaire.)

Lors de la création et du développement d'une application il faut aussi penser à l'utilisateur : il faut le tenir au courant de ce que l'application fait par exemple si elle est en train de charger, d'attendre des données ou autre.

Lorsque l'on a un formulaire que l'utilisateur doit remplir des champs et saisir diverses informations, celui-ci peut faire des erreurs de saisie il faut donc l'en informer.

Le formulaire maison est sous forme de table avec en en-tête le nom du module, table avec plusieurs lignes reprenant les différentes interfaces (Ethernet et Wi-Fi pour la configuration réseau et les différentes sorties vidéo pour la résolution d'écran). Plusieurs colonnes reprennent les champs des différentes informations nécessaires à chaque module (pour le réseau : activer ou non, DHCP ou non, l'adresse IP, le masque du réseau, la passerelle au prochain réseau, le/les serveurs DNS (domain name system), pour la résolution d'écran : la liste des différentes résolutions possibles).

Il y a au total 5 fichiers de travail, un dossier créé automatiquement par Node comportant les différent modules de Node souhaités, installés, et utilisés, ainsi que des fichiers package.JSON et package-lock.JSON reprenant les différentes dépendances entre les fichiers créés aussi par Node.

Les fichiers de travail sont : le **fichier serv.js** qui correspond au serveur Node qui fait le lien entre la partie serveur système et le client, le **fichier infosConfig.js** permettant de récupérer les différentes informations de configuration pour chaque module et de les parser, **le fichier index.html** qui est le fichier de départ de chaque application ou site web ici il lance juste le fichier script App.js, **le fichier App.js** qui créé donc le formulaire en appelant une fonction que j'ai développé permettant de simplifier la création des balises HTML (qui représente des éléments tels que : paragraphe, champs de saisie, case à cocher , etc). Cette fonction se situe dans un **fichier** nommé **util.js**.

Rappel définition d'un script par le site journal du net : En informatique, un script désigne un programme chargé d'exécuter une action prédéfinie quand un utilisateur réalise une action ou qu'une page web est en cours d'affichage sur un écran. Il s'agit d'une suite de commandes simples et souvent peu structurées qui permettent l'automatisation de certaines tâches successives dans un ordre donné.

1ère version de l'application web :

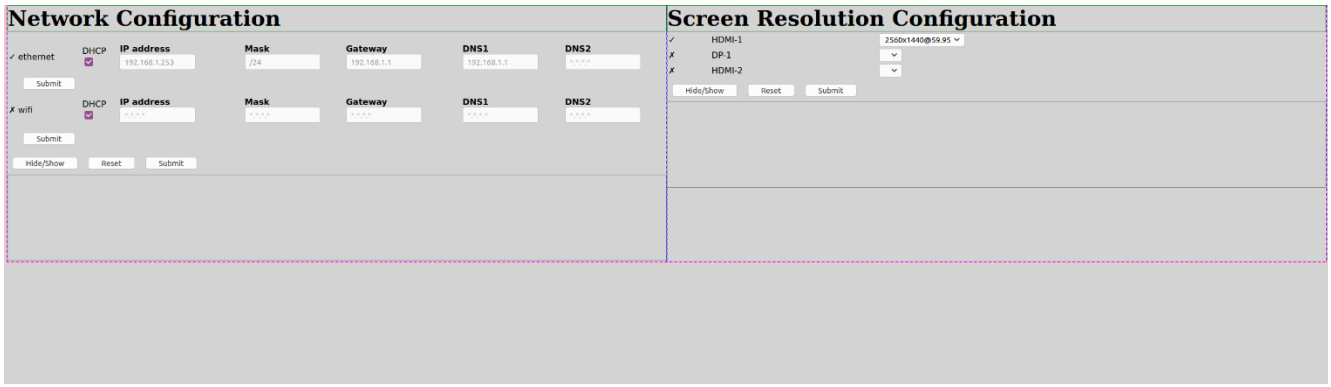


Figure 38 1ère version réalisée de l'application web

Généralement lorsque l'on commence à développer, on ne pense pas forcément à utiliser des noms très explicites pour des variables, objets, méthodes et fonctions, tout se retrouve dans un même fichier ce qui ne facilite pas la lisibilité et la compréhension du code on se retrouve donc parfois un peu perdu et on ne sait plus qui fait quoi.

Ainsi, le code source rédigé est souvent retravaillé, modifié et découpé. C'est ce qu'on appelle du refactoring.

Le but du refactoring du code est purement et simplement de produire un meilleur code. Un code efficace permet de mieux intégrer de nouveaux éléments, sans générer de nouvelles erreurs. Si la lecture du code ne leur demande pas d'effort particulier, les programmeurs peuvent corriger ou éviter les bugs plus facilement. Le refactoring a également pour objectif de simplifier l'analyse des erreurs et la maintenance d'un logiciel.

Ici le fichier App.js contenait le fond et la forme pour chacun de mes modules ce qui ne suivait pas ma logique de réflexion et il fallait encore rajouter les divers évènements propres à chaque module.

Le refactoring est donc une étape importante du développement et doit être effectué plusieurs fois au cours du développement afin de faciliter la programmation, la lisibilité, la compréhension et la maintenance du code

Voici à quoi ressemble l'application après le 1er et 2ème refactoring, et la décision d'ajout des 2 modules supplémentaires qui sont le Bluetooth et le volume du player audio ainsi qu'après la création de la barre de menu afin de choisir le module qu'on veut configurer

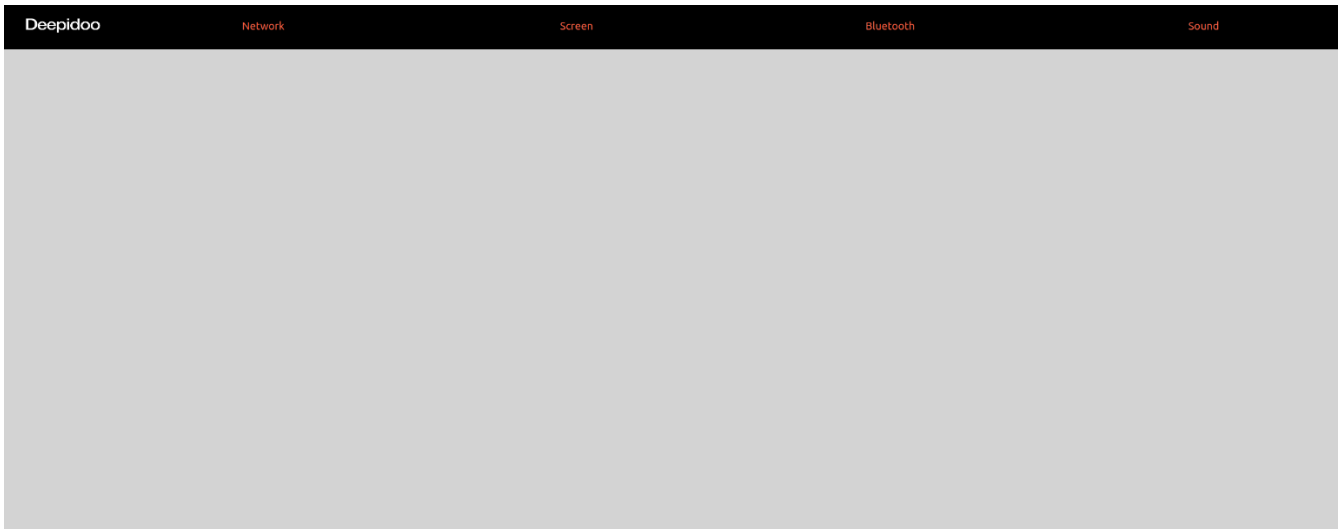


Figure 39 L'application web après refactoring et la décision d'ajout de 2 menu supplémentaire de configuration

La liste des fichiers après les premiers refactoring, m'a permis d'avancer et de faire le début de l'implémentation des mises à jour de la configuration en fonctions des modules lors de l'évènement du « clique » sur le bouton d'envoi :

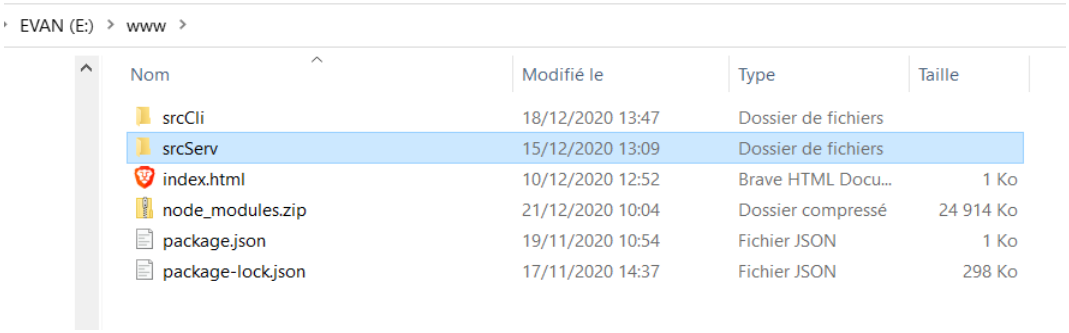


Figure 40 Liste fichiers et dossiers

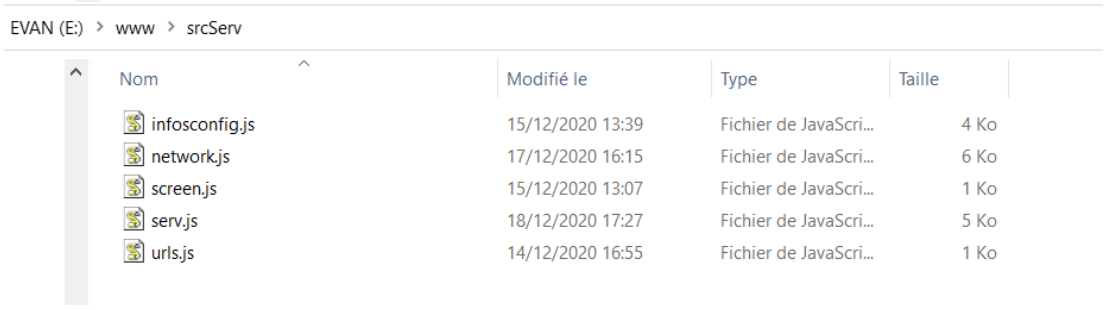


Figure 40 Liste des fichiers côté serveur

Le fichier infosconfig.js est constitué de 3 fonctions/méthodes, chaque fonction permet la récupération des données souhaitées (par la commande shell nécessaire de l'outil Xrandr ou par la commande nmcli concerné du Network Manager), le parsage de ces données et la transmission de celles-ci (par les promesses).

Ces fonctions concernent :
-la liste des réseau Wi-Fi
- la liste les résolutions disponibles pour les sorties vidéos
-la configuration actuelle du réseau des 2 interfaces (carte réseau : Ethernet et Wi-Fi) de la Deepibox.

On peut constater ici que ce fichier regroupe les méthodes de récupération des données des deux modules (Configuration réseau et Configuration de la résolution d'écran) que j'ai décidé de ne pas mélanger avec les méthodes de mise à jour de configuration celles-ci n'étant pas encore totalement terminées et testées dans leur fichier respectif (network.js et screen.js). Infosconfig disparaitra donc au prochain refactoring et les fonctions de récupération rejoindront leur fichier respectif.

Explication des fichiers serv.js et urls.js :

Le fichier serv.js contient la création du serveur Node les différentes adresses web choisi (qui corresponde à la requête du client) avec leur méthode REST (requête HTTP) autorisé dessus (GET pour récupérer, POST pour modifier et envoyer), l'action effectué et la réponse associé du serveur. Req est la requête formuler par l'utilisateur et Res est la réponse que va fournir le serveur (voir glossaire AJAX) EX :

```
var http = require('http');
var fs = require('fs');
var URL = require('url');
const Infosconfig = require('./infosconfig');
const Network = require('./network');
const Screen = require('./screen');
var server = http.createServer(handle_request);
server.listen(8080, '127.0.0.1');

function handle_request(req, res) {
    var parsed = URL.parse(req.url);

    var appmatch = parsed.pathname.match(/app\..*\.js/);
    var appname = appmatch && appmatch.input;

    switch (parsed.pathname) {

        case '/get_screen_config':
            var get_ResolutionList = Infosconfig.getResolutionList();
            get_ResolutionList.then((resolutionList) => {
                res.writeHead(200, { 'Content-Type': 'application/json' });
                res.end(JSON.stringify(resolutionList));
            });
            get_ResolutionList.catch((err) => {
                res.writeHead(500, { 'Content-Type': 'application/json' });
                res.end(JSON.stringify(err));
            });
            break;
            case '/update_screen_config':
                if (req.method == 'POST') {
                    var body = "";
                    req.on('data', (chunk) => { body += chunk.toString(); });
                    req.on('end', () => {
                        const screen_config_params = JSON.parse(body);
                        try {
                            Screen.updateConfig(screen_config_params);
```

Figure 41 Fichier serv.js

Du côté du client dans le fichier app on utilise la fonction fetch(fonction pour récupérer les promesse JS côté client voir glossaire promesse) qui permet de demander de récupérer les informations et qui se traduit par une requête HTTP du côté serveur.

```
import { createElement, createMenuItem, createMenuBar } from './util';
import ScreenTableFormUI from './screenFormUI';
import NetworkTableFormUI from './networkFormUI';
import Button from './button';
import Result from './result';
import URLS from '../srcServ/urls';

let body = document.body;
body.style.margin = '0px';
body.style.padding = '0px';
body.style.backgroundColor = 'lightgrey';
let header = createElement('header', {}, body);
let menu = createMenuBar(['Network', 'Screen', 'Bluetooth', 'Sound'], header);
let globalBox = createElement('div', {
    width: '100%',
    display: 'inline',
    //border :'5px solid pink'
}, body);


const App = {
    scrnForm: undefined,
    netForm:undefined,
    start() {
    var networkMI = createMenuItem(globalBox, 'Network Configuration', 'network');
    //console.time('info ethernet');
    var get_network_config = fetch(URLS.get_network_config);
    get_network_config
        .then(response => response.json())
        .then((dataNetwork) => {
            //console.time('info wifi');
            var get_wifiList = fetch(URLS.get_wifiList_ssids);
            get_wifiList.then(response => response.json()).then((wifiList) => {
                App.netForm = new NetworkTableFormUI(dataNetwork, wifiList, networkMI.content);
                //console.timeEnd('info wifi');
                //console.timeEnd('info ethernet');
            })
```

Figure 42 FIchier app.js

Le fichier urls.js récapitule les adresses choisies dans le fichier serv.js, contient les différentes adresses web personnelles choisies par lesquelles le serveur et le client vont s'échanger les données. Une pour chaque fonction précédemment ci-dessus dans le fichier infosconfig et deux autres pour la fonction de mise à jour de la configuration de chaque module. Ce fichier est placé du côté serveur cependant il est utilisé et appelé que du côté client.

```
const URLS = {
    base_url: "http://localhost:80",

    init() {
        URLS.get_network_config = `${URLS.base_url}/get_network_config`;
    },

    get_network_config: "http://localhost:80/get_network_config",
    update_network_config:"http://localhost:80/update_network_config",
    get_wifiList_ssids:"http://localhost:80/get_wifiList_ssids",
    get_screen_config:"http://localhost:80/get_screen_config",
    update_screen_config:"http://localhost:80/update_screen_config",
};

export default URLS;
```

Figure 43

Fichier urls.js

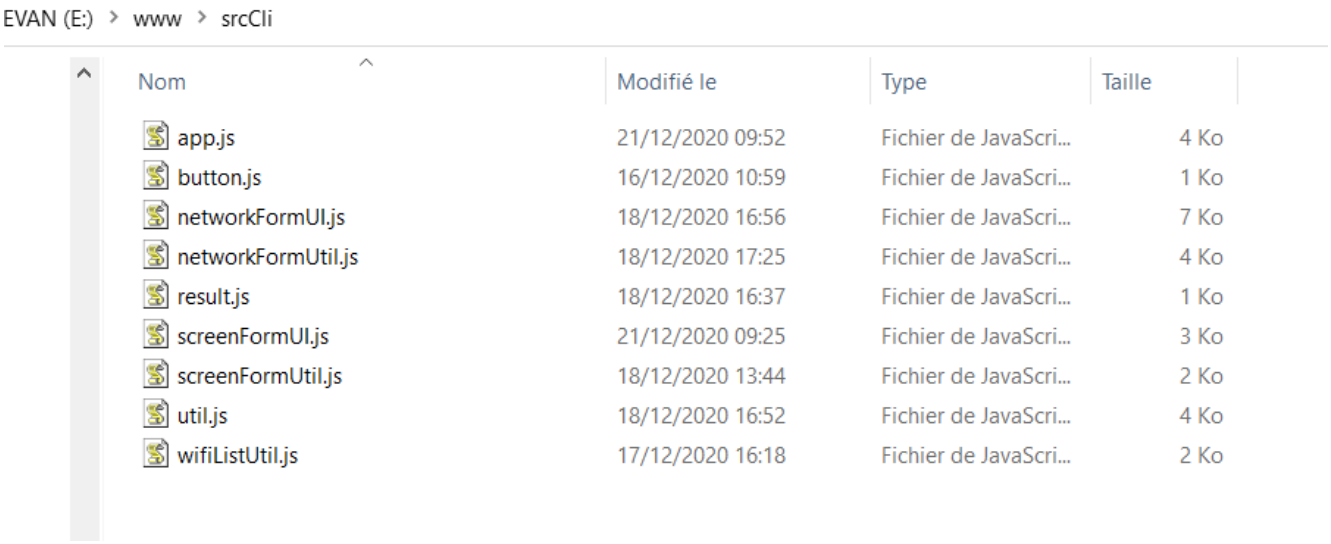


Figure 44 Liste des fichiers côté client

Les fichiers dont le nom sont de la forme « *UI » sont les fichiers des différents modules correspondant au fond et à la forme de l'application web. Ceux dont le nom correspond à « *Util » sont les fichiers des modules qui gère, modifie les données ou applique des évènements en fonction de celles-ci sur le fond et la forme.

Explication des fichiers util, result, button,appUtil et app :

util.js contient différentes fonctions de création d'éléments : la création de la barre de menu, la fonction qui permet de simplifier la création d'éléments html (paragraphe, champs de saisie, case à cocher, etc), une fonction permettant de créer un champ de saisie de mot passe et une fonction évènement qui affiche le menu choisi et cache les autres (afin de respecter le principe d'une application comportant qu'une seule page).

Les fichiers result et button permettent d'instancier des objets qui sont sous forme de classes avec leurs propres méthodes. La classe result crée un objet étant un conteneur (fonction de création d'élément) qui a une méthode s'appelant update prenant en paramètre des données. Cette méthode permet de les afficher. La classe button permet de créer des boutons avec leur évènement correspondant ou des boutons sans évènement qu'on pourra leur fournir plus tard.

appUtil contient seulement une fonction qui définit ce que l'évènement produit lorsque que l'on clique et qui sera transmis au fichier app (qui va le transmettre à son tour au fichier screenFormUI) afin d'instancier ce que le bouton d'envoi, se trouvant dans le module configuration résolution d'écran, doit faire c'est-à-dire récupérer les nouveaux paramètres choisis ou saisis, de les afficher dans l'élément result (en appelant sa fonction update) , ainsi que de les envoyer au serveur (après les avoir changer de format etc).

app est le fichier principal de l'application qui se lance au démarrage de celle-ci. Il initialise les formulaires des modules en appelant leur classe et leur constructeur (avec le mot clé new). Le «. then » permet de dire à l'application d'attendre la réponse de la promesse effectuée du côte serveur qui va lui transmettre une réponse au format JSON.

```
import { createElement, createMenuItem, createMenuBar } from './util';
import ScreenTableFormUI from './screenFormUI';
import NetworkTableFormUI from './networkFormUI';
import AppUtil from './appUtil';
import URLS from '../srcServ/urls';
let body = document.body;
body.style.margin = '0px';
body.style.padding = '0px';
body.style.backgroundColor = 'lightgrey';
let header = createElement('header', {}, body);
let menu = createMenuBar(['Network', 'Screen', 'Bluetooth', 'Sound'], header);
let globalBox = createElement('div', {
    width: '100%',
    display: 'inline',
}, body);
window.onload = function () {
    App.start();
}
const App = {
    start() {
        var networkMI = createMenuItem(globalBox, 'Network Configuration', 'network');
        var get_network_config = fetch(URLS.get_network_config);
        get_network_config
            .then(response => response.json())
            .then((dataNetwork) => {
                var get_wifiList = fetch(URLS.get_wifiList_ssids);
                get_wifiList.then(response => response.json()).then((wifiList) => {
                    App.netForm = new NetworkTableFormUI(dataNetwork, wifiList, networkMI.content);
                })});

        var screenMI = createMenuItem(globalBox, 'Screen Resolution Configuration', 'screen');
        var get_screen_config = fetch(URLS.get_screen_config);
        get_screen_config
            .then(response => response.json())
            .then((resolutionList) => {
                App.scrnForm = new ScreenTableFormUI(resolutionList, screenMI.content);
                App.scrnForm.setOnClick(AppUtil.setOnClick);
```

Figure 45 Fichier app.js

La plus grosse difficulté était de garder des références de chaque objet, de les transmettre à plusieurs autres fichiers afin de modifier leur comportement et leur affichage en fonction des données ou de la manipulation de celles-ci.

L'application au début ne comportait que 2 modules et pas de barre de menu permettant de choisir lequel on voulait modifier. Il fallait charger toutes les informations (des 2 modules) en même temps car les formulaires étaient côte à côte. Ce fonctionnement convenait étant donné que l'application ne consommait peu de ressources et était légère. Maintenant que la forme de l'application a changé avec l'apparition du menu et des 2 modules supplémentaires il faudrait modifier ce fonctionnement pour éviter tout problème et ralentissement.

III. Le module Configuration Résolution d'écran :

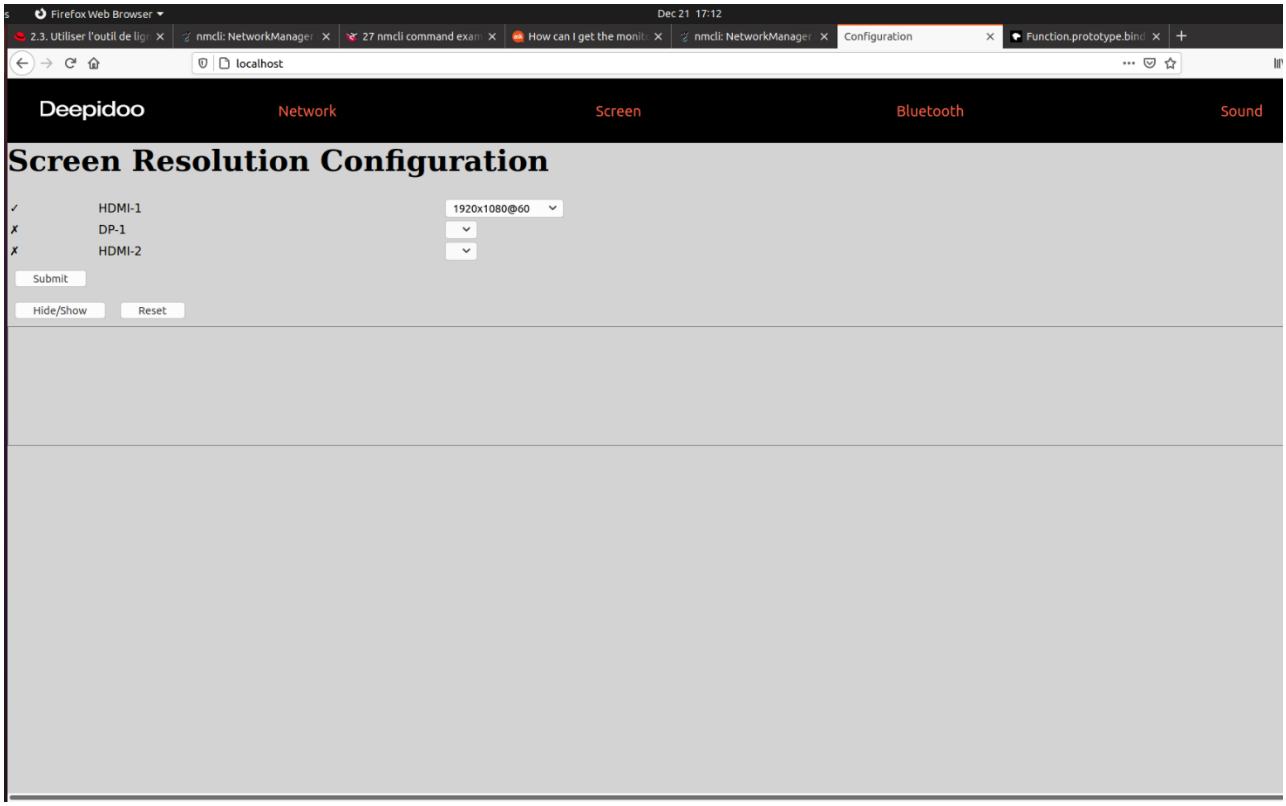


Figure 46 Application web module résolution d'écran

1. Coté serveur

- Utilisation d'une librairie Node Xrandr-parse pour le parsage des données.

Ce parseur que j'ai choisi d'utiliser me permet de récupérer les données et de les mettre en forme sous le format (format JSON voir glossaire) qu'il me fallait automatiquement et m'évitais ainsi de les traiter moi-même.

Gain de temps.

```
const ShellNmcl = require('child_process');
const ShellXrndr = require('child_process').exec;
var xrandrParser = require('../node_modules/xrandr-parse');
const Infosconfig =
{
    init() {

    },
    getResolutionList() {

        return new Promise
            (
                (resolve, reject) => {

                    ShellXrndr('xrandr', function (err, stdout, stderr) {
                        if (err) {
                            reject(err);
                        }
                        else {
                            //parsage
                            var resolutionList = xrandrParser(stdout);
                            resolve(resolutionList);
                        }
                    });
                });
    },
```

Figure 47 Fichier infosconfig.js fonction récupération de la liste des différentes résolutions disponibles

Modification de la configuration :

```
const Shell = require('child_process');

const Screen = {

    init(){

    },
    updateConfig(dataConfig){
        var config =dataConfig[0];
        const modeChoice = config.value.replace('@', ' --rate ');
        const screenInterface =config.name;
        req = 'xrandr --output ' + screenInterface + ' --mode ' + modeChoice;
        Shell.execSync(req);
    },
}
module.exports = Screen;
```

Figure 48 fichier screen.js

Transmission du message d'erreur du terminal à l'utilisateur si rencontre d'un problème et non réalisation de la tâche demander par celui-ci. Sinon transmet le message suivant si la tâche a été réaliser : « result : ok »

```
case '/get_screen_config':
    var get_ResolutionList = Infosconfig.getResolutionList();
    get_ResolutionList.then((resolutionList) => {
        res.writeHead(200, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify(resolutionList));
    });
    get_ResolutionList.catch((err) => {
        res.writeHead(500, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify(err));
    });
    break;
    case '/update_screen_config':
        if (req.method == 'POST') {
            var body = "";
            req.on('data', (chunk) => { body += chunk.toString(); });
            req.on('end', () => {
                const screen_config_params = JSON.parse(body);
                try {
                    Screen.updateConfig(screen_config_params);
                } catch (e) {
                    err = e.message;
                    console.log(e.message);
                    res.writeHead(500, { 'Content-Type': 'application/json' });
                    res.end(JSON.stringify(err));
                }
                res.writeHead(200, { 'ContentType': 'text/plain' });
                res.end('result:ok');
            });
        } else {
            res.end();
        }
        break;
```

Figure 49 Fichier serv.js

2. Côté client

-Retour utilisateur : conteneur/emplacement qui récapitule et affiche ce que l'utilisateur a envoyé comme paramètres de configurations (Cette emplacement m'a aussi été très utiles lors du développement il me permettait de vérifier les données avant de les envoyer au serveur.)

Réponse dans la console du navigateur lorsque la modification de la résolution a été effectuée ou s'il y a une erreur.

Rappel : Pour afficher la console du navigateur faire clique droit dans celui-ci et cliquer sur inspecter élément

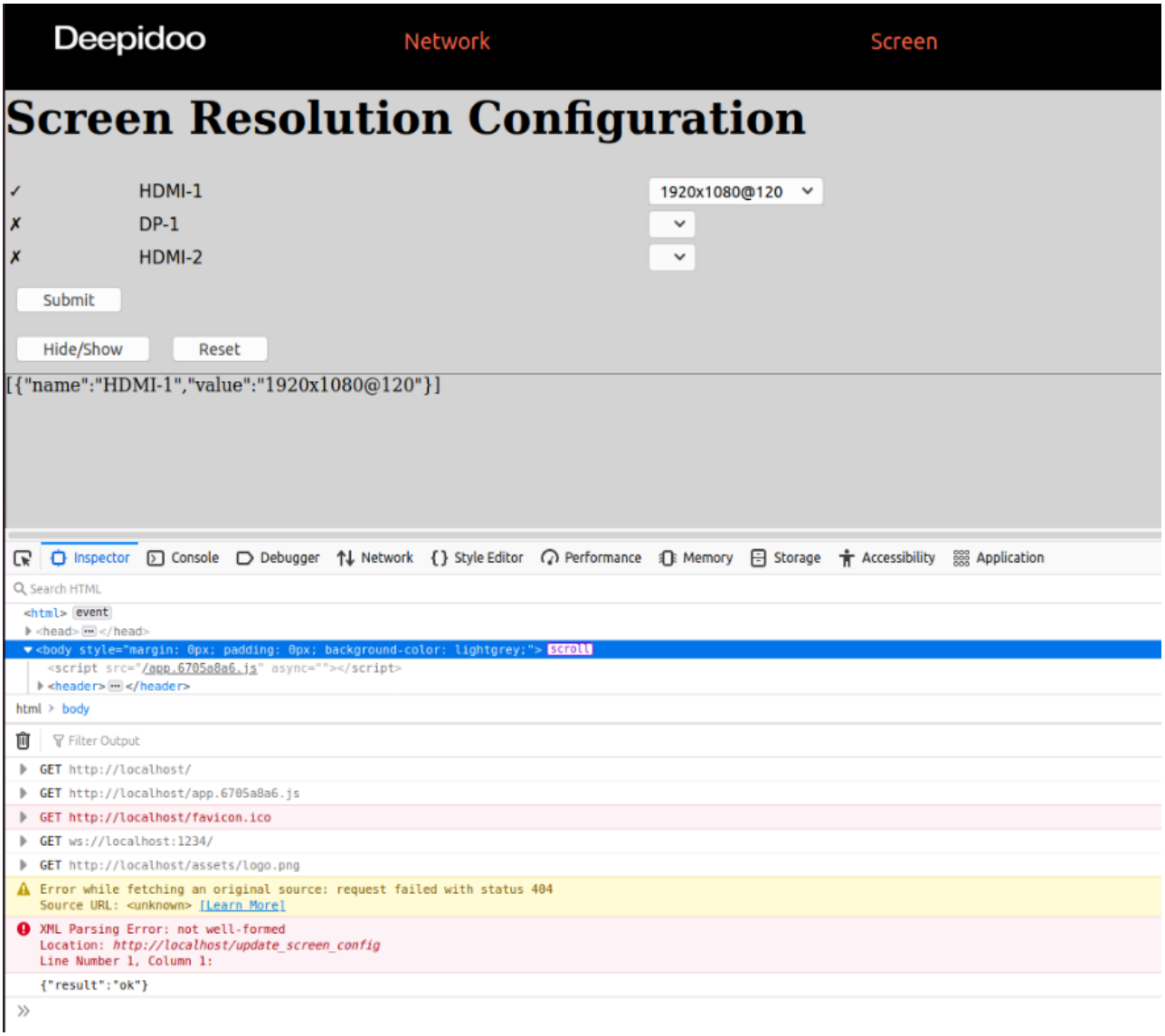


Figure 50 affichage de l'application web après l'envoi de données

IV. Le module Configuration Réseau :

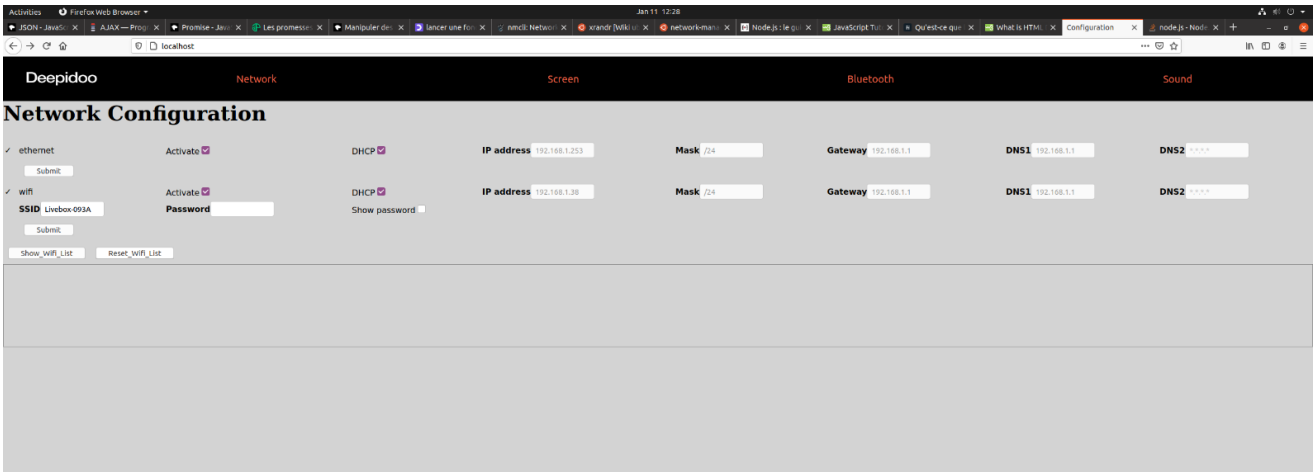


Figure 51 application web module configuration réseau

1. Côté serveur

Dans le fichier infosconfig.js : Mapping des données sur la configuration actuelle pour le parsage.

Mapping : Représentation au format JSON (voir glossaire) des champs, objets avec leur clé et leur valeur de l'ensembles de données en vue d'en faciliter l'accès.

```
const Infosconfig =
{
    init() {
    },
        mapp:
        {
            "GENERAL.CONNECTION": "name",
            "GENERAL.TYPE": "type",
            "GENERAL.DEVICE": "device",
            "GENERAL.STATE": "state",
            //"GENERAL.HWADDR": "mac",
            "IP4.ADDRESS[1]": "ip",
            "IP4.GATEWAY": "gateway",
            "IP4.DNS[1]": "dns1",
            "IP4.DNS[2]": "dns2",
            "IP4.DOMAIN[1]": "area",
            "IP4.ROUTE[1]": "road1",
            "IP4.ROUTE[2]": "road2",
            "IP4.ROUTE[3]": "road3"
        },

    getConfigNetwork() {
        // nmcli -t -f GENERAL.CONNECTION,GENERAL.TYPE,GENERAL.DEVICE,GENERAL.STATE,GENERAL.HWADDR,IP4 device show

        return new Promise((resolve, reject) => {
            ShellNmcl.exec('nmcli -t -f GENERAL.CONNECTION,GENERAL.TYPE,GENERAL.DEVICE,GENERAL.STATE,GENERAL.HWADDR,IP4 device show ', (err, stdout, stderr) => {
                if (err) {
                    reject(err);
```

Figure 52 Fichier infosconfig mapp

Le fichier network.js contient 5 fonctions :

- Une fonction listant les connexions configurées avec Network Manager
- Une fonction comparant avec la liste obtenue par la fonction ci-dessus, le nom de la connexion que l'utilisateur veut modifier afin d'éviter de créer une nouvelle connexion et de la modifier si elle a déjà configuré au préalable. Cette fonction sera appelée/utilisée dans les fonctions de mise à jour de l'Ethernet ou du Wi-Fi qui sont séparées car elles ne sont pas gérées exactement de la même façon pour la modification de la configuration. (Pas les mêmes commandes nmcli du Network Manager) et soit on a une connexion Ethernet soit une connexion Wi-Fi.
-la fonction de configuration de l'Ethernet
-la fonction de configuration du wifi
-Une fonction qui en fonction des nouvelles données, paramètres envoyés par l'utilisateur va appeler la fonction de la configuration de l'Ethernet ou du wifi.

2. Côté client

-Retour utilisateur : conteneur/emplacement qui récapitule et affiche ce que l'utilisateur a envoyé comme paramètres de configurations (Cet emplacement m'a aussi été très utile lors du développement, il me permettait de vérifier les données avant de les envoyer au serveur.)

Réponse dans la console du navigateur lorsque la modification de la résolution a été effectuée ou s'il y a une erreur.

Implémentation de la logique de la configuration d'une interface réseau :

-si la connexion Ethernet ou Wi-Fi n'est pas activée les champs de configuration de l'adresse IP, du Mask, de la gateway et les champs des DNS ne doivent pas être visibles et ne peuvent pas être configurés. Par ailleurs le bouton d'envoi est aussi désactivé dans ce cas-là.

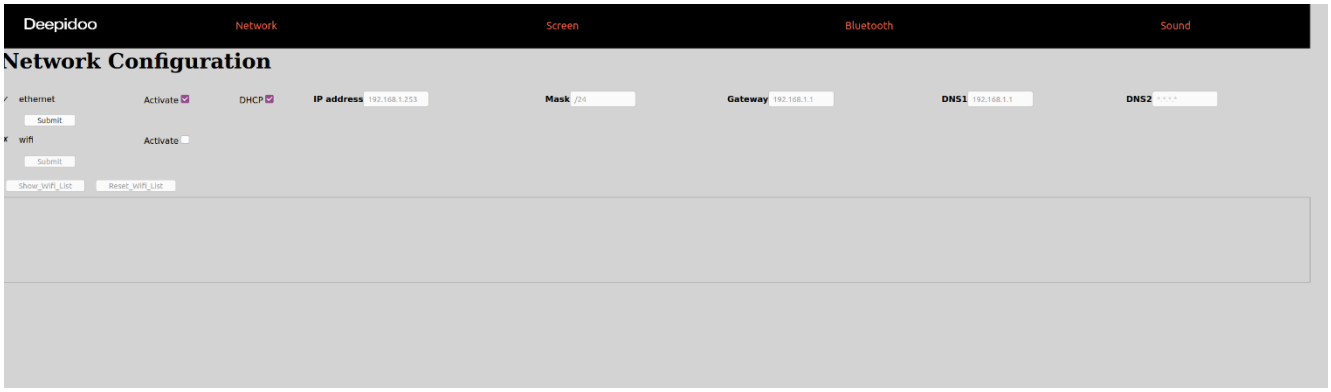


Figure 53 Menu configuration réseau connexion Wi-FI désactivé

- si le protocole d'adressage choisi est DHCP l'utilisateur ne doit pas pouvoir saisir des valeurs pour les différents champs requis, ceux-ci doivent donc être désactivés (grisé sur l'image).

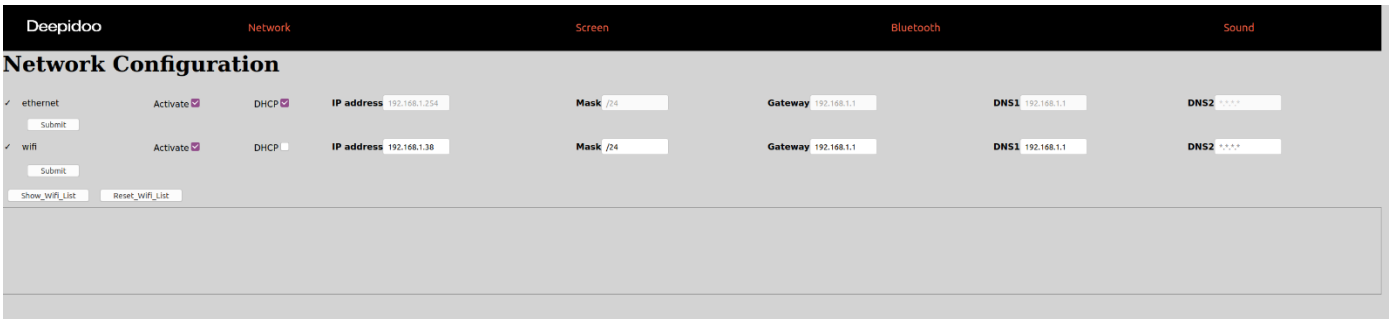


Figure 54 Menu configuration réseau DHCP désactivé saisie manuelle des paramètres

-Vérification des paramètres saisis par l'utilisateur :

Pattern (format des valeurs) pour les adresses réseau (IP,Mask,Gateway,DNS) à respecter(Voir glossaire notions réseau). Si celui-ci ne l'est pas, l'utilisateur ne dois pas pouvoir envoyer les nouvelles données, le bouton d'envoi doit donc être désactiver et l'utilisateur doit être informer que le pattern n'est pas respecté.

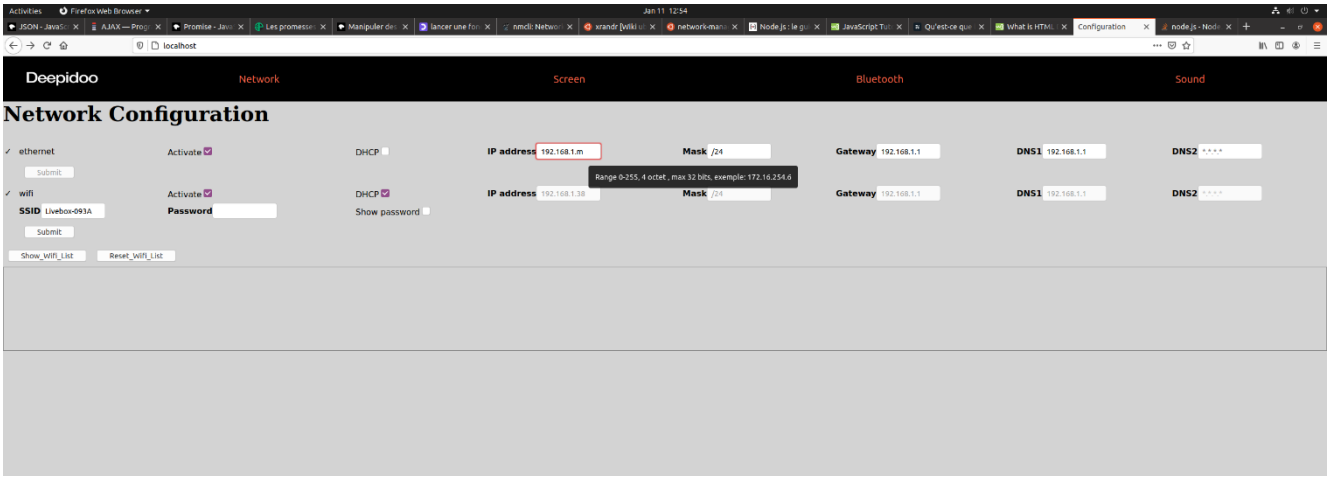


Figure 55 Menu configuration réseau vérification du pattern lors de la saisie

La correspondance afin que la configuration réseau fonctionne entre l'adresse IP, l'adresse réseau et la passerelle (voir glossaire notions réseau) côté serveur n'est pas encore vérifiée elle sera implémentée plus tard.

C. Bilan de l'expérience de stage au sein de Deepidoo

Bilan du travail effectué / des résultats :

Le module de la configuration réseau est selon moi le plus complexe il reste encore aujourd'hui du travail à faire : du côté client, il manque l'information concernant le protocole d'adressage, la case DHCP est tout le temps activée par défaut même si la connexion est configurée en statique.

Du côté serveur il faut donc changer la logique : il ne faut plus récupérer les informations à partir de l'interface avec la commande « nmcli dev show » car il manque l'information sur le protocole d'adressage , il faut donc faire la liste des connections avec « nmcli con show » , récupérer la connexion active et l'interface à laquelle elle est connectée avec « nmcli -a con show » (dans certains cas particulier il peut y en avoir 2 au maximum car 2 cartes Réseau sur la Deepibox une Ethernet et une Wi-Fi mais pas grand intérêt généralement soit on est connecté en Ethernet soit Wi-Fi) afin de récupérer son nom puis faire « nmcli -f [les colonnes qui nous intéresse : le statut, le type de connexion ,le device(interface), adresse ip, mask, gateway, dns , le protocole d'adressage] con show [nom de la connection active] » afin de récupérer les champs qui nous intéressent puis les parser.

Il existe un profil de connexion Ethernet par défaut. Je ne réutilisais pas l'information du nom de l'interface alors que celle- ci est importante car le nom de la connexion Ethernet existante par défaut est le nom de l'interface. Le Network Manager crée des profils de connexion afin de se souvenir des connexions et configurations précédentes. On ne peut pas modifier la connexion Ethernet par défaut car les modifications ne sont pas sauvegardées au redémarrage.

Un autre Refactoring s'impose afin que le code soit plus lisible et compréhensible après les divers ajouts et modifications par suite du Refactoring précédent. Plus besoin du fichier « infosconfig » pour répartir les méthodes en fonction des fichiers (mettre les méthodes du network avec le fichier network et pareil pour la résolution d'écran avec le fichier screen).

Le module résolution d'écran est presque terminé mais il reste encore des petits détails à régler (afficher la résolution actuelle + le script au démarrage pour rendre les modifications persistantes) ou de fonctionnalités supplémentaires à implémenter tels que la gestion de plusieurs écrans simultanés, la rotation d'écran si besoin (cas mineur), ajouter une résolution non listée car dans des cas rare la résolution de l'écran n'est pas disponible par défaut.

Les modules Bluetooth et Volume audio ne sont pas encore amorcés, il faut faire : la recherche des outils nécessaires, la conception, et le développement.

Le code n'est pas commenté mais pour améliorer sa compréhension il est important de le faire.

La page d'accueil de l'application fait relativement vide, il faudrait ajouter des images ou logos.

L'application n'a pas encore été déployée, il s'agit de la phase la plus importante lors de la réalisation d'une application ou d'un logiciel, c'est le rendu du travail effectué. En effet il y a une différence entre l'application fonctionnelle pour les développeurs et l'application fonctionnelle pour tous les utilisateurs. Cela permet d'identifier d'éventuel bugs ou oublis lors du développement et de tester à plus grande échelle.

Bilan des apprentissages techniques, des compétences acquises, consolidées, amorcées…

- Consolidation des compétences en Javascript par l'apprentissage des notions de Promises et Callback.
- Le projet de l'application web m'a permis de voir et pratiquer l'ensemble des pratiques, fonctionnalités et des choses à voir en Javascript que ce soit du côté serveur ou client.
- Faire ses propres fonctions, utiliser des modules/librairies externes.
- Découverte d'outils en ligne de commande Linux consolidations des compétences sur le terminal et le système d'exploitation Linux.
- Approfondissement des connaissances et compétences sur NodeJS, je ne l'ai pas vraiment manipulé lors de mes études à l'iut informatique mais je m'étais renseigné dessus afin de réaliser des exposés et autres.
- Consolidation du travail en autonomie et de la communication, le travail en autonomie ne veut pas dire rester dans son coin et faire tous tout seul.

Il ne faut pas hésiter à poser des questions aux collaborateurs

Afin d'être productif, j'ai dû m'impliquer au maximum dans l'organisation de mon travail, comprendre exactement ce que je faisais et savoir pourquoi et comment je le faisais. De cette expérience j'ai retenu que travailler en autonomie ne signifie pas seulement travailler en toute liberté, cela implique de devoir prendre des initiatives pour planifier et exécuter des tâches, de faire des bilans, d'évaluer son efficacité, de réfléchir aux procédures utilisées. La réflexion est essentielle dans l'acquisition de l'autonomie.

Bilan humain ou relationnel ou communicationnel (découverte d'un environnement de travail, travail en équipe, communication, hiérarchie…) :

Mon stage chez **Deepidoo** a été très instructif. Au cours de ces 2 mois, j'ai ainsi pu observer le fonctionnement d'une entreprise. Au-delà de l'activité de chacun des services/pôles, j'ai pu apprendre comment s'articulent les différents départements d'une telle entreprise. Par ailleurs, les relations humaines entre les différents employés de la société, indépendamment de leurs activités respectives, m'a appris sur le comportement à adopter en entreprise.

La circulation de l'information est l'un des points forts que j'ai retenu de cette société, tant au niveau du travail collaboratif, que dans l'implication de tous dans le bon fonctionnement de la société.

L'atmosphère au sein de la société est très chaleureuse, conviviale, et bienveillante. J'ai ainsi constaté que la hiérarchie des fonctions de la société **Deepidoo** est respectée mais n'empêche pas les échanges entre les employés, favorisant le travail d'équipe.

A titre d'exemple, le temps du déjeuner était régulièrement source d'échanges et de débats.

Concernant les horaires, chaque employé est autonome et libre, tant que le travail est fait. Quotidiennement, nous faisions un point avec mon maitre de stage, ce qui m'a permis de d'avoir des objectifs à court terme et de me motiver.

Grâce à cette convivialité, j'ai pu comprendre que l'activité d'une société est plus performante dans une atmosphère chaleureuse et bienveillante.

Bilan professionnel et personnel

Ce stage m'a permis d'atteindre les objectifs que je m'étais fixés au départ :

-Approfondir mes connaissances

-Développer de nouvelles compétences

-Appréhender le travail en entreprise

Pendant ce stage, je me suis aperçu que les connaissances et les compétences que j'avais acquises à l'IUT m'avaient permis de comprendre les différentes missions qui m'ont été confiées et de comprendre les besoins du client.

Lors des différentes étapes de la création de l'application web, j'ai su me documenter sur Internet, mais aussi demander conseil au développeur senior, la communication dans l'équipe est importante au bon déroulement du projet.

Ce stage fait découvrir une autre vision du développement, de ne plus penser que les utilisateurs vont s'adapter au produit, mais que le produit doit s'adapter à l'utilisateur. Ce que j'ai développé, en plus d'être fonctionnel et optimisé, doit être maintenable et évolutif, c'est-à-dire qu'il faut penser son code de manière qu'il soit lisible par n'importe quel autre développeur. Ce stage a conforté mon envie de continuer et de découvrir de nouvelles technologies qui font la richesse du domaine du développement web.

Ce stage m'a permis de développer des compétences dans le domaine du développement web avec le langage JavaScript et m'a permis de découvrir de nouvelles technologies (les promesses, callback, API fetch, les outils en lignes de commandes Linux Xrandr et Network Manager).

Les différences tâches qui m'ont été confiées m'ont permis de devenir de plus en plus autonome, mais aussi de savoir poser des questions quand je suis dans une impasse. Pour pouvoir poser les questions, il faut pouvoir prendre du recul sur ce que l'on est en train de faire pour pouvoir cibler les recherches et poser les bonnes questions.

La communication est primordiale pour le bon déroulement d'un projet, et cela commence par une bonne intégration au sein d'une entreprise. La taille de l'entreprise étant favorable, connaître tout le monde est plus facile et plus agréable au bon fonctionnement de l'entreprise et au bon déroulement du stage.

Je me suis rendu compte que travailler dans une entreprise à taille humaine est fait pour moi, mais voulant faire de l'alternance en parcours professionnel il faut que la structure de l'entreprise et son budget le permettent, et le nombre de petites structures le permettant est restreint.

Cette expérience professionnelle m'a conforté dans mon choix de me spécialiser dans le développement web bien que la sécurité informatique (cybersécurité)me plaise aussi. Ma formation de DUT Informatique m'ayant formé aux bases de l'informatique, si un jour je veux me réorienter, j'aurais toujours la possibilité de le faire.

Conclusion

Pour conclure, j'ai effectué mon stage de fin d'études de DUT informatique en tant que stagiaire en développement web/Javascript au sein de l'entreprise Deepidoo. Lors de ce stage de 2 mois, j'ai pu mettre en pratique mes connaissances et compétences acquises durant ma formation sur le développement web (HTML, CSS, Javascript) et me suis confronté aux difficultés du monde du travail et du management d'équipes dans le secteur de l'informatique et du marketing.

Ce stage a été très enrichissant pour moi, car il m'a permis de découvrir le domaine du développement web sous une autre forme : des applications et non des sites web vitrines. J'ai aussi découvert l'accompagnement et conseil des clients qui m'a plût : prendre le temps d'écouter le client, de chercher avec lui l'origine du problème, dépanner la solution mise en œuvre dans son magasin.

Ce stage a été une expérience professionnelle significative, car elle m'a permis de choisir la voie dans laquelle je voulais me spécialiser et m'a conforté dans mon choix de vouloir faire de l'alternance pendant ma poursuite d'études. À travers les tâches qui m'ont été confiées, j'ai pu développer mon autonomie, mes connaissances, de nouvelles compétences et découvrir de nouvelles technologies réputées dans le domaine du web. Grâce à ces nouvelles compétences et cette expérience professionnelle en développement web, l'envie de découvrir davantage ce domaine et les multiples technologies a été confortée.

La formation de DUT Informatique, qui a pour but initial de former des techniciens opérationnels, a su m'apporter une base théorique et technique, m'a permis de réaliser ce stage, et me permet aujourd'hui de candidater en cycle professionnel. Une suite que je trouve logique et qui va me permettre, tout au long de ma future carrière de pouvoir évoluer.

Sources :

Commande_shell [Wiki ubuntu-fr]. (s. d.). Consulté 6 janvier 2021, à l'adresse https://doc.ubuntu-fr.org/commande_shell

Console [Wiki ubuntu-fr]. (s. d.). Consulté 6 janvier 2021, à l'adresse https://doc.ubuntu-fr.org/console

JavaScript |MDN. (s. d.). Consulté 5 janvier 2021, à l'adresse https://developer.mozilla.org/fr/docs/Web/JavaScript

JavaScript Tutorial. (s. d.). Consulté 5 janvier 2021, à l'adresse https://www.w3schools.com/js/default.asp

Marketing sensoriel | France | Deepidoo. (s. d.). Deepidoo Français. Consulté 2 janvier 2021, à l'adresse https://www.deepidoo.com

Network-manager [Wiki ubuntu-fr]. (s. d.). Consulté 5 janvier 2021, à l'adresse https://doc.ubuntu-fr.org/network-manager#network-manager_en_ligne_de_commande

nmcli : Manuel de référence de NetworkManager. (s. d.). Consulté 5 janvier 2021, à l'adresse https://developer.gnome.org/NetworkManager/stable/nmcli.html

Node.js. (s. d.-a). Node.js. Node.js. Consulté 5 janvier 2021, à l'adresse https://nodejs.org/fr/

Node.js : Le guide complet. (s. d.-b). Practical Programming. Consulté 5 janvier 2021, à l'adresse https://practicalprogramming.fr/nodejs/

Programmation Web Client Riche—Programmation Web Client Riche. (s. d.). Consulté 11 janvier 2021, à l'adresse https://perso.liris.cnrs.fr/pierre-antoine.champin/enseignement/intro-js/

Terminal [Wiki ubuntu-fr]. (s. d.). Consulté 6 janvier 2021, à l'adresse https://doc.ubuntu-fr.org/terminal

Xrandr [Wiki ubuntu-fr]. (s. d.-a). Consulté 5 janvier 2021, à l'adresse https://doc.ubuntu-fr.org/xrandr

Dossier de presse Deepidoo Consulté 2 janvier 2021, à l'adresse https://566c7d02-b382-4520-a8aa-c8211c9f5cdf.filesusr.com/ugd/cd580e_4d9e1d63190b4c1ea5e9abf259b4db8a.pdf

Différents articles de journaux sur Deepidoo disponibles sur : Marketing sensoriel | France | Deepidoo. (s. d.). Deepidoo Français. Consulté 30 décembre 2020, à l'adresse https://www.deepidoo.com/presse

Sommaire Annexes :

Annexe 1 Histoire de JavaScript :

Le créateur de JavaScript était Brendan Eichen 1995, alors qu'il travaillait chez Netscape Communications. À mesure que le Web gagnait en popularité, une demande progressive de langages de script côté client s'est développée. À l'époque, la majorité des internautes se connectaient via un modem (appareil électronique destinés à faire communiquer des machines entre elles) même lorsque les pages Web augmentaient en taille et en complexité. Netscape a commencé à réfléchir sérieusement au développement du langage de script côté client à un moment précoce, à une époque de pointe de l'innovation technologique, pour gérer un traitement simple. Brendan Eich est la personne qui a travaillé pour Netscape à cette époque et a commencé à développer un langage de script nommé "Mocha", puis nommé "LiveScript", pour la sortie de Netscape Navigator 2 en 1995.Brendan Eich était fasciné par Java lorsque celui-ci est sorti.

Netscape, pendant un certain temps, a fait le meilleur navigateur de l'époque et jouissait d'une suprématie sur le marché. Avant cette date, un aller-retour vers le serveur était nécessaire pour déterminer si un champ obligatoire avait été laissé vide ou si une valeur saisie était invalide. Netscape Navigator voulait changer cela avec l'introduction de JavaScript. Le potentiel et la capacité de gérer certaines validations nécessaires sur le client étaient une nouvelle fonctionnalité intéressante à l'époque où l'utilisation des modems téléphoniques était courante et bien connue. Les vitesses lentes associées ont transformé chaque trajet vers le serveur en un exercice de patience.

Plus tard en 1995, alors que Microsoft affrontait la concurrence avec la menace créée par le Web, le projet Internet Explorer a commencé dans une tentative totale de lutter contre le contrôle de la plate-forme émergente de Netscape.

Peu à peu, Microsoft est devenu une menace mortelle, convaincant Netscape avec son Internet Explorer. Lentement, un processus de normalisation a commencé à se développer pour empêcher Microsoft d'acquérir la puissance du langage JavaScript. En outre, ils se sont associés à Sun pour influencer leur intérêt commun à briser le monopole de Microsoft.

Brendan Eich a supposé qu'avec Sun à bord il fallait surfer sur le raz-de-marée de la construction autour de Java et de positionner JavaScript comme langage complémentaire à Java. Netscape's Mocha/ LiveScript a ensuite été nommé JavaScript, Sun Microsystems et Netscape étaient partenaires, et la machine virtuelle Java de plus en plus populaire. Ce changement de nom servait les intérêts des deux sociétés.

Malheureusement pour JavaScript, sa position de marketing précoce a survécu à son utilité et est devenue plus tard une marque d'acceptation par le marché car elle est devenue une technologie viable en soi.

Depuis, JavaScript est devenu une fonctionnalité importante de tous les principaux navigateurs Web du marché. N'est plus lié à une simple validation des données, JavaScript interagit et fonctionne désormais avec presque tous les aspects de la fenêtre du navigateur ainsi que son contenu. JavaScript est reconnu comme un langage de programmation complet capable de gérer des calculs et des interactions complexes, notamment des fermetures, des fonctions anonymes (lambda) et également la méta-programmation. JavaScript est devenu un élément tellement essentiel du Web que même les navigateurs alternatifs, y compris les navigateurs fonctionnant sur les téléphones mobiles et ceux conçus pour les utilisateurs handicapés, le prennent en charge et le maintiennent. Même Microsoft, avec son langage de script côté client appelé VBScript qui a abouti à son implémentation JavaScript dans Internet Explorer à partir de sa version initiale.

Annexe 2 & 3 Fiche technique Deepibox (shuttle) :

Fourth generation of Shuttle's XPC nano Series brings Whiskey Lake support

The NC10 series is powered by Intel's power-saving ULV (ultra-low-voltage) processors of the Whiskey-Lake-U generation and comprises four models with processors from Celeron to Core i7. All four models support two digital video outputs for UHD/4K displays with 60 Hz and one 2.5" drive that is up to 15 mm in height as well as one M.2-2280 NVMe SSD card. Professional users will appreciate Intel Gigabit-LAN and one serial port which indicates what purposes the NC10 series is mainly intended for: Digital Signage, POS, control, office or even multimedia.

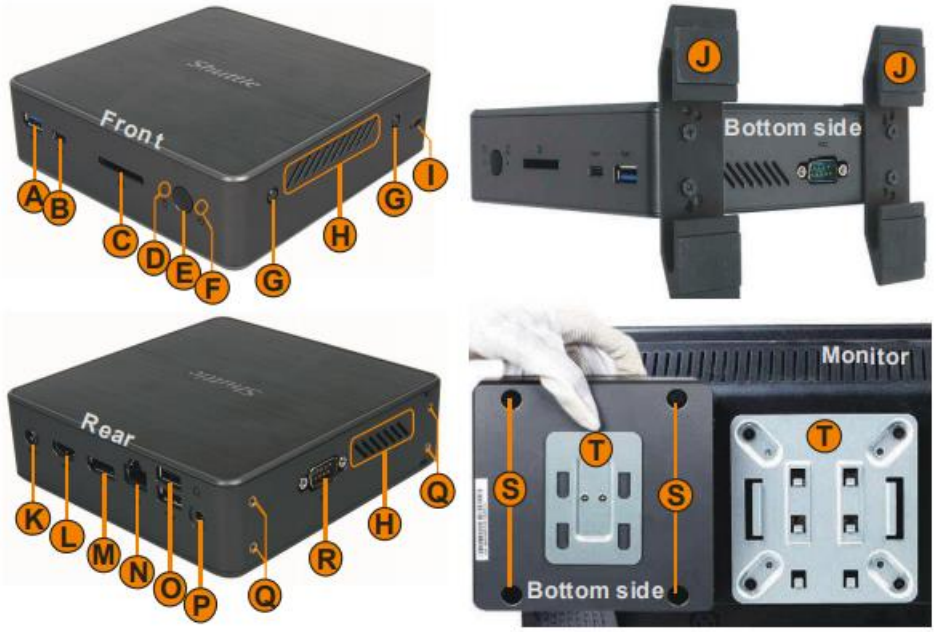
Feature Highlights

Slim Design	• Slim plastic chassis, black • Dimensions: 142x142x42 mm (LWH), 847 ml • Incl. Stand & VESA mount (75/100 mm) • Hole for Kensington Lock • Operating temperature: max. 40 °C
Operating System	• An operating system is not included • Supports Windows 10, Linux (64-bit only)
Processor	• Intel Celeron 4205U, Dual Core, 15 W TDP Intel ULV "Whiskey-Lake-U" Generation • Integrated Intel UHD graphics 610, DX12
Memory	• Supports up to 2x 32 GB DDR4-2133 SO-DIMM memory modules
Drive Bay	• One 6.35 cm / 2.5" bay, 15 mm height supports one SATA hard disk or SSD
M.2 Slot	• M.2-2280 slot supports SSD card (SATA+PCIe)
Connectors	• HDMI 2.0a, DisplayPort 1.2 • 2x USB 3.2 Gen 1 (Type A/C), 2x USB 2.0 • Intel Gigabit LAN, RS232 COM port • SD card reader, Audio Combo
WLAN	• Wireless LAN 802.11n, internal antenna • Optional upgradeable with Shuttle WLN-M
Power Supply	• External 65 W fanless power adapter

XPC nano Barebone NC10U (Celeron)

Images for illustration purposes only.
This product does include the stand and VESA mount, but does not include memory, storage and operating system.

Shuttle XPC nano Barebone NC10U – Product Views



A	USB 3.2 Gen 1 Type A (Blue)	**K**	DC input for power adapter
B	USB 3.2 Gen 1 Type C	**L**	HDMI
C	SD Card reader	**M**	DisplayPort
D	Hard disk LED indicator	**N**	Gigabit LAN (RJ45)
E	On/Off Button	**O**	2x USB 2.0
F	Power-on LED indicator	**P**	Audio Combo (Headphones & Mic)
G	2x perforation for optional WLAN antenna	**Q**	4x Mounting hole for vertical stand
H	Vents	**R**	RS232 COM port *)
I	Hole for Kensington Lock	**S**	4x Rubber foot
J	2x Vertical stand	**T**	VESA mounting kit (2 pieces)

*) Note: The serial connector (COM port) cannot be used, if NC10U is operated in vertical position.

Glossaire technique :

Rappel de notions de réseau : A savoir : Il existe deux types d'adresse réseau ivp4 et ipv6 nous ne parlerons ici que des adresses réseau ipv4 car l'ipv6 n'est pas totalement mis en place en France et dans le monde et n'est presque pas utiliser.

Octet : une unité de mesure. Les octets permettent le codage de caractères ou de valeurs numériques. Un octet est composé de 8 bits, l'unité de base en informatique

Bits : unité adressable d'un ordinateur. Unité élémentaire d'information ne pouvant prendre que deux valeurs, notées 0 ou 1.

Une adresse réseau est codée sur 4 octets soit 32 bytes. Les valeurs possibles sont donc 2^8 -1 soit de 0 à 255 Il s'agit du pattern(format) d'une adresse réseau.

Exemple d'une adresse : 192.168.1.1

L'originalité de ce format d'adressage réside dans l'association de l'identification du réseau avec l'identification de l'hôte (l'ordinateur).

La partie réseau est commune à l'ensemble des hôtes d'un même réseau,

La partie hôte est unique à l'intérieur d'un même réseau.

Le masque de réseau sert à séparer les parties réseau et hôte d'une adresse. On retrouve l'adresse du réseau en effectuant un ET logique bit à bit entre une adresse complète et le masque de réseau.

Le masque réseau permet de retrouver sur quel réseau est située la machine à partir de son adresse IP, il possède deux notations, le pattern vu précédemment ou la notation CIDR (/valeur, valeur max 32) calculé à partir de tous les Bytes ayant pour valeurs 1 en partant de la gauche. En masque en notation CIDR ayant pour valeur /24 correspond à 255.255.255.0.

Certaines adresses sont réservées et ne peuvent pas être utiliser tels que l'adresse de loopback (127.0.0.1) qui permet la communication entre les processus locaux du système de la machine ou encore l'adresse de diffusion (*.*.*.255) d'un réseau qui permet d'envoyer des informations ou messages à l'ensemble des ordinateurs présent sur ce réseau. Une vérification par encore implanté qui sera fait plus tard.

Tableau 1. Exemple : adresse IP `192.168.1.1`

Adresse complète	192.168.__1.__1
Masque de réseau	255.255.255.__0
Partie réseau	192.168.__1.___
Partie hôte	___.___.___.__1
Adresse Réseau	192.168.__1.__0
Adresse de diffusion	192.168.__1.255

Protocole d'adressage :

Le protocole d'adressage ou de configuration peut être statique (qui veut dire manuel) ou dynamique (DHCP).
Si le protocole de configuration est manuel il faut renseigner en plus la configuration IP (adresse IP, masque du réseau, passerelle du réseau, un/des serveurs DNS).
Le protocole DHCP permet d'obtenir dynamiquement sa configuration réseau (c'est-à-dire sans intervention particulière) par un serveur du réseau.

Serveur DNS : Le serveur DNS (Domain Name System, ou Système de noms de domaine en français) est un service d'Internet dont la principale fonction est de traduire un nom de domaine en adresse IP compréhensible par les ordinateurs et les réseaux. Pour simplifier, le serveur DNS agit comme un annuaire que consulte un ordinateur au moment d'accéder à un autre ordinateur via un réseau. Autrement dit, le serveur DNS est ce service qui permet d'associer un nom de domaine à une adresse IP.

Adresse internet

Serveur HTTP : est un serveur web. Les serveurs web publics sont reliés à Internet et hébergent des ressources (pages web, images, vidéos, etc.) du Web (toile mondiale qui est une application/service d'Internet). Ces ressources peuvent être statiques (servies telle-quelles) ou dynamiques (construites à la demande par le serveur). La fonction principale d'un serveur Web est de stocker et délivrer des pages web qui sont généralement rendues en HTML. Le protocole de communication HTTP (qui est le protocole du web) afin de dialoguer via le réseau avec le logiciel client : un navigateur web

Certains serveurs sont seulement accessibles sur des réseaux privés (intranets) et hébergent des sites utilisateurs, des documents, ou des logiciels, internes à une entreprise, une administration, etc.

Mots-clés :

B2B: business to business qui veut dire d'entreprise à entreprise.

Processus : Un processus (process en anglais) est une séquence d'instructions (en langage machine, c'est-à-dire le binaire) chargée en mémoire (RAM) qui s'exécutent par le processeur. Un processus a une notion séquentielle dans le temps, ce qui veut dire qu'il a un début et une fin. L'exécution d'un processus peut être manuel ou automatique.

Refactoring : retouche du code déjà écrit afin que celui-ci soit plus lisible, compréhensible, plus facile à maintenir et à être repris par d'autres programmeurs. Cela se fait par le découpage du code et des fonctions ou méthodes trop longues, mettre des noms explicites aux variables, etc

Parsage : mise en forme des données sous un certain format.

Classe (informatique) définition de wikipédia: En programmation orientée objet, la déclaration d'une classe regroupe des membres, méthodes et propriétés communs à un ensemble d'objets. La classe déclare, d'une part, des attributs représentant l'état des objets et, d'autre part, des méthodes représentant leur comportement. Une classe représente donc une catégorie d'objets.

JSON : JSON (JavaScript Object Notation – Notation Objet issue de JavaScript) est un sous-ensemble du langage Javascript, utilisé comme format de données sur le Web.Il s'agit d'un format léger d'échange de données. Il est facile à lire ou à écrire pour des humains. Il est aisément analysable ou générable par des machines. JSON se base sur deux structures : Une collection de couples nom/valeur, Une liste de valeurs ordonnées. Exemple :

```
"HDMI": {
    "connected": false,
    "modes": [],
    "index": 0
  },
```

AJAX : Ajax (Asynchronous Javascript And XML) est une combinaison de technologies déjà existantes : HTML/CSS, Javascript/DOM, XML et les requêtes HTTP. Cette technologie ou méthode permet de créer des applications web pouvant communiquer avec le serveur, sans avoir à se recharger totalement :
Conservation de l'état de l'application
Fluidité pour l'utilisateur
Mais cela suppose que le code Javascript ne se bloque pas en attendant la réponse du serveur
→ appel asynchrone (événements)
Permet de modifier une partie de l'application web affichée sur le poste client sans avoir à rafraichir la page ou l'application web complètement

Permet de construire des applications Web et des sites web dynamiques interactifs

Pour exécuter la requête HTTP (POST, GET) au serveur, le client utilise l'API XMLHttpRequest ou l'API fetch: le futur de XMLHttpRequest.

Asynchrone fait référence à la capacité de lancer une opération et passer aux suivantes sans attendre que celle-ci soit terminée.

Callback JavaScript : Un rappel est une fonction transmise comme argument à une autre fonction
Cette technique permet à une fonction d'appeler une autre fonction
Une fonction de rappel peut s'exécuter une fois qu'une autre fonction est terminée.
Les fonctions JavaScript sont exécutées dans la séquence qu'elles sont appelées. Pas dans la séquence, ils sont définis. Meilleur contrôle sur le moment de l'exécution et moins de code car moins de fonction.

Promesse JavaScript : Une promesse est un objet (Promise) qui représente la complétion ou l'échec d'une opération asynchrone. Une promesse est un objet qui est renvoyé et auquel on attache des callbacks plutôt que de passer des callbacks à une fonction. Nécessité des mots clés try catch afin de récupérer l'échec de la promesse.

À la différence des imbrications de callbacks, une promesse apporte certaines garanties :

Les callbacks ne seront jamais appelés avant la fin du parcours de la boucle d'évènements JavaScript courante

Les callbacks ajoutés grâce à then seront appelés, y compris après le succès ou l'échec de l'opération asynchrone

Plusieurs callbacks peuvent être ajoutés en appelant then plusieurs fois, ils seront alors exécutés l'un après l'autre selon l'ordre dans lequel ils ont été insérés.

Un besoin fréquent est d'exécuter deux ou plus d'opérations asynchrones les unes à la suite des autres, avec chaque opération qui démarre lorsque la précédente a réussi et en utilisant le résultat de l'étape précédente. Ceci peut être réalisé en créant une chaîne de promesses.