



DOSSIER DE VALIDATION DEVELOPPEUR FULL STACK (DFS)

Titre de niveau 6 inscrit au RNCP

Titre :

Démontrer l'acquisition des compétences décrites dans le référentiel de la certification visée

Marc DE LARREA

24/11/2023

Tuteur : Cédric NOTIN / Sofiane MESSAOUDI

Remerciements

Je tiens à exprimer ma gratitude envers toutes les personnes qui m'ont soutenu et aidé tout au long de mon parcours en tant que développeur Full Stack. Leur soutien et leurs précieux conseils ont été essentiels pour mon apprentissage et ma réussite.

Tout d'abord, je souhaite remercier chaleureusement **Baptiste Coulaud**, mon mentor, qui m'a initié à ce vaste monde du développement et m'a transmis ses connaissances avec passion. Sa patience et sa disponibilité ont été d'une aide précieuse, et je lui suis reconnaissant de m'avoir encouragé à me dépasser constamment. C'est un honneur pour moi que d'avoir été l'élève d'un développeur aussi talentueux que lui.

Un grand merci également à **mes collègues de travail en formation**, qui m'ont accompagné jour après jour dans cette aventure. Leur expertise, leur soutien et leur esprit d'équipe ont créé un environnement propice à l'apprentissage et à l'épanouissement professionnel. Je suis reconnaissant d'avoir eu la chance de travailler avec des personnes aussi talentueuses et bienveillantes.

Je tiens également à exprimer ma reconnaissance envers l'**IT-Akademy**, mon organisme de formation. Leurs programmes d'apprentissage complets et leurs formateurs compétents m'ont permis de devenir un meilleur développeur jour après jour. Je suis reconnaissant envers toute l'équipe pédagogique pour leur engagement et leur dévouement.

Enfin, je souhaite remercier mon entreprise, **ARC Europe France**, qui m'a offert l'opportunité de mettre en pratique mes compétences et d'approfondir mes connaissances. Leur confiance en moi et leur soutien constant ont été déterminants dans ma progression professionnelle. Je suis reconnaissant envers toute l'équipe d'ARC France pour leur accompagnement et leur encouragement. Un grand merci tout particulier à **Aurélien Vaudan** pour toutes les explications et aides qu'il m'a apporté jour après jour dans la société. C'est un plaisir de travailler avec des personnes si bienveillantes et attachantes que cet homme.

Avant-Propos

Ce rapport a été réalisé dans le cadre de ma formation en tant que développeur Full Stack à l' IT-Akademy, du 17 novembre 2022 au 24 novembre 2023. Son objectif est de valider les différents blocs de compétences requis pour l'obtention du diplôme.

Mon parcours personnel est marqué par une diversité de métiers exercés, où la minutie et la bienveillance ont toujours été les maîtres-mots. J'ai eu l'opportunité d'explorer plusieurs domaines et de devenir compétent dans ceux-ci. Cependant, ma passion pour l'apprentissage et mon désir d'aider les autres ont continué de grandir.

C'est pourquoi j'ai décidé de me lancer dans un nouveau défi, un défi de taille : devenir un développeur accompli. J'ai observé les incroyables possibilités offertes par le monde numérique et j'ai décidé de m'immerger dans cet univers en constante évolution.

Au cours de cette formation, j'ai eu l'opportunité d'apprendre les bases fondamentales du développement, de me familiariser avec les langages de programmation et les technologies les plus couramment utilisés. J'ai également acquis une compréhension globale de la conception et du déploiement d'applications web, ainsi que de la gestion des bases de données.

Je tiens à remercier chaleureusement l' IT-Akademy pour la qualité de son programme de formation et l'encadrement pédagogique attentif de ses formateurs. Leur expertise et leur passion pour le développement m'ont inspiré et stimulé tout au long de ce parcours.

J'aborde ce projet de soutenance avec enthousiasme et détermination.

Je suis fier du chemin parcouru jusqu'à présent et je suis convaincu que cette expérience en tant que développeur Full Stack me permettra d'ouvrir de nouvelles perspectives professionnelles passionnantes.

Ce rapport témoigne de mes compétences, de mes réalisations et de mon engagement dans cette formation.

Je tiens à exprimer ma gratitude envers tous ceux qui m'ont soutenu et encouragé dans cette aventure, ainsi qu'envers mes collègues de classe qui ont partagé cette expérience avec moi.

Table des matières

Pages :

| | |
|---|----|
| Remerciements | 3 |
| Avant-Propos | 4 |
| Table des matières | 5 |
| Correspondance Référentiel | 8 |
| Introduction..... | 9 |
| Le projet IOT (Purpaws) de mon école IT-Akademy | 9 |
| Le stage en entreprise (Arc Europe France) | 9 |
| PROJET IOT (Internet des objets) | 10 |
| Scrum..... | 15 |
| Canva | 16 |
| Les débuts du travail d'équipe | 17 |
| Notion | 17 |
| Wireframes | 18 |
| Maquettes..... | 19 |
| UML du projet..... | 20 |
| Diagrammes de Cas d'Utilisation : | 20 |
| Diagrammes de Séquences..... | 21 |
| Méthode d'Organisation et d'Analyse (MOA) | 21 |
| Modèle Conceptuel de Données (MCD) : | 22 |
| Modèle Physique de Données (MPD) | 22 |
| Risques et Criticité | 23 |
| Chef de projet..... | 24 |
| Cahier des charges | 25 |
| GitHub | 25 |
| DevOps : | 27 |
| Création de branches localisées | 28 |
| CI/CD | 28 |
| Maintien de Scrum | 29 |
| Modéliser une maquette..... | 30 |

| | |
|---|----|
| Composant Header.js | 31 |
| Header.scss | 32 |
| Visuels de l'application | 33 |
| API..... | 34 |
| Api.json | 35 |
| Imprévu | 37 |
| Etude de marché..... | 38 |
| IA..... | 38 |
| Soutenance du projet Purrpaws..... | 39 |
| UX..... | 40 |
| NotificationButton.js | 41 |
| Mockoon..... | 42 |
| Postman | 43 |
| SEO..... | 44 |
| Accessibilité web et son importance pour notre site | 45 |
| L'entreprise | 47 |
| Déroulé du stage..... | 48 |
| Cartographie Applicative Exacte..... | 49 |
| Cartographie Applicative localisée | 50 |
| Compléments | 53 |
| Back API node express | 53 |
| .env | 55 |
| .gitignore..... | 55 |
| RGPD | 56 |
| Plateforme d'hébergement | 57 |
| SSIS..... | 58 |
| Transactions PayPal | 59 |
| CI/CD (Partie 2) | 61 |
| Sécurité..... | 63 |
| 1.HTTPS | 63 |
| 2.Les tokens :..... | 63 |
| 3.Les Mots de passe en base de données :..... | 63 |
| 4.RGPD | 64 |
| 5.OWASP | 64 |

| | |
|-----------------------------------|----|
| 6.Double authentification : | 64 |
| 7.Chat Gpt | 65 |
| DEVOPS..... | 66 |
| Mise en Production | 66 |
| Accès serveurs..... | 66 |
| Entente avec l'infra | 66 |
| Conclusion | 67 |
| CV..... | 68 |

Correspondance Référentiel

| | Compétences | Pages | | Compétences | Pages |
|-----|-------------|---------|-----|-------------|---------|
| A 1 | C 1 | 9 | A 4 | C 15 | 35 |
| | C 2 | 16 | | C 16 | 52 |
| | C 3 | 9 | | C 17 | 35 |
| | C 4 | 17 | | C 18 | 35 |
| | C 5 | 9 | | C 19 | 50 |
| | C 6 | 14 | | C 20 | 27 / 61 |
| | C 7 | 16 | | C 21 | 16 / 35 |
| A 2 | Compétences | Pages | A 5 | Compétences | Pages |
| | C 8 | 19 | | C 22 | 56 |
| | C 9 | 19 | | C 23 | 54 |
| | C 10 | 19 | | C 24 | 47 |
| A 3 | C 11 | 12 / 19 | A 6 | Compétences | Pages |
| | Compétences | Pages | | C 25 | 55 |
| | C 12 | 30 | | C 26 | 44 |
| | C 13 | 32 | | C 27 | 58 |
| | C 14 | 30 | | C 28 | 43 |
| | | | | C 29 | 59 |

Introduction

Le projet IOT (Purpaws) de mon école IT-Akademy

Au cours de ma formation à l'IT-Akademy, j'ai eu l'opportunité de participer à un projet passionnant axé sur l'IoT (Internet des Objets).

En tant que chef de projet, j'ai été chargé de superviser l'avancement et de gérer de manière Agile/Scrum le travail d'équipe.

Dès le premier jour, mes collègues de projet (Mel, Ryan, Yannis) m'ont désigné pour ce rôle en raison de mon expérience professionnelle, estimant que j'étais le mieux qualifié pour assumer cette responsabilité.

Dans le cadre de ce projet, nous avons dû identifier les technologies adaptées et concevoir une gamelle connectée pour chat. Grâce à cet exercice, nous avons pu mettre en pratique de nombreux concepts et compétences requis pour la validation de notre formation.

L'école se situe à Charbonnières-les-Bains, j'y étais de décembre 2022 au juillet 2023

Le stage en entreprise (Arc Europe France)

Parallèlement à ce projet, j'ai également eu la chance d'effectuer un stage au sein de l'entreprise ARC.

ARC Europe France est une entreprise spécialisée dans le dépannage automobile. Mon travail au sein de cette société a été axé sur le développement et maintien d'une application destinée à aider les conducteurs en détresse (Cactus). J'ai eu l'occasion de travailler en équipe sur ce projet, contribuant ainsi à son développement et à son amélioration.

Mon stage s'est déroulé du 17 juillet au 24 novembre 2023.

PROJET IOT (Internet des objets)

L'IT-Akademy nous a demandé de réaliser un projet IOT (Internet Of Things en anglais), nous avons donc (toute la classe) écrit sur des petits papiers des idées.

Après avoir lu et noté la pertinence des idées, nous avons voté pour celles qui nous intéressaient le plus et ensuite créé des équipes de 4.

J'ai personnellement choisi de participer à la création d'une gamelle pour chat connectée à son application web et mobile ainsi qu'à son site marchand. J'ai fait ce choix car, en tant que propriétaire d'un chat, j'ai pu facilement m'identifier aux problèmes potentiels des propriétaires qui ont des chats qui ne savent pas réguler leur appétit. Cette gamelle gère l'approvisionnement en croquettes des chats selon les paramètres souhaités par leurs maîtres. Le site marchand vend la gamelle, service tout en un.

Dans ce projet, la première étape a été la création de l'équipe. J'ai eu la chance de travailler avec d'excellents coéquipiers :

- **Ryan**, qui possède une logique impressionnante et une aisance remarquable en algorithmie,
- **Mel**, qui a un goût prononcé pour le front-end et une réflexion hors du commun,
- Enfin, **Yannis**, qui possède une expérience solide en programmation, un grand sens de l'humour, ainsi qu'une source de connaissances et de motivation très appréciable pour l'apprentissage.

The slide is titled "PRÉSENTATION" in a blue header bar with a magnifying glass icon. It features four team members: Marc (Chef de Projet), Yannis (Lead Tech), Ryan (Back Lead), and Mel (Front Lead). Each member has a portrait photo and a paw print icon above their name. The background is white with some abstract shapes.

| CHEF DE PROJET | LEAD TECH | BACK LEAD | FRONT LEAD |
|----------------|---------------|-------------|------------|
| Marc | Yannis | Ryan | Mel |

Lorsque nous nous sommes concertés pour déterminer comment le projet allait se dérouler, il est apparu évident à mes coéquipiers que ma place serait celle de chef de projet.

En effet, mes expériences passées m'ont fait apparaître comme un leader à leurs yeux, et ils m'ont donc confié cette responsabilité sans même me laisser d'autre choix.

Cela m'a permis de remplir plusieurs tâches cruciales du projet, notamment :

- Arbitrer les décisions en cas de litiges ou d'indécisions,
- Collaborer avec l'équipe pour définir l'organisation et les objectifs (utilisation de Scrum et définition de l'objectif final), **(C3)**
- Gérer les petits conflits qui peuvent survenir,
- Organiser le groupe afin de minimiser les éventuels ralentissements, **(C5)**
- Évaluer les forces et les faiblesses de chacun afin d'équilibrer la répartition du travail et de garantir au maximum que le projet puisse être mené à bien dans les délais impartis.

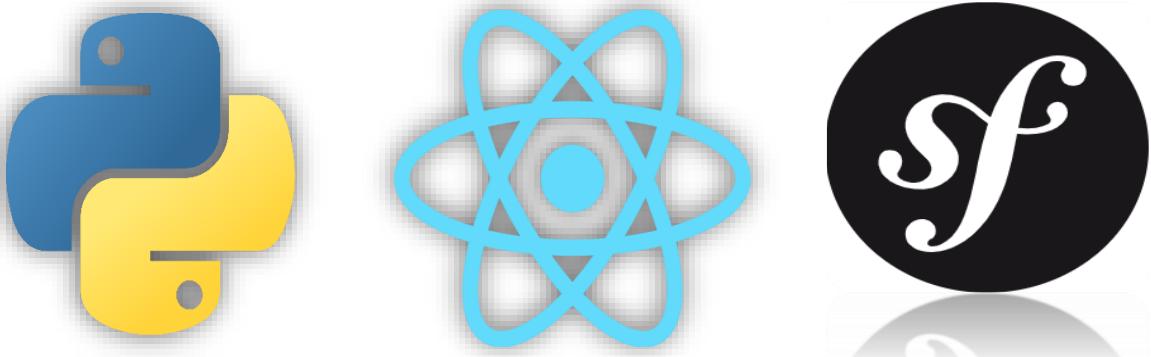
La deuxième étape a été de déterminer comment nous allions réaliser une gamelle connectée pour chat intégrée à une application web. Cette phase a soulevé plusieurs questions essentielles : **(C1)**

- 1.** Quel langage de programmation devons-nous utiliser ?
- 2.** Quelle architecture convient le mieux ?
- 3.** Quelles technologies devons-nous intégrer ?
- 4.** Comment construire le dispositif en lui-même ?
- 5.** Quelles sont les exigences de l'école à ce sujet ?
- 6.** Comment pouvons-nous travailler de manière efficace en équipe ?
- 7.** Qui sera responsable de quelles tâches ?

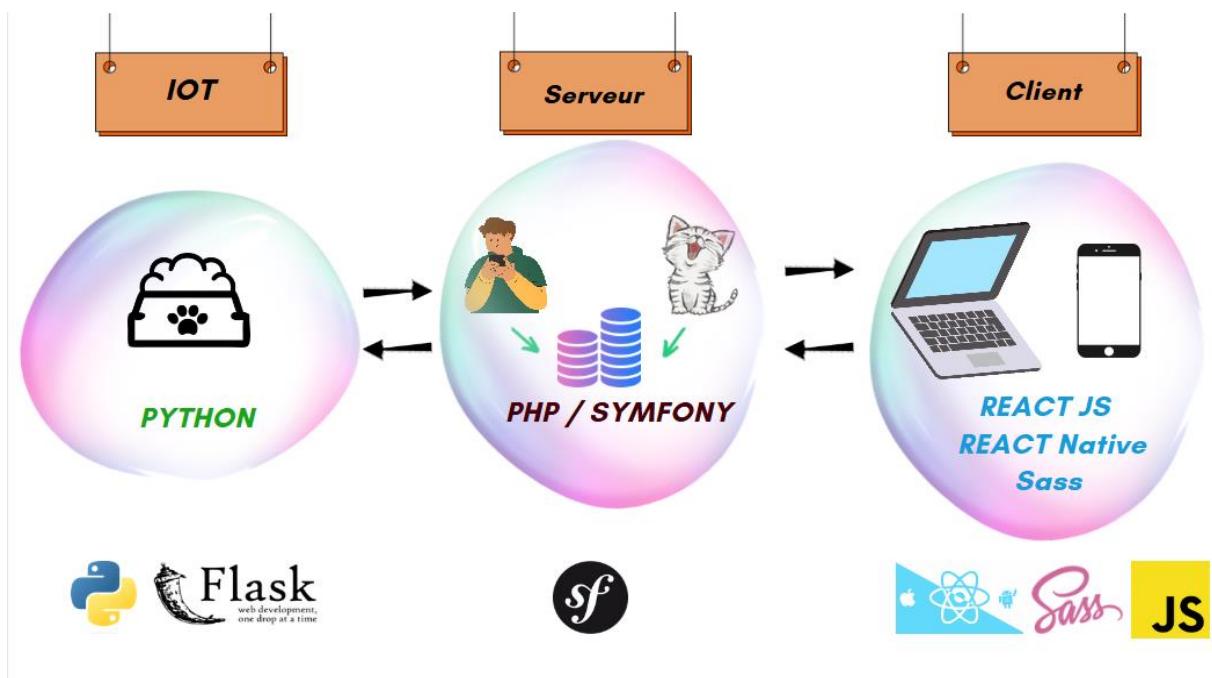
Nous avons abordé chacune de ces questions en groupe, et j'ai suggéré une idée supplémentaire : la nécessité de documenter l'ensemble du processus, en créant une présentation.

En fin de compte, nous avons pris les décisions suivantes : **(C12)**

1. Langages de programmation : **Python** pour la partie embarquée, **React** pour le front-end, et **PHP** pour le back-end.



Et l'interaction entre ces langages nécessitait une vue claire :



Le langage **python** sera utilisé pour faire le lien entre la gamelle et le serveur,

PHP symfony sera le langage serveur (**Backend**) et fera le lien entre la gamelle et l'application web (**Frontend**)

React JS sera le langage de l'application web et fera le lien entre le serveur et l'application.

Nous avons préféré rester sur des langages de programmation que nous avions vus en cours, le temps nous faisant défaut. Connaître même un peu le langage étant un gain de temps, les meilleures options que nous avions étaient donc **PHP**, **React** et **Python**.

2. Architecture MVC (Modèle-Vue-Contrôleur) : (C11)

C'est un modèle de conception logicielle qui divise une application en trois composants interconnectés : le modèle, la vue et le contrôleur;

- **Le modèle** représente la logique de l'application et les données. Dans notre projet, le modèle est responsable de la gestion des données de la gamelle connectée, telles que les informations sur la nourriture, les horaires d'alimentation, le nom du chat, etc.
- **La vue** est responsable de l'interface utilisateur. Dans notre cas, il s'agit de l'interface web que les utilisateurs utiliseront pour interagir avec la gamelle connectée.
- **Le contrôleur** agit comme un intermédiaire entre le modèle et la vue. Il traite les demandes de l'utilisateur et met à jour le modèle en conséquence.

Par exemple, lorsqu'un utilisateur planifie l'alimentation de son chat via l'application, le contrôleur s'assure que ces informations sont enregistrées dans le modèle et que la vue est mise à jour pour refléter les changements.

L'architecture MVC offre une séparation claire des fichiers du code ainsi qu'une logique plus facile à entrevoir, ce qui facilite la maintenance et l'évolutivité de l'application.

3. Utilisation d'un Raspberry Pi ainsi qu'une puce pour le suivi du chat :

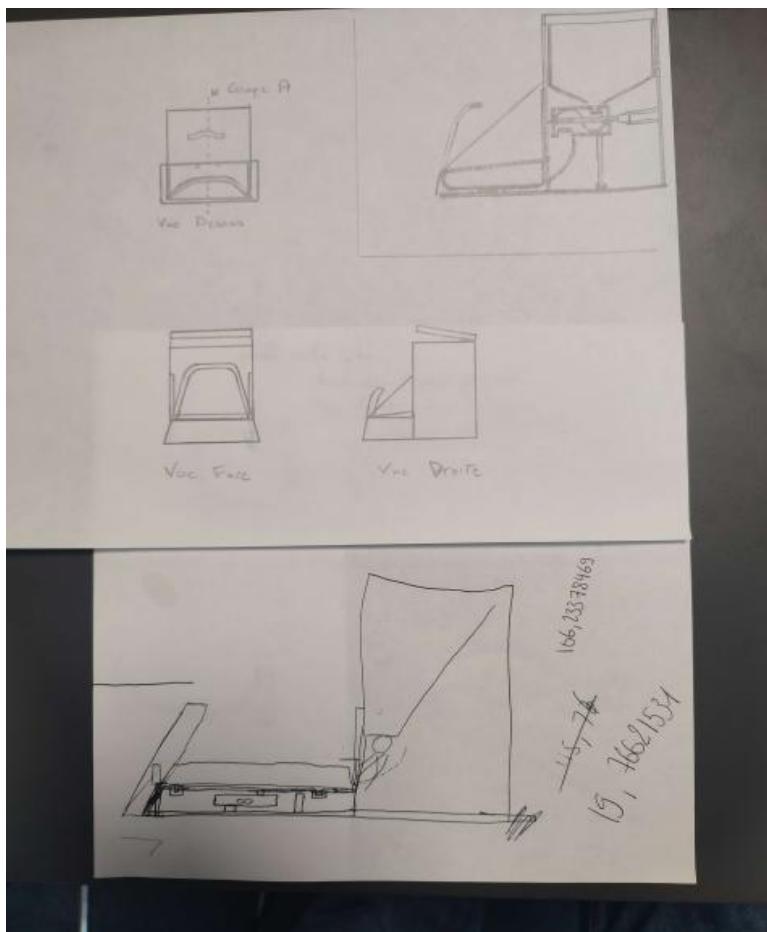
Le Raspberry Pi est un petit ordinateur monocarte abordable qui peut être utilisé dans une variété d'applications, y compris les projets IoT.

Dans notre projet, nous avons choisi d'utiliser un Raspberry Pi pour plusieurs raisons :

- Il est compact et économique en énergie, ce qui en fait un choix idéal pour un dispositif embarqué.
- Il dispose de ports d'entrée/sortie permettant de connecter des capteurs et des actionneurs.
- Il peut exécuter une version de Python, ce qui facilite la programmation de la partie embarquée de notre gamelle connectée.

Le Raspberry Pi sera responsable de la collecte de données à partir de capteurs sur la gamelle, comme le poids de la nourriture, et de l'envoi de ces données au serveur back-end via une connexion Internet. Cela permet aux utilisateurs d'accéder à ces informations via l'application web.

4. Nous avons élaboré plusieurs plans individuellement pour ne pas influencer les idées des autres membres, puis nous avons fusionné nos meilleures idées pour créer un prototype.



5. L'école nous avait imposé certains langages, mais nous avons pu justifier notre choix d'utiliser Python. Nos recherches sur les interactions entre les objets et les différentes nécessités des connexions (serveurs, applications) pointaient vers ce langage.

L'école est également de fait notre client. L'IT-Akademy nous avait demandé de réaliser un projet IoT (Internet des objets) ainsi qu'un "Fil Rouge" qui consistait initialement en la création séparée d'un site marchand et d'un objet IoT.

Cependant, nous avons négocié avec l'école la possibilité de fusionner ces deux objectifs en un seul projet, en développant la gamelle connectée ainsi que son site marchand associé.

6. Nous avons adopté la méthode agile Scrum pour travailler ensemble de manière efficace tout au long de notre aventure.

Scrum

La méthodologie Scrum est un cadre de travail agile qui favorise la collaboration, la transparence et l'adaptabilité dans le développement de logiciels. Dans notre projet, nous avons choisi d'adopter Scrum pour plusieurs raisons :

- Scrum nous permet de travailler de manière itérative et incrémentale, ce qui signifie que nous développons notre projet par étapes successives et fonctionnelles.
- Les "sprints" de Scrum sont des périodes de développement définies, généralement de deux à quatre semaines (dans notre cas, nous faisions un nouveau sprint toutes les deux semaines), au cours desquelles nous nous concentrons sur la réalisation d'objectifs spécifiques.
- Scrum encourage la communication régulière au sein de l'équipe, ce qui est essentiel pour résoudre rapidement les problèmes et s'assurer que le projet avance sans problème.

Au fil des jours, il m'est apparu évident que notre rythme et notre manière de travailler ensemble pouvaient encore être optimisés. J'ai donc décidé de tenir une brève réunion tous les soirs, d'une durée de 15 à 20 minutes, pour résumer la journée en trois points : **(C6)**

- *Qu'as-tu accompli aujourd'hui ?*

- *Quels problèmes as-tu rencontrés ?*

- *Qu'as-tu prévu de faire demain ?*

Cela présentait plusieurs avantages :

- Chacun d'entre nous savait où en étaient les autres,
- Nous échangions sur nos problèmes, et si l'un d'entre nous avait une proposition, le problème était résolu facilement le lendemain matin, ce qui représentait un gain de temps et d'énergie,
- Le maintien des échanges contribuait à maintenir une atmosphère positive au sein de l'équipe.

7. L'équilibre. Nous avons constaté que nous étions parfaitement équilibrés avec deux membres spécialisés en front-end et deux en back-end, avec Ryan et Yannis pour le back-end, et Mel et moi pour le front-end.

Canva

En parallèle, j'ai commencé sur Canva une présentation. Je me suis dit que garder une trace écrite de l'avancée du projet pourrait être une bonne idée.

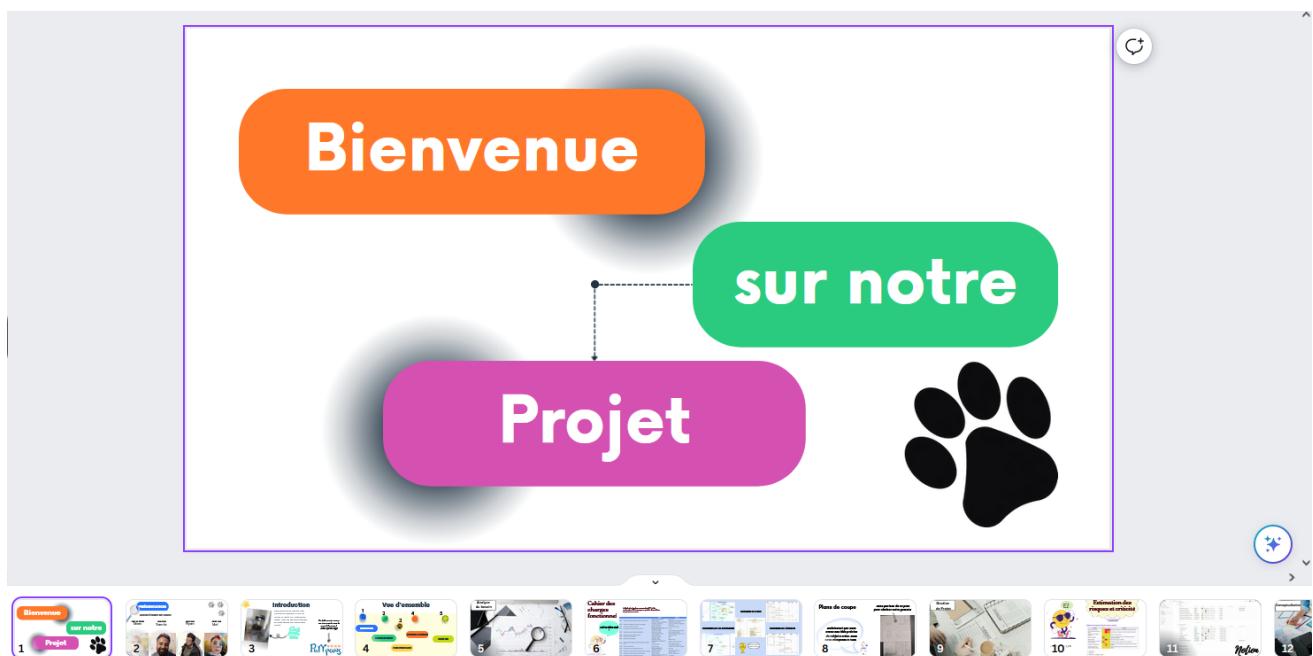
Mes collègues ont débattu sur l'utilité de tout noter, tout enregistrer, coucher sur papier ou Canva tous les détails de notre périple. Ils ont pensé à l'unanimité que cela n'était pas nécessaire au vu des demandes clients actuelles.

Mais de mon point de vue, il était important de noter nos difficultés, nos choix, nos créations diverses.

Dans un premier temps, parce que j'étais fier de créer quelque chose en partant de rien, et ensuite pour garder une trace des différentes options qui ont été choisies, tout simplement pour ne pas oublier par où nous étions passés pour arriver à ces résultats.

J'ai donc préféré être prudent et continuer malgré tout à mettre une quantité non exagérée de données de côté (pour le cas où).

J'en ai également profité pour jeter un œil sur le travail de mes collègues, et parfaire ma compréhension des langages divers utilisés. (**C21**)



Les débuts du travail d'équipe

Nous nous sommes répartis les tâches de la manière suivante, en vue de réaliser notre projet : (**C2, C7**)

Notion

Nous avons créé une section sur Notion. Notion est une plateforme de gestion de projet et de collaboration en ligne qui permet aux utilisateurs de créer, organiser et partager des informations de manière flexible.

Elle combine des fonctionnalités de prise de notes, de gestion de tâches, de création de bases de données et de collaboration en temps réel au sein d'une seule application (**C21**).

Nous avons réparti les tâches en fonction de nos forces et faiblesses. Étant le membre le moins difficile de l'équipe Front (le "Front-end" est, pour simplifier, la partie visible de l'application), j'ai laissé à ma collègue le soin d'attribuer les tâches et d'approuver ou de rejeter ses choix. J'ai finalement pris en charge les sujets les plus longs à réaliser, tandis que Mel a pris en charge ceux qui nécessitaient le plus de liens de redirection et de travail sur le CSS (le CSS est ce qui permet de modifier l'aspect visuel des différents éléments de l'application).

The screenshot shows a Notion workspace with a sidebar on the left containing navigation links like 'PurrPaws', 'General', 'Tasks', 'Projects', 'Pages privées', 'Modèles', 'Importer', and 'Corbeille'. The main area displays two tables of tasks:

Tasks Section:

| Aa Task name | Status | Assign | Due | Project | Description | Fichiers et médias |
|--|-------------|-------------------|---------------|-------------|-----------------------------------|--------------------|
| Appi React | 16 | ... | + + | | | |
| Page CGU | Not Started | E | El_Tr4ns1sToR | Appli React | | |
| Page FAQ | Not Started | E | El_Tr4ns1sToR | Appli React | | |
| Page Paramètres | Not Started | E | El_Tr4ns1sToR | Appli React | | |
| App.js/Finir de construire les pages (?) | In Progress | Rainbow Butterfly | | Appli React | | |
| Composant Pop up suppression | Done | Marc | | Appli React | | |
| Component Header | Done | Marc | | Appli React | Insérer dans le code le drawer c | |
| Component Nav bar | Done | Rainbow Butterfly | | Appli React | Bien penser à faire les condition | |
| Composant Delete Button | Done | Rainbow Butterfly | | Appli React | | |
| Component Return button | Done | Rainbow Butterfly | | Appli React | | |
| Component Modified button | Done | Rainbow Butterfly | | Appli React | | |
| Component Save button | Done | Rainbow Butterfly | | Appli React | | |
| Component Display | Done | Marc | | Appli React | | |
| Component See Order button | Done | Rainbow Butterfly | | Appli React | | |
| Component pop-up (layout) | Done | Rainbow Butterfly | | Appli React | | |
| Création de toutes les pages (vide) | Done | Rainbow Butterfly | | Appli React | | |
| Tous les routings entre page | Done | Rainbow Butterfly | | Appli React | | |

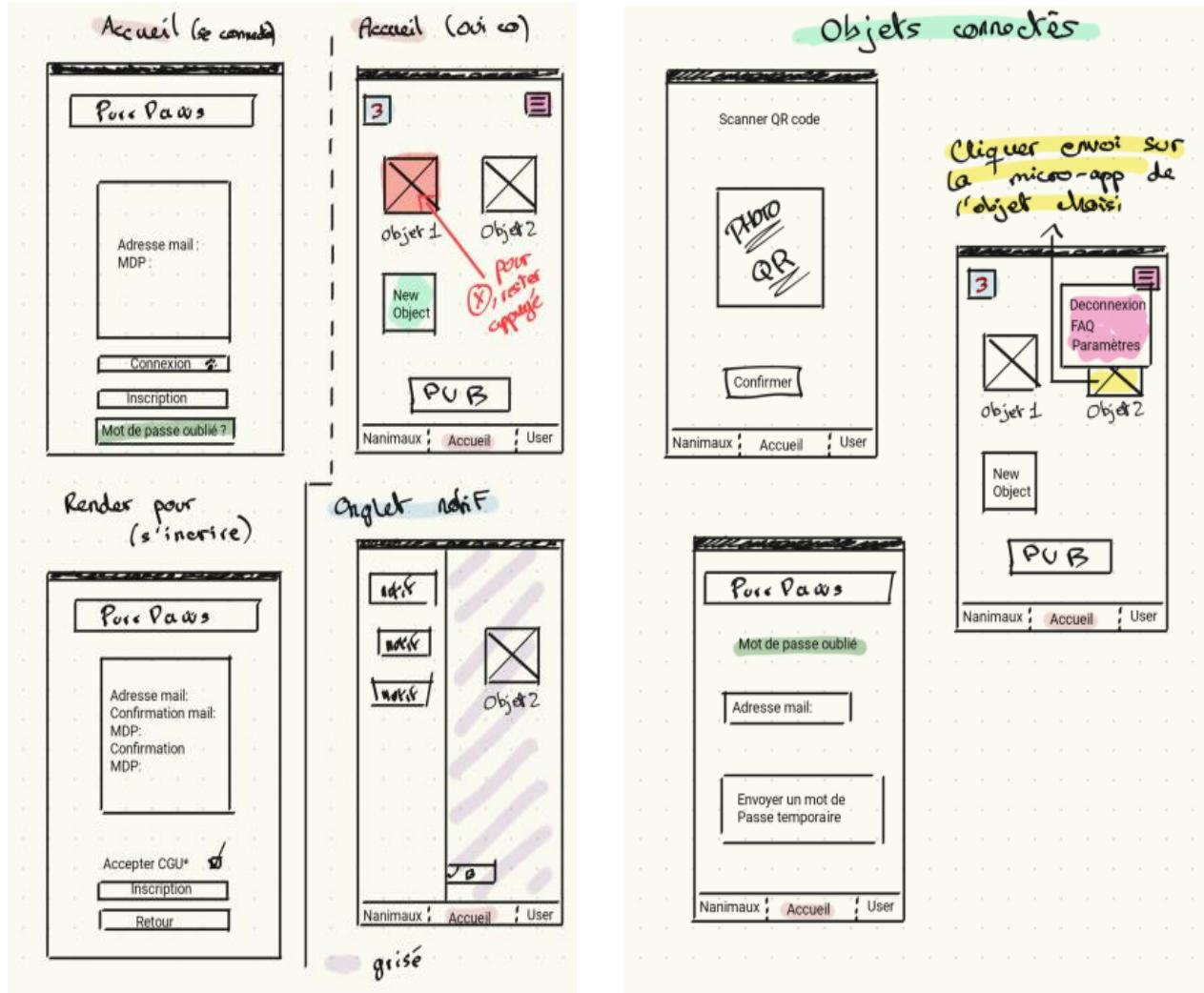
Design Section:

| Aa Task name | Status | Assign | Due | Project | Description | Fichiers et médias |
|-----------------------|--------|-------------------|------|--------------|-------------|--|
| Maquette Appli (site) | Done | Rainbow Butterfly | Marc | 2 avril 2023 | Design | https://www.figma... |

Wireframes

Les wireframes sont des schémas de conception initiale de l'interface de l'application. (C4)

Ils nous ont permis de visualiser à quoi ressemblerait l'application avant de passer à la phase de développement.



Maquettes

Les maquettes sont des représentations plus détaillées de l'interface, montrant les éléments de design, les couleurs et les emplacements des éléments de l'application.

Elles nous ont permis d'avoir une idée précise du rendu final de l'application.

Elles ont également soulevé une question essentielle, la charte graphique.

Nous avons choisi en couleur principale le bleu, et en couleur secondaire l'orange. Le bleu a été choisi d'un commun accord, la couleur du ciel, de la liberté.

Pour ce qui est de l'orange, Mel nous a indiqué que c'était une couleur chaleureuse qui répondait bien avec le bleu, nous avons donc tous accepté son choix sans plus de questions.

Et donc, voici nos maquettes :

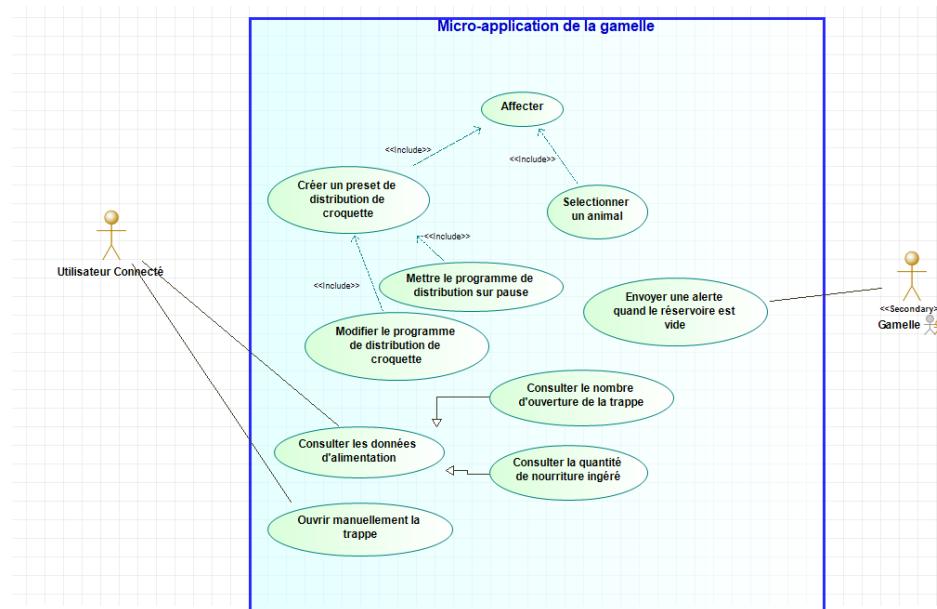


UML du projet

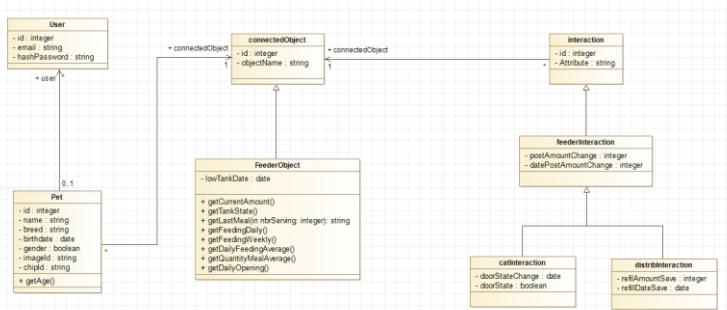
Nous avons utilisé l'**UML (Unified Modeling Language)** pour modéliser différents aspects du projet, notamment les diagrammes de cas d'utilisations, les diagrammes de séquences, la spécification des besoins, la Méthode d'Organisation et d'Analyse (**MOA**), le modèle conceptuel de données (**MCD**) et le modèle physique de données (**MPD**). (**C8, C9, C10, C11**)

Petite explication :

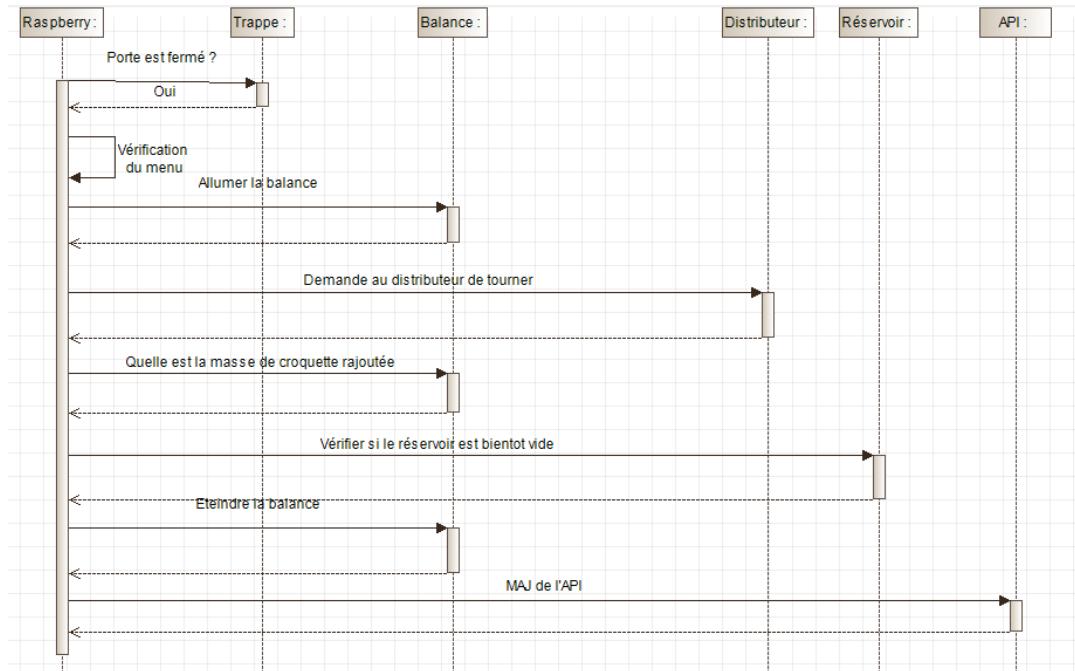
Diagrammes de Cas d'Utilisation : Les diagrammes de cas d'utilisation sont des outils de modélisation UML qui permettent de décrire les interactions entre un système (comme une application ou un site web) et ses utilisateurs. Ils montrent les différentes actions ou fonctionnalités que les utilisateurs peuvent effectuer dans le système et comment ces actions sont liées les unes aux autres. Ces diagrammes nous ont été utiles pour comprendre comment le système se comporte du point de vue de l'utilisateur.



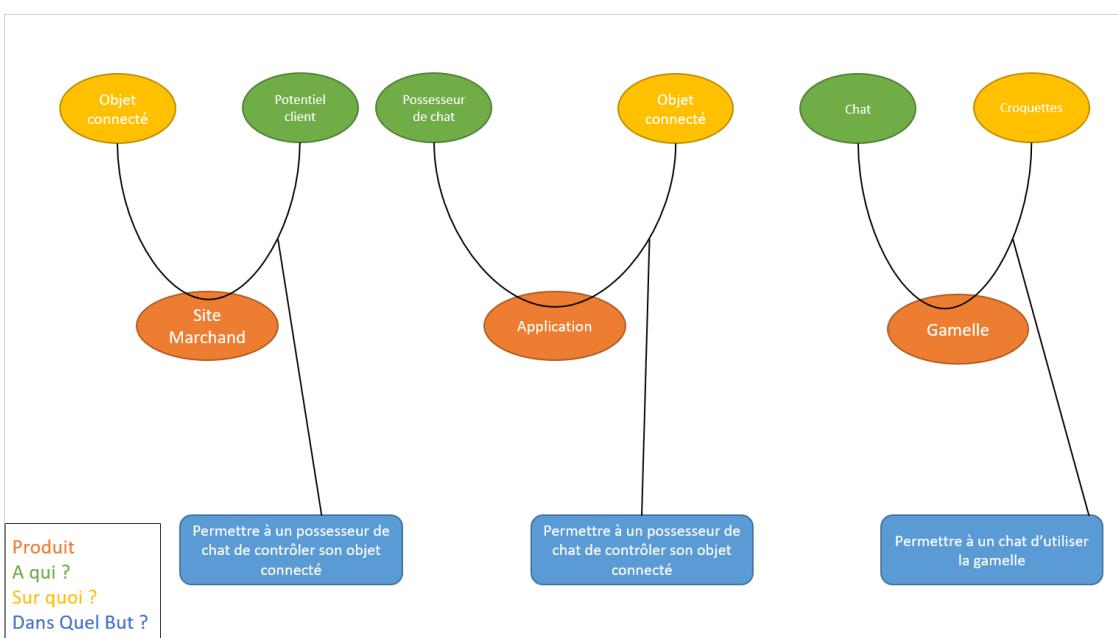
Le diagramme de classe : C'est un élément clé de la modélisation UML qui représente la structure statique d'un système logiciel, en mettant en évidence les classes, les attributs, les méthodes et les relations entre les classes. Il aide à visualiser comment les objets seront organisés et interconnectés dans le système. Le diagramme de classe est particulièrement utile pour les développeurs, car ils nous ont permis de planifier la conception du code et de s'assurer que toutes les classes et leurs relations sont bien définies avant de commencer la programmation.



Diagrammes de Séquences : Les diagrammes de séquences se concentrent sur la manière dont les objets interagissent dans un système et la chronologie de ces interactions. Ils montrent comment les différentes parties du système communiquent entre elles pour accomplir une tâche spécifique. Les diagrammes de séquences sont particulièrement utiles pour représenter les processus complexes requis par cette application.

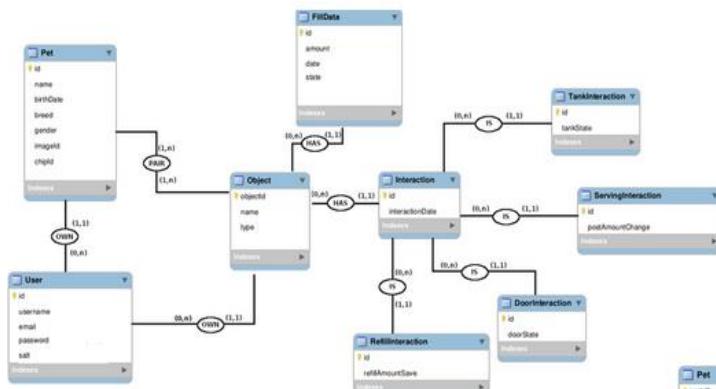


Méthode d'Organisation et d'Analyse (MOA) : La MOA est une approche méthodologique qui vise à organiser et analyser de manière structurée les processus, les données et les besoins au sein d'une organisation ou d'un projet. Elle aide à améliorer l'efficacité, la compréhension et la gestion des activités en utilisant des techniques de modélisation, de documentation et d'analyse pour prendre des décisions éclairées et optimiser les opérations. Cette vision simpliste nous a aidé à y voir plus clair sur qui fait quoi.



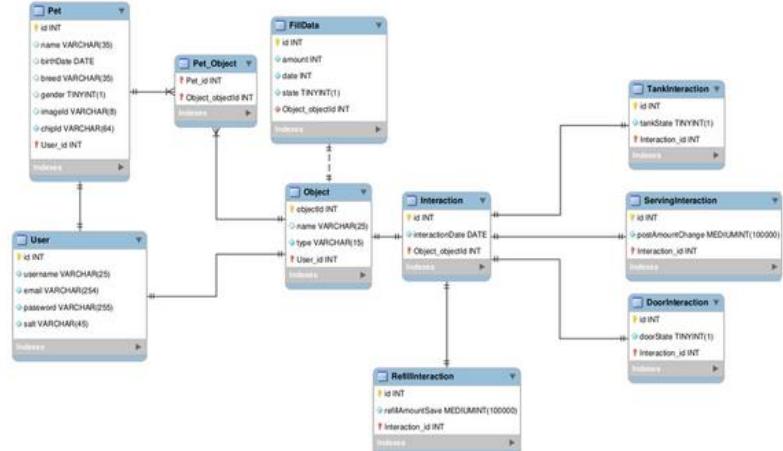
Modèle Conceptuel de Données (MCD) : Le MCD est une représentation abstraite des données et de leurs relations dans une base de données. Il se concentre sur la structure logique des informations, sans se soucier des détails techniques de stockage ou d'implémentation. Dans notre projet de formation, prenons l'exemple d'un **utilisateur**. Cet utilisateur contiendra les données de connexion. A côté de cela, nous aurons un chat qui, en mangeant normalement, alimentera les données relatives aux fréquences d'utilisations, ainsi que la possibilité de le lier à un utilisateur. Nous pourrons également inclure des informations sur les quantités de nourriture distribuées, les heures de distribution de croquettes et bien d'autres éléments.

Modèle Physique de Données (MPD) : Le MPD, quant à lui, représente la façon dont les données seront réellement stockées et organisées dans une base de données, en prenant en compte les contraintes techniques, les types de données, les index, les clés étrangères, etc. Le MPD traduit le MCD en une structure concrète qui peut être mise en œuvre dans une base de données spécifique.



Modèle Conceptuel des Données

Modèle Physique des Données



Dans le **MCD** (en haut à gauche) on s'occupe surtout de la logique dans la BDD. Par exemple, un utilisateur peut avoir plusieurs chats, mais un chat ne peut avoir qu'un seul utilisateur, ou bien un utilisateur peut avoir plusieurs objets, mais un objet ne peut avoir qu'un seul utilisateur.

En revanche dans le **MPD**, nous allons rentrer dans le détail en spécifiant bien le type de données qu'il y aura dans la BDD, par exemple, l'animal aura un ID en INT (nombre), un nom en VARCHAR (lettre) ou une date de naissance de type DATE (c'est un type de donnée très spécifique qui gère les dates selon un format prédéfini).

Ces différentes étapes de planification et de documentation ont contribué à assurer la cohérence et la clarté de notre projet tout au long de son développement, de la base de données à l'orientation que prendra le code de notre application.

Risques et Criticité

Tableau des risques éventuels et leurs criticités

| Risques | ? | ! | ⚠ | Gestion |
|--|---|---|----|---|
| De ne pas finir à temps | 3 | 4 | 12 | Être assidu, bien réaliser le retroplanning et respecter les sprints |
| De ne pas obtenir les pièces à temps | 2 | 4 | 8 | Commander sur amazone prime, vérifier les délais de livraisons |
| Ne pas trouver de solution technique | 1 | 4 | 4 | Soliciter une aide extérieur |
| Manque de budget | 3 | 1 | 3 | Prévoir des composant à bas prix, investissement personnel, recherche de sponsors |
| Manque d'outils | 1 | 4 | 4 | Faire appel à l'usine ou un fablab |
| Perte de temps du à la mise en commun des gits | 3 | 2 | 6 | Respecter des bonnes pratiques de travail collaboratif |
| Problème de cybersécurité | 1 | 3 | 3 | Respecter les bonnes pratiques de cybersécurité et faire intervenir chatGPT |
| Perte de temps du à un disfonctionnement imprévu | 2 | 2 | 4 | Tester le matériel au plus tôt, commander des pièces en plus, tests unitaires |

?: Probabilité (/4)

!: Gravité (/4)

⚠ : Criticité (/16) = (Probabilité x Gravité)

Un document "Risque et Criticité" est un outil de gestion de projet qui vise à identifier, évaluer et classer les risques potentiels associés à un projet. Voici une explication détaillée de ce qu'est ce document :

1. Identification des risques : Dans ce document, on recense tous les risques possibles qui pourraient affecter la réussite du projet. Ces risques peuvent être de différentes natures, comme des retards, des problèmes techniques, des contraintes budgétaires, des conflits internes, etc.

2. Évaluation des risques : Chaque risque est ensuite évalué en termes de probabilité de survenue (du pessimiste au plus optimiste) et d'impact potentiel sur le projet en cas de réalisation. On utilise souvent une échelle de notation pour ces deux critères.

3. Calcul de la criticité : Pour déterminer la criticité de chaque risque, on applique généralement une formule mathématique, qui combine les estimations pessimistes et optimistes. Cette criticité permet de hiérarchiser les risques en fonction de leur importance pour le projet.

Pour ce faire, nous avons donc appliqué cette formule mathématique simple :

$$(P + 2xR + O) / 4$$

J'explique :

- P représente l'estimation "Pessimiste", c'est-à-dire la probabilité que le risque se matérialise de la pire manière possible.
- R signifie "Réaliste", et il s'agit de l'estimation moyenne ou d'une évaluation basée sur une perspective pragmatique.
- O correspond à l'estimation "Optimiste", décrivant la probabilité que le risque se matérialise de la meilleure manière possible.

En utilisant cette formule, en prenant en compte à la fois l'avis pessimiste et optimiste, puis en ajoutant le résultat à deux estimations réalistes, nous obtenons un indicateur qui permet d'évaluer la probabilité globale d'occurrence d'un risque.

Nous divisons ensuite ce résultat par quatre pour normaliser l'évaluation et ainsi obtenir une mesure plus représentative de la probabilité du risque.

Cette méthode permet d'obtenir une estimation de la probabilité de manière équilibrée en tenant compte des perspectives pessimistes, réalistes et optimistes. Elle nous a donc aidés à mieux évaluer et gérer les risques dans notre projet.

4. Plan d'action: Une fois les risques évalués et classés, on élabore des plans d'action pour chaque risque critique. Ces plans décrivent les mesures préventives et correctives à prendre en cas de réalisation du risque, afin de minimiser son impact sur le projet.

5. Suivi et mise à jour : Le document "Risque et Criticité" n'est pas statique. Il est régulièrement mis à jour tout au long du projet pour refléter l'évolution des risques. On peut ainsi anticiper les problèmes potentiels et prendre des mesures proactives pour les gérer.

Pour résumer, le document "Risque et Criticité" que nous avons réalisé tous ensemble est un outil de gestion de projet essentiel qui permet d'identifier, évaluer et classer les risques potentiels, tout en élaborant des plans d'action pour minimiser leur impact sur la réussite du projet.

Chef de projet

Personnellement, ce fut un moment véritablement agréable d'échange entre coéquipiers.

J'ai ainsi pu évaluer la psychologie de mes collègues de travail, ce qui m'a beaucoup aidé dans la gestion du projet. J'ai pu voir où étaient les points bloquants pour eux, où étaient les incertitudes/doutes, et où se situaient leurs points forts, là où ils se sentaient le mieux ou ce qu'ils étaient sûrs de savoir/pouvoir faire.

C'est également à ce moment que je me suis rendu compte que, comparé à eux, j'étais très souvent le pessimiste du groupe. En effet, mes expériences passées me pousse à toujours penser qu'il y'a peut-être un élément surprenant qui peut intervenir et ralentir ou complexifier notre projet (Livraisons en retard ou bien portefeuille de l'IT qui aurait des délais par exemple).

C'est en partie grâce à cet échange que j'ai pu établir une **stratégie à court et moyen terme** pour atteindre notre objectif. Je me suis aperçu que la cohésion de l'équipe devait être gérée avant même de pouvoir développer notre application. Je pourrai débattre plus avant de tout cela pendant la présentation de ma soutenance si vous le souhaitez car c'est un long sujet qui mériterai bien plus qu'une simple partie de page.

Nous avons aussi fait le **cahier des charges**, pour expliquer très simplement, qui décrit de manière détaillée les besoins, les objectifs, les spécifications et les contraintes de notre projet IOT. Il définit clairement ce qui doit être fait, mais une image vaut mille mots donc :

Cahier des charges

Cahier des charges fonctionnel

(détails des fonctions principales (FP) et de contraintes (FC) ainsi que ce qu'elles doivent faire ou permettre de faire)

Qui va faire quoi ?



| | | Critère | Niveau |
|-------|--|--|--|
| FP1.1 | Permettre à l'utilisateur d'ajouter et gérer un objet connecté | Ajout de l'objet à l'application Gestion de l'objet Connexion entre l'objet et l'application | par QR code Mini application dans l'application global Architecture IOT-server-client |
| FP1.2 | Permettre l'enregistrement d'animaux de compagnies | Enregistrer via l'application | Scanner la puce RFID via l'application |
| FC1.1 | Doit vérifier que l'objet soit connecté | Créer un profil animal Maintenir une connexion | ID RFID, nom ... Ping toutes les 30 minutes |
| FC1.2 | Doit respecter les bonnes pratiques de green IT | Réduire au minimum les appels à l'API et la taille des requêtes | 20 / jour |
| FP2.1 | Permettre à un chat de manger des croquettes | Ouverture de la trappe Fermeture de la trappe Programme d'apprentissage | Par reconnaissance de la puce RFID Après 5 seconde d'inactivité 4 niveaux d'apprentissage pour le chat |
| FC2.1 | Doit vérifier que la puce RFID correspond | Lecture de la puces | des 2 côtés du chat |
| FC2.2 | Doit être connectée à internet | Protocole de connexion | WIFI 6 |
| FC2.3 | Doit être fourni en électricité | Alimentation secteur Alimentation auxiliaire | 12 V Batterie ou piles |
| FC2.4 | Doit contenir des croquettes et informer l'utilisateur si quantité basse | Ouverture pour le remplissage de croquette Capteur de masse | Clapet sur le dessus du réservoir Type contacteur à ressort 2 niveaux (plein, presque vide) |
| FC2.5 | Doit pouvoir être nettoyé | Pièces amovibles | Le réservoir, la vis sans fin, la gamelle |
| FC2.6 | Doit respecter les bonnes pratiques de green IT | Diminuer au maximum la consommation électrique | mode veille |
| FP3.1 | Permettre à un client potentiel de consulter et acheter un produit | Consulter les produits | 1 Présentations attractives minimum 3 étapes |
| FC3.1 | Doit pouvoir accepter plusieurs modes de paiement | Ajouter aux paniers | Carte bleu, paypal, bitcoin |
| FC3.2 | Doit respecter les bonnes pratiques de green IT | Réduire au minimum les appels à l'API et la taille des requêtes | 3 / paiements |

GitHub

The screenshot shows the GitHub repository page for `appli-web-IOT`. The repository is private and was forked from `PurrPaws/appli-web`. It has 7 branches and 0 tags. The main branch is 1 commit behind `PurrPaws:main`. The repository contains 169 commits from `admin` and `admin`. The commits are listed in descending order of age, with the most recent being "ajout de l'adresse dans setupTests.js" 5 months ago. Other files listed include `public`, `src`, `.env`, `.gitignore`, `README.md`, `package-lock.json`, and `package.json`. The README file describes the project as "README de notre projet PurrPaws, pour la partie web-front sous reactJS.". The repository has 0 watching and 2 forks. The About section indicates it's an Application Web. The Languages section shows a breakdown: JavaScript (83.7%), SCSS (14.7%), HTML (1.1%), and CSS (0.5%).

GitHub est un outil de gestion de version qui permet aux développeurs de collaborer et de suivre les modifications apportées à leur code source. On y enregistre dans un **repository** la presque totalité de notre dossier. Je précise que vous pouvez très bien avoir plusieurs projets en même temps et les voir sur votre GitHub. Nous avons donc créé le GitHub du projet IoT, ajouté les fichiers ` `. **Git ignore` et `README.md`**, ainsi que configuré le fichier ` `package.json` avec certaines dépendances, comme par exemple :

"**axios": "1.4.0"** ===> Bibliothèque HTTP pour les requêtes réseau.

"**dotenv": "16.3.1"** ===> Gestion des variables d'environnement.

"**expo": "48.0.9"** ===> Permet de voir via l'application « expo go » sur notre tel mobile le résultat du code.

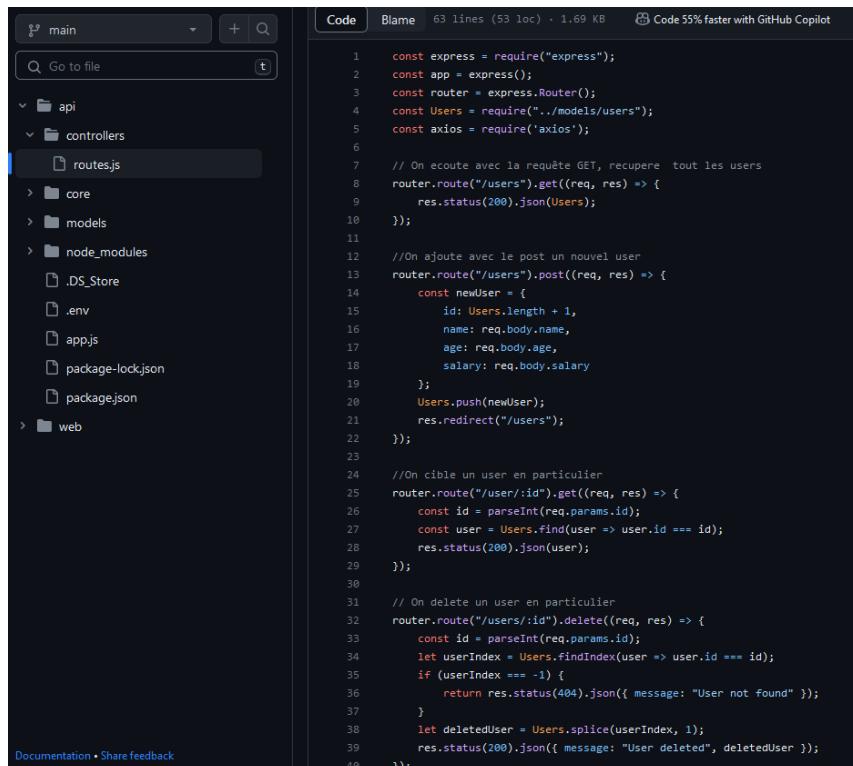
"**path": "0.12.7"** ===> Utilitaire pour la gestion des chemins de fichiers.

"**react": "18.2.0"** ===> Langage React avec sa dernière version (à l'initiation du projet).

- **.gitignore** : Un fichier .gitignore est utilisé dans les projets Git pour spécifier les fichiers et répertoires qui doivent être ignorés par le système de contrôle de version Git. (Plus de détails [page 55](#))

- **README** : Le fichier README est un document texte placé à la racine d'un projet logiciel. Il sert à fournir des informations essentielles sur le projet, telles que son but, son fonctionnement, les étapes d'installation, l'utilisation de base, et d'autres détails pertinents. Les README sont couramment utilisés pour informer les utilisateurs et les collaborateurs sur la manière de travailler avec le projet et sont souvent affichés sur la page d'accueil du référentiel sur des plateformes comme GitHub.

- **package.json** : Le fichier package.json est un fichier de configuration utilisé dans les projets Node.js. Il contient des métadonnées sur le projet, telles que son nom, sa version, ses dépendances (bibliothèques externes nécessaires au projet), des scripts personnalisés, et d'autres informations liées au projet. Ce fichier est essentiel pour la gestion des dépendances, la construction du projet, l'exécution de scripts, et facilite la distribution et la gestion des packages Node.js.



The screenshot shows the GitHub interface with the file structure on the left and the code content on the right. The file structure includes main, api, controllers (with routes.js selected), core, models, node_modules, .DS_Store, .env, app.js, package-lock.json, package.json, and web. The code content is as follows:

```
const express = require("express");
const app = express();
const router = express.Router();
const Users = require("../models/users");
const axios = require('axios');

// On écoute avec la requête GET, récupère tout les users
router.route("/users").get((req, res) => {
  res.status(200).json(Users);
});

//On ajoute avec le post un nouvel user
router.route("/users").post((req, res) => {
  const newUser = {
    id: Users.length + 1,
    name: req.body.name,
    age: req.body.age,
    salary: req.body.salary
  };
  Users.push(newUser);
  res.redirect("/users");
});

//On cible un user en particulier
router.route("/user/:id").get((req, res) => {
  const id = parseInt(req.params.id);
  const user = Users.findIndex(user => user.id === id);
  res.status(200).json(user);
});

// On delete un user en particulier
router.route("/users/:id").delete((req, res) => {
  const id = parseInt(req.params.id);
  let userIndex = Users.findIndex(user => user.id === id);
  if (userIndex === -1) {
    return res.status(404).json({ message: "User not found" });
  }
  let deletedUser = Users.splice(userIndex, 1);
  res.status(200).json({ message: "User deleted", deletedUser });
});
```

Ensuite, nous avons commencé à coder nos premières lignes dans `App.js`, qui est la pierre angulaire de notre application. Nous importons dans un premier temps tous les langages et dépendances nécessaires pour que le code ne manque de rien. Nous importons également toutes les routes qui redirigeront l'utilisateur à l'endroit souhaité, et donc les pages principales de l'application ainsi que les pages de toutes les micro-applications.

```
1  import React from 'react';
2  import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
3  import AddObject from './pages/AddObject';
4  import AddPet from './pages/AddPet';
5  import CGU from './pages/CGU';
6  import Faq from './pages/Faq';
7  import FollowOrder from './pages/FollowOrder';
8  import ForgottenPassword from './pages/ForgottenPassword';
9  // Import des pages de l'app principale
10 import Home from './pages/Home';
11 import LastOrder from './pages/LastOrder';
12 import Login from './pages/Login';
13 import OrderDetail from './pages/OrderDetail';
14 import Parameters from './pages/Parameters';
15 import Pet from './pages/Pet';
16 import PetDetail from './pages/PetDetail';
17 import SignIn from './pages/SignIn';
18 import User from './pages/User';
19 import BaseTemplate from './component/BaseTemplate.js';
20 // Import des pages MicroAppGamelle
21 import MicroAppHome from './microAppFeeder/MicroAppHome';
22 import DetailPageFeeder from './microAppFeeder/DetailPageFeeder';
23
24 export default function App() {
25
26  return (
27    <>
28      <Router>
29        <BaseTemplate />
30        <Routes>
31          {/*Partie principale de l'app*/}
32          <Route path="/connexion" element={<Login />} />
33          <Route path="/recuperationMdp" element={<ForgottenPassword />} />
34          <Route path="/inscription" element={<SignIn />} />
35          <Route path="/accueil" element={<Home />} />
36          <Route path="/accueil/ajoutObjet" element={<AddObject />} />
37          <Route path="/accueil/parametres" element={<Parameters />} />
38          <Route path="/accueil/faq" element={<Faq />} />
39          <Route path="/utilisateur" element={<User />} />
40          <Route path="/utilisateur/suivrecommande" element={<FollowOrder />} />
```

Nous avons également créé plusieurs règles en suivant au mieux ce qui nous a été enseigné, notamment :

DevOps : Le DevOps est une méthode de développement logiciel qui vise à améliorer la collaboration et la communication entre les équipes de développement (**Dev**) et les équipes d'exploitation (**Ops**).

Cette approche met l'accent sur l'automatisation, la mesure et le partage des informations pour atteindre un déploiement plus rapide et plus fiable des logiciels. Yannis s'occupait de toute la partie infra de la gamelle, j'ai donc tout fait pour que son travail soit cohérent avec nos fonctionnalités et que, de notre côté, nos fonctions s'adaptent le plus possible au matériel utilisé.

Création de branches localisées en fonction du travail à faire, selon la logique suivante :

- Actions à faire, par exemple :

feature: Ajout d'une nouvelle fonctionnalité

Bugfix: Correction d'un bug

- Travail ciblé, par exemple :

ModificationsBoutons

AjoutHeader

Cela donne un résultat qui ressemble à :

- **feature/ModificationsBoutons**

- **bugfix/header**

Avant de faire une pull-request, il fallait que le/la coéquipier/ère accepte le push de l'autre.

Nous avons donc ajouté la règle de faire une revue de code ensemble (**code review**) et de vérifier si tout était exactement comme il fallait, tant au niveau de la propreté du code, de l'efficacité énergétique (green IT), de la sécurité, des commentaires insuffisants, etc.

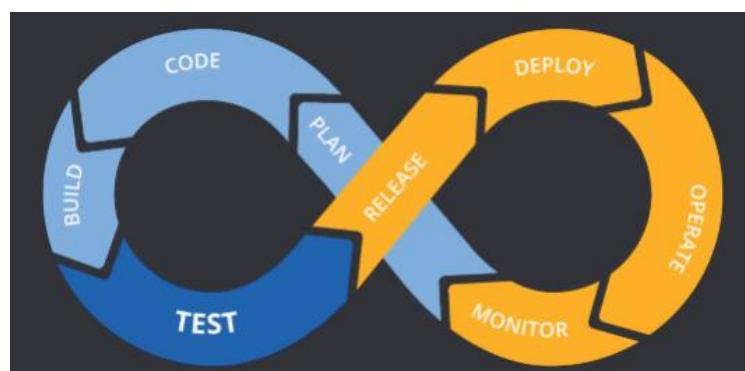
Et c'est seulement une fois validé que nous pouvions pull le code et continuer à avancer.

De plus, nous nous sommes efforcés de ne pas toucher aux codes de l'autre, isoler nos travaux est donc devenu une pierre angulaire de notre travail d'équipe. Si une retouche devait être faite, c'est celui ou celle qui l'a fait qui devra s'en occuper.

CI/CD

La CI (**Intégration Continue**) est une pratique de développement où les développeurs intègrent fréquemment leur code dans un référentiel partagé. (**C 20**)

Le CD (**Livraison Continue**) va plus loin en automatisant la mise en production du code intégré et testé.



Une fois que le code a passé les tests automatisés, il est prêt à être déployé en production sans étapes manuelles supplémentaires. Pour notre projet, nous n'avons pas eu le temps de l'appliquer comme nous le voulions mais j'ai eu l'occasion d'en faire en entreprise (Voir plus loin dans le dossier de soutenance **page 60**).

Maintien de Scrum

J'ai essayé de pratiquer le Scrum en répartissant le travail de la manière la plus efficace possible au sein de l'équipe,

- en déléguant certaines tâches,
- en définissant l'objectif à atteindre,
- en maintenant les sprints agiles des deux équipes (l'une front, l'autre back).

Je tiens à préciser que je ne suis pas Scrum Master, mais plutôt, pour le moment, un Scrum But (dit autrement, un débutant). Mes compétences dans ce domaine ont été apprises lors de mes anciennes responsabilités et consolidées par les cours que nous avons eu au cours de l'année.

La distinction entre Scrum Master, Chef de projet et Chef d'équipe n'est toujours pas très évidente à mettre en pratique à ce stade, mais je tiens à remercier toute l'équipe pour son sérieux, son dévouement et sa bienveillance au quotidien, des aspects qui ont été hautement appréciés par chacun d'entre nous.

Pour la répartition, **Yannis** s'occupait de la partie "industrielle" de l'objet,

Ryan a configuré le Raspberry Pi pour qu'il serve d'intermédiaire entre la machine et le back-end (le back-end, pour simplifier comme le Front-end, gère le traitement des données recueillies par la gamelle ou indiquées par l'utilisateur via l'application mobile).

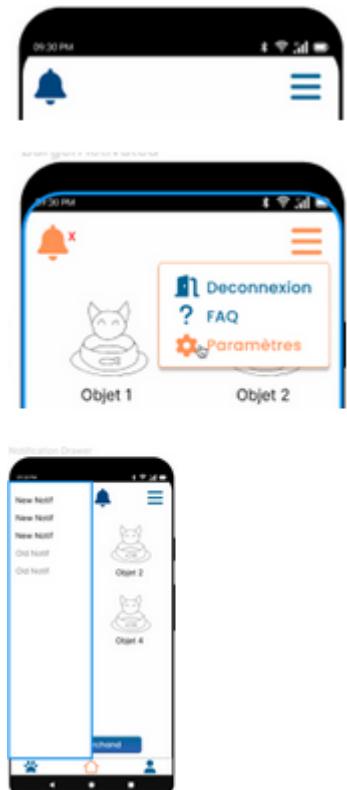
Mel a organisé, en tant que frontLead, les composants de l'application et ajouté les tâches à faire sur Notion.

Quant à moi, je me suis concentré sur les demandes des clients telles que :

- Restrictions budgétaires pour l'objet,
- Vérification des obligations linguistiques,
- Vérification des contraintes matérielles,
- Faire le lien entre les demandes client et nos capacités actuelle.

Une fois ces échanges conclus sur des accords mutuels, j'ai suivi les tâches qui m'ont été attribuées via Notion, j'ai commencé à réaliser certains composants essentiels, tels que le header et les différentes pages de connexion. Pour ce faire, nous nous sommes basés sur les maquettes que nous avions faites au préalable.

Modéliser une maquette



J'ai commencé par le composant Header.js. Cette partie de l'application contient une cloche de notification et un bouton de connexion à l'état fermé de base.

Il y avait plusieurs détails à prendre en compte, le premier étant l'emplacement (en haut), le deuxième étant la couleur. Nous avons donc appliqué notre charte graphique.

Les couleurs ont été ajoutées en tant que variables dans le SCSS, que nous avons nommées "**PrimaryColor**" (pour le bleu) et "**SecondaryColor**" (pour l'orange).

Si le bouton du header est cliqué et/ou actif, alors il devient orange. Sinon, son état "naturel" est le bleu.

La cloche des notifications, si elle est cliquée, déroule une liste de notifications et s'ouvre lentement de la gauche vers le centre de l'écran.

Le bouton burger, lui, doit afficher un menu avec à l'intérieur des textes cliquables qui redirigent vers la page voulue.

Nous avons donc créé des composants React qui sont, par nature, réutilisables. Un seul composant de l'en-tête (**header**) sera donc utilisé sur toutes les pages de l'application. Nous avons également réutilisé le composant "button" dans toutes les pages, mais l'un s'appelle "Retour" tandis que le même bouton à un autre endroit s'appelle "Accepter". Il s'agit toujours du même bouton, mais réutilisé de manière différente.

Nous avons appliqué ce principe en respectant ainsi le Green IT imposé par l'école et en automatisant le code. (C12, C14)

Composant Header.js

```
1 import React from "react";
2 import "./style/header.scss";
3 import { useLocation } from "react-router-dom";
4 import BurgerButton from "./BurgerButton";
5 import NotificationButton from "./NotificationButton";
6
7 const Header = () => {
8   const location = useLocation();
9
10  return (
11    <div className="header">
12      <NotificationButton
13        location={location}
14      />
15      <BurgerButton
16        location={location}
17      />
18    </div>
19  );
20};
21 export default Header;
```

Header.scss

```
1      @import "./utils/variable.scss";
2      @import "./mixin/burgerIcons";
3
4      .header {
5          padding: 10px;
6          height: 55px;
7          display: flex;
8          justify-content: space-between;
9          list-style: none;
10         position: relative;
11
12         &__notifButton {
13             border: none;
14             background-color: white;
15             cursor: pointer;
16         }
17         &__notifInact {
18             margin-left: 6px;
19             margin-top: 10px;
20             width: 30px;
21             height: 35px;
22             transition: all 1s ease-in-out;
23
24             &.show {
25                 margin-left: 206px;
26             }
27         }
28         &__drawer {
29             position: fixed;
30             top: 0;
31             left: 0;
32             height: 100%;
33             background-color: white;
34             box-shadow: 2px 0 10px rgba(0, 0, 0, 0.3);
35             transition: all 1s ease-in-out;
36             overflow: hidden;
37             z-index: 999;
38
39         & > * {
40             transition: opacity 1s ease-in-out;
41             width: 180px;
```

Visuels de l'application

J'ai commencé à faire un formulaire sur l'application en respectant, bien sûr, les couleurs, le logo et les liens de redirection. **(C13)**

Il faut bien comprendre que dès que l'on clique sur un simple bouton, au niveau du code qui va interpréter cette action, il y'a toute une logique qui s'exécute en arrière pour faire en sorte de, par exemple, changer de pages, ou alors de pouvoir enregistrer son mail.

Sur l'image suivante, j'ai fait la page de connexion. Elle se compose de deux éléments principaux, le logo en haut et le formulaire au centre.



Comme expliqué plus haut, le formulaire est un composant, les boutons sont des composants et les champs cliquable ou l'utilisateur entrera ses informations personnelles sont également des composants.

J'ai rencontré plusieurs problèmes inattendus au niveau du **CSS** et des placements des différents composants, mais avec quelques paramètres modifiés, le rendu visuel était impeccable.

En revanche, pour les liens de redirection (connexion/inscription) j'ai eu du mal à envoyer les informations reçues dans les champs au Back End. Nous avions décidé d'utiliser une API pour l'échange de données.

API

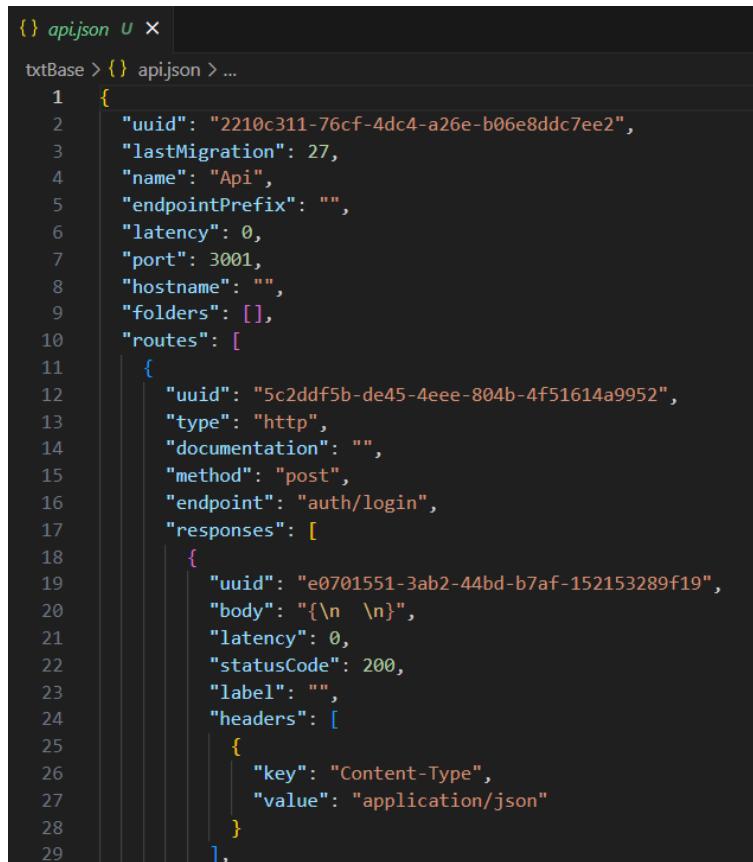
En parallèle, Ryan et Yannis ont commencé le Back-end, et il nous fallait une liaison entre le Back et le Front. Nous avons donc décidé de créer une **API REST** qui enverra les données reçues de l'utilisateur via le Front-end de l'application vers le Back-end, qui lui traitera ou sauvegardera ces données.

Je vais tenter de vous expliquer ce qu'est, selon moi, une API. J'utiliserais donc une analogie. Il faut voir une API comme un bus qui transporte des gens. Sa mission est d'emmener des personnes d'un endroit « A » à un endroit « B »:

1. **Le Bus** : Une API est comme un bus qui transporte des données, mais il suit un ensemble spécifique de règles et de conventions pour gérer ces données.
2. **Les Passagers** : Les données transportées par une API sont comme les passagers dans le bus. Chacune d'entre elles sont identifiées par une adresse (URI), et ces données peuvent être de différents types, tels que des textes (txt), des images (JPEG), des vidéos (MPEG).
3. **L'Arrêt** : L'arrêt du bus représente une ressource sur le serveur web qui peut être consultée via une adresse URL. Par exemple, une API pourrait avoir un arrêt (ressource) appelé "/users" qui représente une liste d'utilisateurs.
4. **Le Conducteur** : L'API RESTful est comme le conducteur du bus, son comportement peut être régi par les méthodes HTTP standard, notamment GET (pour la lecture), POST (pour la création), PUT (pour la mise à jour), DELETE (pour la suppression). Chacune de ces méthodes correspond à une action spécifique que l'API peut effectuer.
5. **Le Passage des Passagers** : Les passagers (les données) montent et descendent du bus en utilisant les méthodes HTTP appropriées. Par exemple, pour obtenir la liste des utilisateurs, on peut envoyer une requête GET à l'adresse "/users".
6. **Les Actions de l'Agence de Voyage (Back-end)** : L'agence de voyage (Back-end) comprend les demandes des passagers (les requêtes HTTP) et réagit en conséquence. Par exemple, si une requête demande l'ajout d'un nouvel utilisateur, l'API effectuera l'action appropriée pour créer cet utilisateur selon le traitement du Back-end.
7. **Le Choix du Passager** : Les passagers (les clients) peuvent choisir de demander des informations spécifiques auxquelles ils sont intéressés. Par exemple, ils peuvent demander de filtrer une liste d'utilisateur (/users) en fonction de certains critères (seuls ceux qui sont majeurs peuvent passer).
8. **La Destination** : Le service qui reçoit les données est la destination finale où les données sont traitées. Le service qui aura besoin de recevoir les données de notre API peut s'il le souhaite choisir de traiter toutes les données ou seulement celles qui l'intéressent en fonction des requêtes (si les données ont plus de 6 mois, alors je ne les prends pas).

Donc chaque fois que vous entendrez parler d'une API, pensez à un bus rempli d'informations.

Api.json



```
{ apijson u x
txtBase > {} apijson > ...
1  {
2    "uuid": "2210c311-76cf-4dc4-a26e-b06e8ddc7ee2",
3    "lastMigration": 27,
4    "name": "Api",
5    "endpointPrefix": "",
6    "latency": 0,
7    "port": 3001,
8    "hostname": "",
9    "folders": [],
10   "routes": [
11     {
12       "uuid": "5c2ddf5b-de45-4eee-804b-4f51614a9952",
13       "type": "http",
14       "documentation": "",
15       "method": "post",
16       "endpoint": "auth/login",
17       "responses": [
18         {
19           "uuid": "e0701551-3ab2-44bd-b7af-152153289f19",
20           "body": "{\n  \n}",
21           "latency": 0,
22           "statusCode": 200,
23           "label": "",
24           "headers": [
25             {
26               "key": "Content-Type",
27               "value": "application/json"
28             }
29           ]
29         ]
29       ]
29     ]
29   ]
29 }
```

"uuid": Ceci définit un identifiant unique pour l'API, qui est utilisé pour l'identifier.

"name": Il s'agit du nom de l'API, qui est simplement défini comme "Api", nous nous sommes dit que s'il fallait préciser plus tard, nous le ferions mais au final, « Api » pour un nom d'API reste cohérent. Simple mais cohérent. Mais simple, mais Bref vous avez saisi l'idée.

"endpointPrefix": C'est un préfixe d'URL pour les points de terminaison de l'API. Pour l'instant, il est défini comme une chaîne vide, ce qui signifie que les points de terminaison utiliseront la racine de l'URL.

"latency": La latence, définie à 0, indique qu'il n'y a pas de délai artificiel spécifique défini pour l'API. Cela signifie que les réponses devraient être aussi rapides que possible. Si d'aventure nous voulions faire patienter la réponse, on modifierait cette donnée au-dessus de 1.

"port": C'est le numéro de port sur lequel l'API écoutera les requêtes HTTP. Dans ce cas, il est configuré pour écouter sur le port 3001 en Localhost, ce qui signifie qu'il est en local sur nos postes de travail mais pas en ligne.

"hostname": Le nom d'hôte auquel l'API sera liée. S'il est défini comme une chaîne vide, cela signifie qu'il écoutera sur toutes les interfaces disponibles de la machine.

"routes" : Ceci est une liste de routes ou de points de terminaison pour l'API. Chaque élément de la liste décrit une route spécifique avec ses propriétés. Il y a plusieurs routes configurées dedans mais par soucis de place (une photo de 8 pages aurait été un poil excessif je pense) je vais résumer la première d'entre elles.

"type": Le type de la route, qui est HTTP dans ce cas. Les programmes utilisent des protocoles pour discuter entre eux. Il existe par exemple HTTP (World Wide Web), FTP (transfert de fichiers), SMTP (messagerie), SSH (connexion à distance sécurisée) et pleins d'autres.

"method": La méthode HTTP associée à cette route, qui est "POST" dans ce cas, cela veut dire que cette route va s'occuper des **ajouts** en base de données.

"endpoint": L'URL ou le point de terminaison de cette route, qui est "auth/login", donc en suivant les informations vues avant, on va **ajouter** grâce à cette route un **login** en base de données.

"headers" : Une liste de paires **key = value** (clé = valeur) pour les en-têtes de la réponse.

"key": La clé de l'en-tête, qui est "Content-Type".

"value": La valeur de l'en-tête, qui est "application/json", indiquant que le contenu de la réponse est au format JSON.

Le **JSON (JavaScript Object Notation)** est un format de données léger utilisé pour structurer et échanger des données entre applications. Il est facile à lire pour les humains et à analyser pour les machines, ce qui en fait un choix populaire pour la communication de données sur le web.

Donc notre **API** utilise le format **JSON**.

Nous avons avancé ainsi pendant plusieurs jours, en nous adaptant aux différentes nécessités du moment.
(C15 C17 C18 C21)

Les réunions devenaient de plus en plus précises et habituelles, le dialogue entre nous s'est installé de manière naturelle, et la progression était très satisfaisante.

Ryan était responsable d'un suivi pour conserver nos évolutions :

Compte rendu 04/04 (Jour Projet)

Marc: composant résumé/display objet connecté, aussi utilisé pour les animaux.

Plan pour demain: header notification → notification
burger → drawer

Ryan: Mock JSON & MERISE (dictionnaire de données)

Plan pour demain: MCD MPD données calculées à rajouter au dictionnaire de données

Mel: NavBar -changement de couleur en fonction de la page (algorithme avec usestate)

Plan pour demain: tuto sur les routes. Boutons (CSS, recherche bibliothèque react)

Yannis: Circuit électronique, antenne planifiée mais composants pas assez bien calibrés. Liste du matériel presque prête. Courant alternatif testé.

Plan pour demain: test du moteur pas à pas. Modélisation 3D. Diagramme de classe

Compte rendu 05/04 (Jour Projet)

Mel: Boutons et tuto sur les routes

Plan pour demain: component ajout objet avec QR code

Ryan: MERISE – mise à jour dictionnaire de données, MCD

Plan pour demain: MCD au propre, MPD et finir les mockJSON

Marc: Buttons et burger. Mis à la bonne taille des icônes, css on click. Burger -> icônes et boutons.

Plan pour demain: finir burger button et modal.

Yannis : mise à jour dictionnaire de données, conception diagramme de classes

Plan pour demain: modélisation 3d et fin diagramme de classes

Sans jamais oublier de mettre à jour notre Notion.

Homepage Appli React 5 ... +

| Aa Task name | % Status | Assign | Due | Project | Description | Fichiers et médias |
|---------------------------------------|----------|-------------------|-----|----------------------|-------------|--------------------|
| Composant Store Button | Done | Rainbow Butterfly | | Homepage Appli React | | |
| Composant Pop up Notif | Done | Rainbow Butterfly | | Homepage Appli React | | |
| Page Add Object | Done | Rainbow Butterfly | | Homepage Appli React | | |
| Page HomePage | Done | Rainbow Butterfly | | Homepage Appli React | | |
| Composant Add Object manualy (pop-up) | Done | Rainbow Butterfly | | Homepage Appli React | | |

+ Nouvelle page

TERMINÉ 5/5

Log-in/Sign-in Appli React 5 ... +

| Aa Task name | % Status | Assign | Due | Project | Description | Fichiers et médias |
|--------------------------|----------|--------|-----|----------------------------|-------------|--------------------|
| Page Log-in | Done | Marc | | Log-in/Sign-in Appli React | | |
| Page Sign-in | Done | Marc | | Log-in/Sign-in Appli React | | |
| Page Forgotten Password | Done | Marc | | Log-in/Sign-in Appli React | | |
| Composant Form (log-in) | Done | Marc | | Log-in/Sign-in Appli React | | |
| Composant Form (sign-in) | Done | Marc | | Log-in/Sign-in Appli React | | |

+ Nouvelle page

TERMINÉ 5/5

Il nous fallait donc simuler notre API puis la tester. Mais un jour, un mardi matin (pour être précis), nos objectifs de la semaine furent chamboulés.

Imprévu

L'école nous a envoyé un mail pour nous demander une Oui oui, une présentation !

Et là, je ne vous cache pas que je me suis retenu au maximum, mais la phrase

"HAHA Je vous l'avais bien dit !!" a été sortie.

Il nous a donc fallu reporter l'avance de notre API pour nous consacrer à plein temps à cette présentation car il faut le savoir, la date limite pour la présentation était de seulement deux jours (présentation le vendredi même).

Et quel bonheur d'avoir préparé, noté et illustré cette présentation Canva tout au long du projet, à chaque nouvelle image, chaque nouvelle étape, tout était mis de côté, pour le cas où.

Ce qui s'est avéré être au final non pas du temps perdu mais bel et bien une étape du projet réglée avant même que le client (l'école) y est pensé.

Heureusement pour nous, environ 80% du travail avait déjà été réalisé, il nous restait principalement à définir le contenu verbal, à décider qui prononcerait quel discours, et à ajouter quelques slides supplémentaires pour plus de détails.

Exemple de slides ajoutées:

Etude de marché



Etude de marché

- ▶ 1. Présenter entreprise
- ▶ 2. Définition positionnement
- ▶ 3. PESTEL
- ▶ 4. SWOT
- ▼ 5. PORTER

| Force de Porter | Description | Indice sur 5 |
|---|--|--------------|
| 1. Concurrence intra-sectorielle | Le nombre et la puissance des concurrents sur le marché des gammes connectées en fonction de la différence de fonctionnalités. | 3 |
| 2. Menace des nouveaux entrants | La facilité ou non de nouveaux acteurs d'entrer sur le marché des gammes connectées | 3 |
| 3. Pouvoir de négociation des fournisseurs | Le pouvoir de négociation des fournisseurs de matières premières et objets connectés PurPaws | 3 |
| 4. Pouvoir de négociation des clients | Le pouvoir de négociation des clients sur les prix et la qualité des gammes connectées | 2 |
| 5. Menace des produits de substitution | La possibilité pour les clients d'acheter des gammes avec fonctions similaires | 5 |
| 6. Pouvoir de négociation des distributeurs | Le pouvoir de négociation des distributeurs de produits PurPaws | 1 |
| 7. Réglementation | Les lois et réglementations qui encadrent la vente d'objets connectés de distribution de nourriture pour animaux et qui peuvent avoir un impact sur les activités de PurPaws | 4 |



Stratégie Marketing

Valeurs de l'entreprise

▶ Produit

▼ Prix

| | |
|-----------------------|---|
| Tarif | 150-200€, avec la possibilité d'ajuster la remise selon les besoins des clients |
| Condition de paiement | Carte bleue, Paypal, Bitcoin |
| Condition de crédit | Max 3 fois sans frais |

Stratégie tarifaire
Écramage pour positionner notre produit comme haut de gamme, avec une certaine flexibilité pour ajuster notre positionnement au besoin

En résumé, notre produit coutera entre 150 et 200€, qui sera payé par carte bleue, Paypal, Bitcoin. Les conditions de crédit seront déterminées prochainement. Nous adoptons une stratégie tarifaire d'écramage pour positionner notre produit comme haut de gamme, tout en conservant une certaine flexibilité pour ajuster notre positionnement en fonction de l'évolution de la demande et de la concurrence.

▶ Distribution

▶ Communication



Tunnel de vente

- ▶ Prise de conscience
- ▶ Considération
- ▶ Conversion
- ▶ Fidélisation



Les actions marketing digital

▶ Promotions Vidéos

▶ Promotions Réseaux Sociaux

▼ SEO

- Outils utilisés :

- Outils d'analyse de mots-clés
- Outils d'analyse de trafic
- Outils de création de contenu
- Google Analytics

▪ Etapes :

1. Effectuer une recherche de mots-clés pertinents liés à notre produit et notre audience cible.
2. Utiliser ces mots-clés dans le contenu de notre site internet, les balises meta, les URLs, les titres et les descriptions des pages.
3. Créez du contenu de qualité et pertinent pour les utilisateurs en utilisant ces mots-clés.
4. Améliorer la structure et la navigation du site internet pour une meilleure expérience utilisateur.

▪ Raisonnement :

- Le référencement naturel est une stratégie importante pour attirer du trafic qualifié et augmenter la visibilité de notre site Internet sur les moteurs de recherche. Cela nous permettra d'atteindre notre audience cible et de générer des prospects qualifiés pour notre produit.

▪ Objectifs :

- Optimiser le site internet et le contenu pour les moteurs de recherche afin d'augmenter la visibilité et attirer plus de trafic qualifié.

▪ Mesures :

- Suivi du trafic sur le site Web
- Suivi des actions utilisateurs en %
- Classement des mots-clés
- Suivi du nombre et de la qualité des liens entrants vers le site Web contribuant à l'augmentation de la visibilité et l'autorité du site Web

Pour compléter notre présentation, nous avons donc ajouté une étude de marché, la stratégie marketing de notre marque « **Purpaws** » et quelques éléments supplémentaires.

Nous avons tous participé cette fois-ci et avons comblé certaines lacunes, clarifié des incompréhensions concernant l'ordre dans lequel nous avons réalisé les différentes tâches, ajouté des éléments tels que le **MCD** et le **MPD**, introduit un début et une fin, et inclus des photos de l'équipe retouchées par une IA, une **Intelligence Artificielle** (dans notre cas, elle a été utilisée pour légèrement modifier nos photos avec un style unique, donnant ainsi un aspect original à nos photos de présentation).

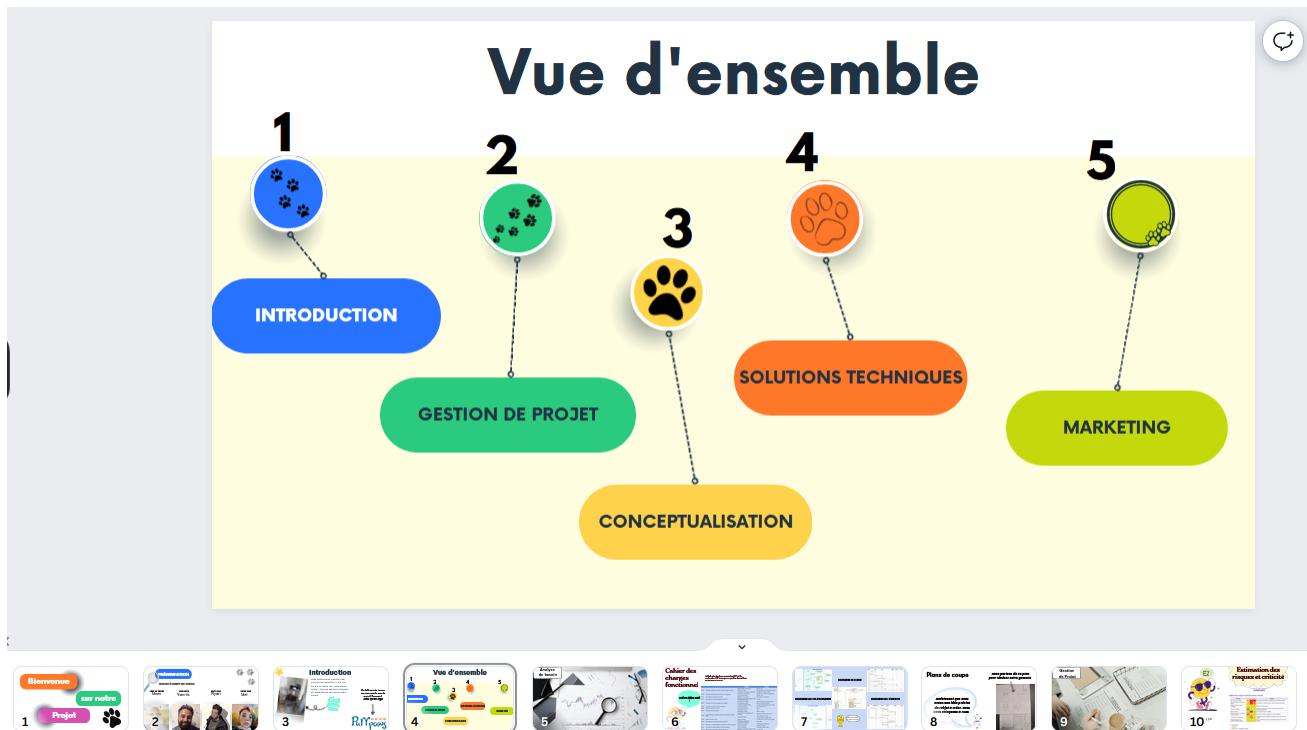
IA

Je tiens à préciser que le terme IA n'est en fait en aucun cas de l'intelligence. L'intelligence est, selon moi et dans le sens « humain » de la chose, la faculté d'adaptation. Certes l'IA s'améliore par la répétition (beaucoup, BEAUCOUP de répétitions) mais n'est en aucun cas de l'intelligence à proprement parlé.

L'IA ne comprend pas, elle doit être orientée vers ce qui est une bonne ou une mauvaise réponse (selon des critères définis par les responsables) aux demandes des utilisateurs.

Et enfin, le **Canva** fut prêt !

Soutenance du projet Purrpaws



Pour ma part, une présentation orale devant des juges est une nouvelle expérience. Il nous a fallu expliquer de manière claire et précise, tout en restant simple, le commencement, le déroulement et l'objectif final du projet de la Gamelle connectée.

Ce qui, en fin de compte, s'est avéré bien plus complexe que ce à quoi je m'attendais. Écrire une expérience et la raconter sont totalement différents.

Un oubli ou une erreur dans un écrit se résout facilement avec une relecture et une réécriture. Perdre le fil ou oublier, même vaguement, un sujet à l'oral est bien plus complexe à rattraper.

Nous nous sommes répartis le plus équitablement possible les différentes slides comme ceci :

- 1 - Démarrage du PowerPoint et explication (Marc)
- 2 - Présentation (Tout le monde)
- 3 - Introduction (Mel)
- 4 - Présentation des catégories (Mel)
- 5 - Slide Analyse du besoin
- 6 - Cahier des charges (Yannis)
- 7 - Diagrammes UML (Yannis)
- 8 - Plan de coupe (Marc)
- 9 - Slide Gestion de Projet
- 10 - Gestion risque et criticité (Marc)
- 11 - Notion (Ryan)

- 12 - Slide conceptualisation
- 13 - MCD (Ryan)
- 14 - Wireframes (Mel)
- 15 - Maquettes (Mel)
- 16 - Slide solution technique
- 17 - Ingénierie (Yannis)
- 18 - Ingénierie (Yannis)
- 19 - Langages (Marc)
- 20 - Schéma Python <-> Symfony <-> React (Ryan)
- 21 - Marketing (Yannis)
- 22 - Marketing ((everyone))
- 23 - Questions (everyone)

Nous avons donc fait notre présentation. J'ai pour ma part un peu oublié certains détails mais dans l'ensemble, tout s'est bien passé.

Notre jury était composé d'un débutant en développement et d'un expert avec un certain bagage technique et de gestion. Les questions posées par eux avaient donc un écart significatif en terme de cible.

L'un nous demandait quels ont été les problèmes rencontrés et comment nous les avons fixés. Comment nous étions arrivés à telle ou telle conclusion ou qu'est ce qui nous a amené à choisir telle ou telle technologie ou langage informatique.

Tandis que l'autre chercher plus à comprendre comment faire pour utiliser l'application et faire ce dont il avait besoin pour nourrir le chat, se connecter, ou modifier ses choix. Pourquoi nous avons choisi la couleur bleue et orange, ou encore s'il avait besoin d'internet pour que la gamelle fonctionne.

Nous lui avons donc expliqué ce qu'est l'**UX**.

UX

Ce que l'on appelle **l'Expérience Utilisateur (User Experience)** consiste à rendre le plus agréable possible la sensation qu'aura l'utilisateur en consommant notre application. Au plus l'utilisateur prendra de plaisir à rester sur notre application ou notre site, le plus de chance nous aurons de pouvoir vendre notre produit.

Après plusieurs minutes, nous avons étanché leur soif de questions et fini notre présentation sur une note détendue et souriante.

Notre objectif étant atteint, nous avons pu reprendre la suite de notre périple.

Au cours des jours qui suivirent l'annonce de notre présentation et après concertation avec l'équipe, il fallait impérativement que je finisse un travail avant de reprendre l'API : le **bouton des notifications**.

NotificationButton.js

```
8   // Composant NotificationButton
9  const NotificationButton = () => {
10    const [isDrawerOpen, setIsDrawerOpen] = useState(false);
11    const [isClosing, setIsClosing] = useState(false);
12    const [notifications, setNotifications] = useState([]);
13
14    // Gestion du clic sur le bouton de notification
15    const handleNotificationClick = (event) => {
16      if (!isClosing) {
17        event.stopPropagation();
18        setIsClosing(true);
19
20        // Gère le click extérieur du bouton notification en mettant un délai pour l'ordre d'exécution
21        setTimeout(() => {
22          setIsDrawerOpen(!isDrawerOpen);
23          setIsClosing(false);
24        }, 0);
25      }
26    };
27
28    const drawerRef = useRef();
29
30    useEffect(() => {
31      // Gestion du clic en dehors du drawer pour le fermer
32      const handleClickOutside = (event) => {
33        if (drawerRef.current && !drawerRef.current.contains(event.target)) {
34          setIsClosing(true);
35
36          // Gère le click extérieur du bouton notification en mettant un délai pour l'ordre d'exécution
37          setTimeout(() => {
38            setIsDrawerOpen(false);
39            setIsClosing(false);
40          }, 0);
41        }
42      };
43
44      // Ajout des écouteurs d'événements lors de l'ouverture du drawer
45      if (isDrawerOpen) {
46        document.addEventListener("mousedown", handleClickOutside);
47      } else {
48        document.removeEventListener("mousedown", handleClickOutside);
49      }
50
51      // Nettoyage des écouteurs d'événements lors du démontage du composant
52      return () => {
53        document.removeEventListener("mousedown", handleClickOutside);
54      };
55    }, [isDrawerOpen]);
```

Nous nous sommes rendus compte qu'il fallait que l'onglet des notifications se ferme si on clique à côté dudit onglet, (**handleClickOutside**), et en mettant un léger délai à l'ouverture/fermeture, cela rend le tout plus interactif, l'**UX** est beaucoup plus agréable donc, idée conservée (**UX** signifie Expérience utilisateur).

Puis j'ai attaqué les pages de connexions, le burger bouton, le CSS qui gère les composants React.

Mais qui dit « formulaire » dit « champ à remplir par l'utilisateur » et donc, données à sauvegarder (notamment le mail et le mot de passe).

Il nous a donc fallu simuler un échange « requête/réponse » entre le front, l'api et le back, c'est là qu'arrive Postman et Mockoon.

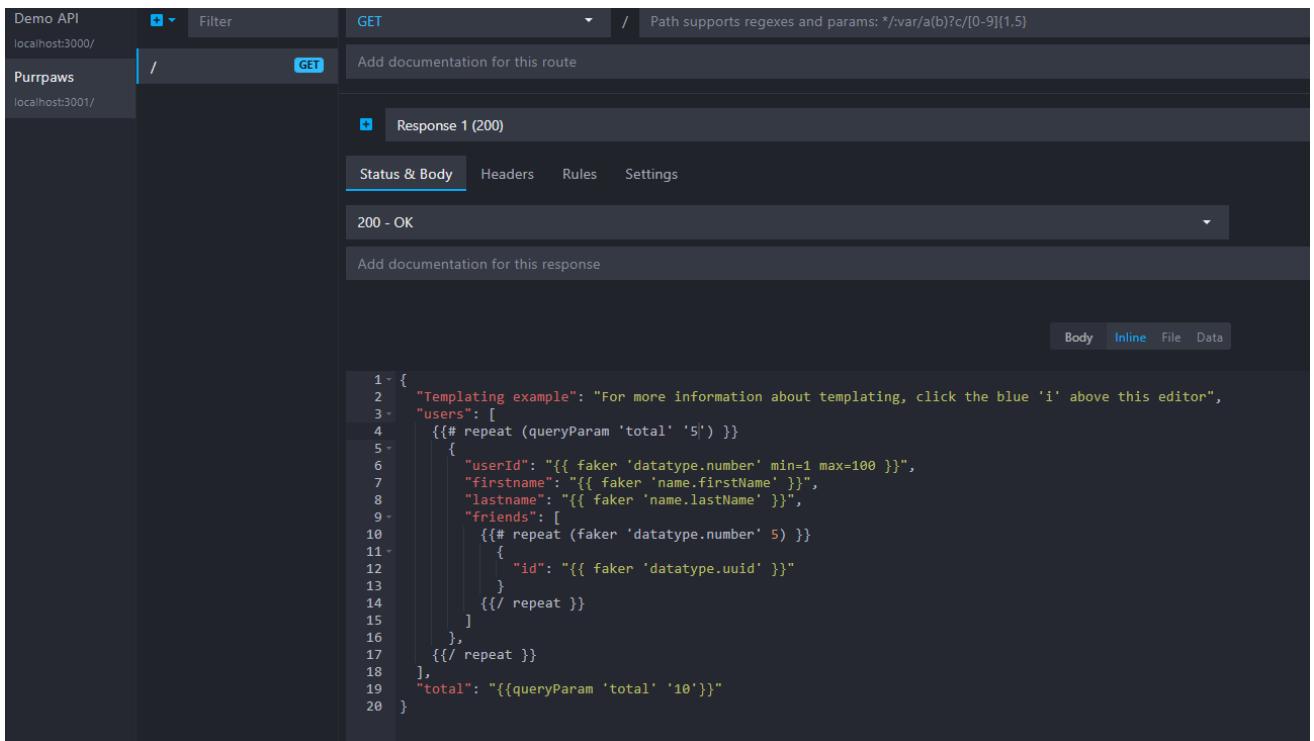
Mockoon

Mockoon est un outil qui vous permet de créer des serveurs **simulés** (mocks) pour simuler le comportement d'une API. Il est principalement utilisé pour tester des clients d'API lorsque le serveur réel n'est pas encore disponible. On va l'utiliser pour optimiser la création de notre API et, au passage, apprendre cet outil.

Principaux avantages de Mockoon :

Création de mocks d'API : Mockoon vous permet de créer des points de terminaison (endpoints) simulés avec des réponses préconfigurées. Si vous voulez paramétrier votre API de différentes manières, vous pouvez le tester grâce à Mockoon.

Simulation de scénarios : Vous pouvez simuler différents scénarios en configurant des routes et des réponses différentes pour chaque route. Cela vous permet de tester comment votre application va réagir aux diverses réponses d'API.



The screenshot shows the Mockoon interface for a 'Purrpaws' API. A 'GET' request is defined for the '/users' endpoint. The response body is a JSON template:

```
1 - {
2   "Templating example": "For more information about templating, click the blue 'i' above this editor",
3   "users": [
4     {"# repeat (QueryParam 'total' '5') "}
5     {
6       "userId": "{{ faker 'datatype.number' min=1 max=100 }}",
7       "firstname": "{{ faker 'name.firstName' }}",
8       "lastname": "{{ faker 'name.lastName' }}",
9       "friends": [
10         {"# repeat (faker 'datatype.number' 5) "}
11         {
12           "id": "{{ faker 'datatype.uuid' }}"
13         }
14         {/ repeat }
15       ],
16     },
17   ],
18   "total": "{{QueryParam 'total' '10'}}"
19 }
20 }
```

Ici, nous envoyons les données de base de création d'un nouveau compte utilisateur.

J'ai donc commencé par :

- Un « userId » qui sera unique pour chaque utilisateur
- Un Nom
- Un Prénom
- Le champ « friends » sera le champ de l'animal (ou des animaux) et un identifiant à l'intérieur qui sera lui aussi unique.

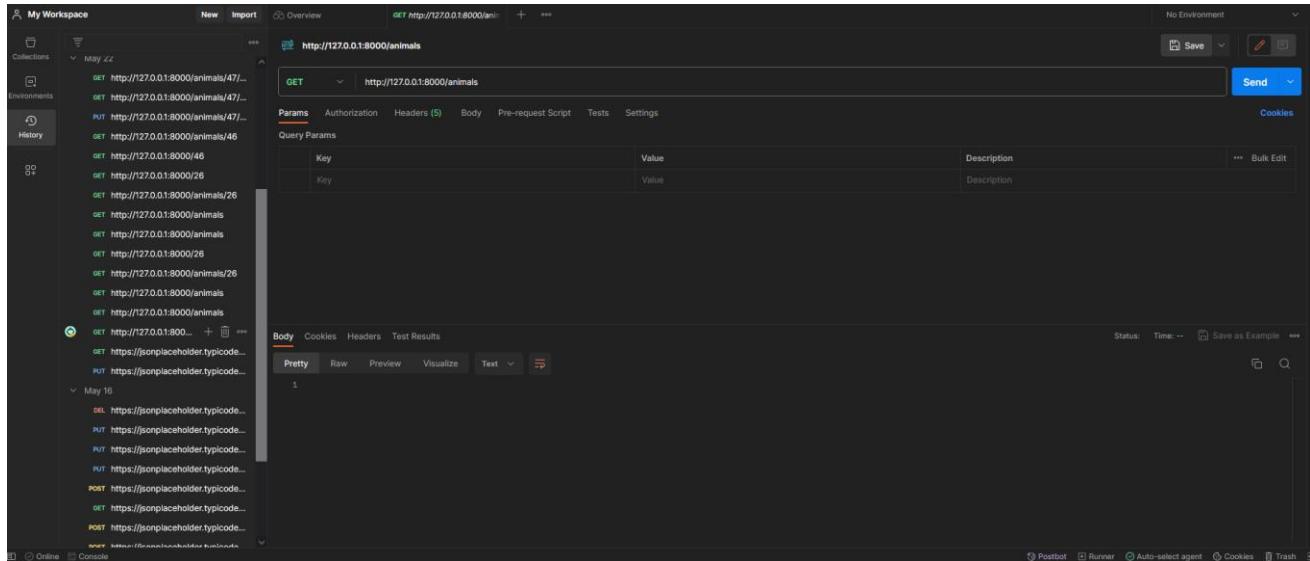
Pour information, les « faker » qui définissent les valeurs dans les champs génèrent aléatoirement des noms, des dates, des chiffres, cet outil est très pratique pour remplir des champs vides.

Ces « faker » sont donc les premières données que je voulais envoyer pour voir ce que ça donnerait après réception dans Postman.

Postman

Postman est essentiellement la même chose que Mockoon, mais de l'autre côté de la barrière. Comme écrit plus haut, toute réponse demande requête. Mockoon simule l'envoie de données API, Postman simule la réception de ces données et le traitement qui devra s'adapter à ces requêtes.

C'est également une application de développement API qui permet aux développeurs de tester, documenter et automatiser les interactions avec des API, facilitant ainsi le développement et le débogage des services web.



The screenshot shows the Postman interface with the following details:

- Left Sidebar:** Shows 'My Workspace' with a collection named 'May 22' containing various API requests (GET, PUT, POST) to 'http://127.0.0.1:8000/animals' and 'https://jsonplaceholder.typicode.com'.
- Overview Tab:** Shows a summary of the current request: GET http://127.0.0.1:8000/animals.
- Request Details:**
 - Method:** GET
 - URL:** http://127.0.0.1:8000/animals
 - Params:** An empty table.
 - Body:** A JSON object with one item: { "1": null }.
 - Headers:** Authorization, Headers, Body, Pre-request Script, Tests, Settings.
- Bottom Status Bar:** Shows 'Postbot' and other status indicators.

L'adresse écoutée est donc ici <https://127.0.0.1:8000/animals>. Mockoon devra donc utiliser cette adresse pour envoyer la donnée, et Postman devra recevoir le JSON de l'API et afficher les données grâce au GET.

SEO

Avec les cours que nous avons suivis, il fallait nous occuper de notre **référencement** (C'est la visibilité sur internet et dans les recherches Google par exemple), nous avons donc chercher des noms percutants et ciblés pour nous donner de la visibilité. Mais d'abord, petite explication de ce qu'est le **SEO** :

Le **SEO**, ou **Search Engine Optimization** en anglais, est l'ensemble des pratiques visant à améliorer la visibilité et le classement d'un site web dans les résultats des moteurs de recherche. Nous avons choisi de répertorier un ensemble de mots clés que pourrait rechercher un client potentiel. Le but étant d'optimiser le contenu et la structure du site pour correspondre aux critères et aux algorithmes de recherche des moteurs. On aurait pu choisir des mots (pardonnez-moi l'expression) « putaclick », mais nous sommes restés sur des mots pertinents.

Je me permets une **parenthèse** ici car, les algorithmes de recherche sont les moyens les plus efficace d'avoir de la visibilité sur internet, le problème me demanderez-vous ? Et bien, si un jour, par exemple, un créateur créé le meilleur site internet ou la meilleure vidéo YouTube, mais que sont SEO n'est pas bon, alors ils ne seront jamais vu. Tout cela parce que les mots clés ou le titre de ceux-ci ne seront pas pertinents ou « putaclick ». Je trouve vraiment dommage que la qualité du contenu ne l'emporte pas sur le titre ou les mots clés associés à ce site ou vidéo mais bon ... (Fin de la **parenthèse**)

Pour notre site internet, nous avons, sur Notion, mis en place notre stratégie SEO, voici les suggestions de l'équipe :

Mettre en place une stratégie SEO

Mots clés et champ lexical liés à notre activité après recherche de volume mensuels sur GoogleTrends et GoogleAdwords :

- *Gamelle pour chat*
- *Gamelle connectée*
- *Gamelle intelligente*
- *Gamelle automatique*
- *Gamelle programmable*
- *Gamelle autonome*
- *Acheter gamelle pour chat connectée*
- *Où trouver gamelle pour chat connectée*
- *Prix gamelle pour chat connectée*
- *Comparatif gamelle pour chat connectée*
- *Gamelle pour chat connectée pas chère*
- *Offre gamelle pour chat connectée*
- *Gamelle pour chat connectée en promotion*
- *Gamelle pour chat connectée en soldes*
- *Boutique gamelle pour chat connectée*
- *Site pour acheter une gamelle pour chat connectée*
- *Alimentation intelligente pour chat*
- *Gamelle pour chien*
- *Gamelle avec réservoir*

Et avec ces suggestions, voici le début de notre site :

```

<title>Gamelle Connectée pour Animaux | PurrPaws</title>
<meta name="description" content="Découvrez notre gamelle connectée pour animaux. Idéale pour votre chat, ce distributeur de croquettes automatique simplifie l'alimentation de votre boule de poil.">
</head>
<body>
    <h1>Gamelle Connectée pour Animaux</h1>
    <p>Notre gamelle connectée pour animaux, également idéale pour votre chat, est bien plus qu'un simple distributeur croquette automatique<br>
    <a href="/produitVente" title="Découvrir notre gamelle connectée pour animaux">Gamelle Connectée pour Animaux</a>

```

Il y'a plusieurs manières d'optimiser le référencement, mais nous avons choisi celui-ci. Il existe aussi le choix des balises « **meta** » mais apparemment, elles sont moins pertinentes aujourd'hui, les moteurs de recherche se concentrent davantage sur le contenu et les balises donc, dans l'image « **gamelleV1** » et dans le « **alt** » nous irons.

Accessibilité web et son importance pour notre site

L'accessibilité web se réfère à la pratique de concevoir et de développer des sites web de manière à ce qu'ils soient utilisables et accessibles par tous, y compris les personnes ayant des handicaps ou des limitations physiques, sensorielles ou cognitives. (**C26**)

Cela permet à un plus grand nombre de personnes de naviguer sur notre site web et d'y accéder facilement. Cela peut inclure des personnes atteintes de déficiences visuelles, auditives ou motrices, ainsi que des personnes ayant des troubles tels que la dyslexie. Le tout étant règlementé par le **RGAA**.

Le **RGAA** ou **Référentiel général d'amélioration de l'accessibilité** énonce des directives techniques et des recommandations pour garantir que les contenus numériques sont utilisables par un large public, y compris les personnes handicapées de différentes manières. Le **RGAA** a été élaboré en réponse à la loi pour l'égalité des droits et des chances, la participation et la citoyenneté des personnes handicapées de 2005.

Je vous montre une partie de notre **Notion** où nous avons déterminer comment nous allions gérer l'accessibilité à notre site :

Méthodes pour rendre notre site web accessible pour les personnes ayant des handicaps :

1. Visuels

- Utiliser des couleurs de texte et de fond suffisamment contrastées pour que le texte soit facile à lire.
- Utiliser des tailles de police suffisamment grandes pour que le texte soit facile à lire, même pour les personnes ayant une déficience visuelle.
- Utiliser des descriptions alternatives (alt-text) pour toutes les images afin que les personnes malvoyantes puissent comprendre leur contenu.

2. Auditifs

- Utiliser des sous-titres pour les vidéos et les enregistrements audio.
- Fournir des transcriptions textuelles pour les contenus audio et vidéo.
- Éviter d'utiliser des contenus audio et vidéo qui dépendent uniquement du son pour transmettre l'information.

3. Moteurs

- *S'assurer que le site web peut être navigué à l'aide d'un clavier uniquement, sans nécessiter l'utilisation d'une souris ou d'un autre périphérique de pointage.*
- *Utiliser des boutons et des liens suffisamment grands pour être facilement cliquables avec une souris ou un autre périphérique de pointage.*
- *Éviter d'utiliser des contenus qui clignotent rapidement, car cela peut être dérangeant pour les personnes atteintes d'épilepsie ou d'autres troubles neurologiques.*

4. Cognitifs

- *Utiliser un langage simple et clair pour que les informations soient faciles à comprendre.*
- *Utiliser une structure de site web claire et facile à naviguer pour aider les utilisateurs à trouver rapidement l'information dont ils ont besoin.*
- *Éviter d'utiliser des termes techniques ou des acronymes qui peuvent être difficiles à comprendre pour les personnes ayant des handicaps cognitifs.*

Je tiens à préciser qu'ayant au sein même de notre équipe une personne en situation d'handicap, cette section, qui je l'avoue, n'était pas du tout ma priorité quand nous avons organisé notre programme, est devenue une petite révélation pour moi. Malgré quelques problèmes physiques, je me porte bien et n'est pas de problèmes particuliers, ce qui n'est pas le cas de tout le monde, y compris des potentiels acheteurs.

Nous avons donc tout naturellement revue notre manière d'ajouter des options dans notre code front end pour que tous/tes puissent à minima s'orienter correctement dans notre application.

Et c'est à ce moment qu'arrive : **mes débuts de stage en entreprise !**

L'entreprise



ARC Europe France, filiale du groupe ARC EUROPE est située à Limonest, à côté de Lyon (69). C'est un centre d'appels spécialisé dans le déploiement de prestations d'assistance automobile pour des clients du monde de l'automobile, du leasing et de l'assurance.

Ils interviennent pour des bénéficiaires en besoin d'assistance en bord de route en France et en Europe. Afin d'assurer un service de qualité et de proximité, ARC Europe s'est doté d'un réseau performant de prestataires (dépanneurs, loueurs de véhicules, taxis...) permettant une plus grande réactivité.

L'entreprise compte 450 salariés permanents environ.

Cette expérience est une véritable aventure, où chaque élément appris renferme en lui-même des sujets qui pourraient être discutés bien plus longuement que le temps alloué à la présentation orale.

Ainsi, le mot d'ordre sera la concision ! Mais sans précision, ça ne sera pas forcément facile à cerner donc, je vous explique le fonctionnement de la société Arc.

Imaginez que vous êtes en train de conduire votre voiture pour aller au travail, mais, pour une raison inconnue, votre voiture s'arrête. Vous êtes en panne au bord de la route ! Après avoir retrouvé votre calme (et cessé de proférer des jurons à la chance ou plutôt à l'infortune qui est la vôtre), vous évaluez la situation et déterminez ce que vous devez faire en premier. La première étape à suivre est de contacter votre assurance ! Vous composez le numéro et dites : « Allo, j'ai besoin d'aide, je suis en panne. » Vous êtes donc le **bénéficiaire**.

De l'autre côté de la ligne, la personne qui vous répond est située dans notre bureau à Arc, et elle s'appelle un **chargé** d'assistance. Il/elle s'assure que vous allez bien, que vous êtes en sécurité, puis recueille des informations vous concernant, comme votre nom ou votre marque de voiture. Le chargé vous demande l'autorisation de vous **géolocaliser** afin de dépêcher une dépanneuse exactement à votre emplacement. Si vous avez des questions, il ou elle est là pour vous fournir toutes les réponses nécessaires. La dépanneuse arrive, prend en charge votre voiture, et l'achemine vers un garage capable de réaliser les réparations nécessaires. Ces garages sont désignés comme **réparateurs agréés (RA)**. Le RA effectue une évaluation de votre véhicule, effectue les réparations requises, puis restitue votre voiture réparée en moins d'une semaine.

Enfin, la facture est émise par le service comptable, puis envoyée au bénéficiaire.

Ma reconversion professionnelle devait avoir du sens pour moi, et aider des personnes qui en ont besoin m'a tout de suite intéressé. Je me suis donc lancé dans l'aventure Arc Europe France.

Déroulé du stage

Pendant les deux premières semaines de mon expérience, nous avons consacré du temps à la mise en place de mon environnement de travail et j'ai reçu une présentation approfondie du fonctionnement de l'entreprise. Cette période a également inclus des visites des locaux, des observations des appels avec les bénéficiaires pour comprendre le processus de traitement des appels, ainsi qu'une exploration détaillée des détails de mon futur espace de travail, tant sur le plan physique qu'environnemental.

Ensuite, nous avons plongé dans la résolution de problèmes, suivi d'un examen complet de tous les éléments interactifs de l'application **Cactus**. J'ai relevé plusieurs problèmes à ce stade :

- Certains formulaires étaient en réalité présentés sous forme de tableaux (utilisant les balises tr/td), dans lesquels du texte était inséré directement, sans recours à des variables ou des composants réutilisables.
- Certains éléments censés être des boutons étaient en fait des images (gif/jpg) cliquables agissant comme des liens de redirection.
- Une portion raisonnable du code datait de plus de 20 ans, ce que l'on appelle souvent du code "**legacy**".

Cette phase m'a permis de saisir une réalité manifeste mais parfois négligée : alors que nos formations nous enseignent à construire des solutions à partir de zéro, le monde professionnel peut nécessiter des compromis dans le code, que ce soit pour des raisons de durabilité environnementale (évoquée sous le terme "**Green IT**") ou pour faciliter les révisions du code par des pairs ("**code review**").

En collaboration avec Damien (collègue développeur), nous avons entrepris la transformation des 72 images agissant comme des boutons (une pratique discutable aujourd'hui mais rependue à l'époque où a été créer le code) en véritables boutons textuels (avec une balise HTML). Cette transformation permettait une meilleure gestion des boutons, la suppression des images de la base de données et la création d'un style CSS approprié. (**Green It**)

Ensuite, nous avons exécuté une révision du code (**code review**), employant des pratiques similaires à celles utilisées avec Git, suivie de la création d'une "**release note**" détaillant nos modifications pour la version 2.26.

Nous avons également rédigé une **documentation** dans laquelle nous avons indiqué les changements effectués, en y associant une date. (**C24**)

Nous avons ensuite identifié des problèmes mineurs, que nous avons résolus rapidement, avant de les soumettre à un processus de validation (**tests**) en environnement de recette.

Cette séquence de développement suivait le parcours : développement (**dev**) => recette(**rec**) => production (**prod**). Ce qui veut dire que nous développons en DEV, nous testons le code en REC, et poussons le résultat validé par tous en PROD.

De plus, nous avons **automatisé** des requêtes SQL, comme l'exemple suivant :

```
INSERT INTO ATP_Event VALUES (353233,'2015-01-5','14:28:43',NULL,'Renault',2,NULL,2,NULL,NULL,NULL);
```

Après mes vacances, Cédric m'a expliqué les "**nugets**", qui sont des bibliothèques de dépendances utilisées lors de la compilation. Suite à cela, notre mission a consisté à chiffrer le niveau de difficulté pour résoudre différents problèmes, en d'autres termes, à estimer la charge de travail associée à chaque ticket.

Cartographie Applicative Exacte

Je ne vais pas vous mentir, il m'a fallu quelques semaines pour comprendre où je mettais les pieds, voyez par vous-même :

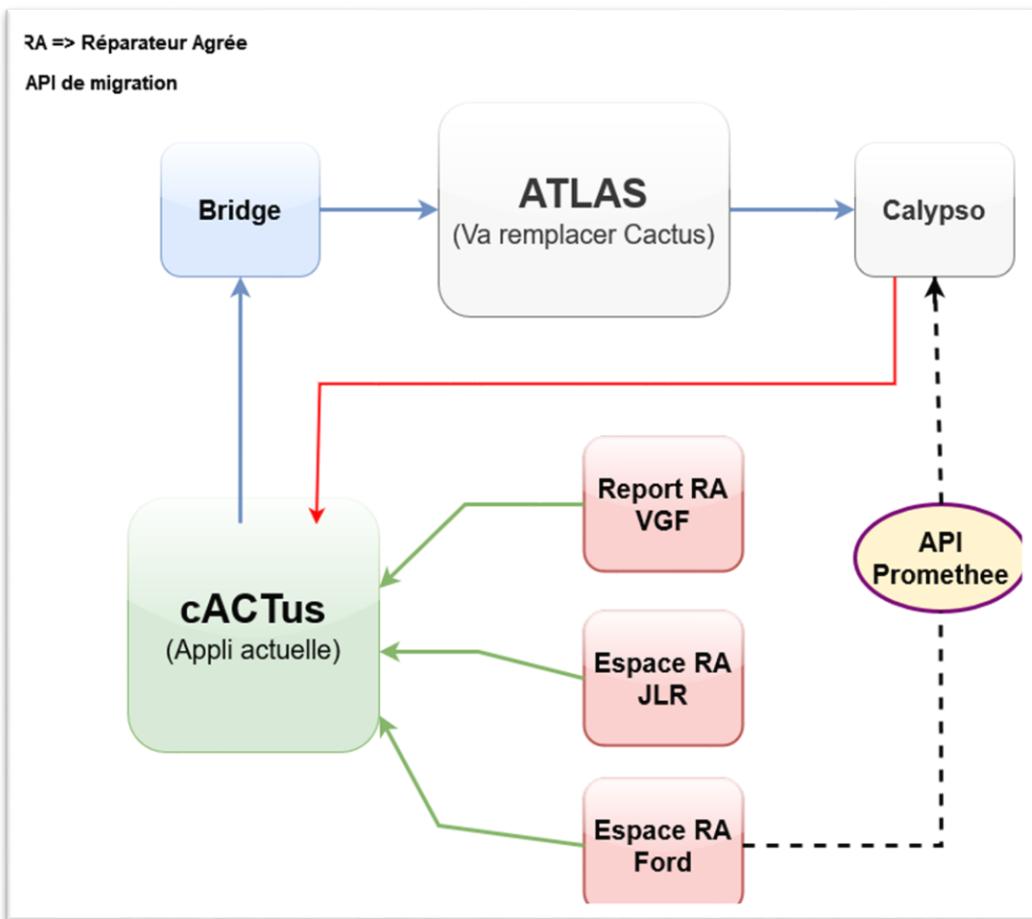


Sachez que je ne touche pas à la totalité de ces applications, mais juste à quelques une d'entre elles (heureusement pour moi).

En tant que stagiaire, je me suis concentré sur les applications principales : **cACTus**, **Bridge**, **Resimage**, **Becall**, **Espace RA**. J'ai eu l'occasion de suivre les mises en recette du projet "**Bridge/cACTus**". Nous avons revu la procédure, car celle-ci était incomplète, puis nous avons soumis le projet "**Bridge**" à la phase de recette.

Pour ce travail, nous avons consacré un total de 6 heures avec Damien, et une assistance de 4 heures de la part d'Aurélien.

Cartographie Applicative localisée



Pour résumer l'image ci-dessus, **cACTus** est l'application principale de la société. **Atlas** est la version 2022 Polonaise (AEF, AEP pour Arc Europe France / Pologne). **Bridge** fait le pont entre **Atlas** et **cACTus** pour le transfert de données, **Calypso** est la DataBase, le but de l'API est d'envoyer toutes les données de Ford (un de nos clients) vers la DataBase Polonaise. **VGF** (Volkswagen) et **JLR** (Jaguar, Land Rover) suivront une fois que tout sera stable.

On m'a confié le développement de l'API destinée à faciliter la migration entre les deux applications, qui sera précisément entre l'**Espace RA Ford** et **Calypso**. J'ai réussi à mettre en place une version fonctionnelle de cette API, avec des tests réalisés à l'aide de **Postman**, **Swagger**, **TFS** et **MVS**. Son nom ? **Promethee**.

Cependant, cette version n'a pas satisfait les exigences du client et mon responsable a décidé de la refondre entièrement.

Pour cette nouvelle version de l'API, nous avons repris le processus depuis le début, en utilisant MVS comme base. Nous avons créé le projet via TFS et l'avons ensuite cloné dans MVS.

Nous avons ensuite ajouté les différentes solutions nécessaires, ainsi qu'un répertoire dédié à la base de données (BDD) et un autre pour les librarys (adresse de jonction avec la BDD).

Les dépendances nécessaires ont également été intégrées à l'aide de nugets. Un **NuGet** dans Microsoft Visual Studio est un module de code préconstruit qui ajoute des fonctionnalités ou des bibliothèques externes à un projet, simplifiant ainsi le développement en réutilisant du code existant.

Exemple :

Swagger : Dans le domaine du développement web, est un outil essentiel qui permet de concevoir, documenter et tester des APIs de manière efficiente. Il propose une interface interactive pour interagir avec les points d'accès (endpoints) et génère automatiquement une documentation claire et compréhensible pour les développeurs.

Newtonsoft.Json : Souvent simplement désignée sous le nom de Json.NET, c'est une bibliothèque largement reconnue pour le traitement des données au format JSON (JavaScript Object Notation) dans le domaine du développement logiciel. Cette bibliothèque facilite la **sérialisation** (conversion d'objets en JSON) ainsi que la **désérialisation** (conversion de JSON en objets) grâce à une approche souple et pratique.

Une fois cette étape accomplie, nous avons construit les bases de notre future API. Dans le but de garantir une compatibilité optimale pour la migration (serveur de 2013, code de 2023), nous avons intégré **LLBLGen Pro**.

LLBLGen Pro se révèle être un outil puissant et flexible de mappage objet-relationnel (ORM) qui simplifie l'accès aux bases de données. En permettant aux développeurs de manipuler les données au moyen d'objets et de requêtes orientées objet plutôt que d'interactions directes avec les bases de données relationnelles, cet outil améliore grandement l'efficacité et la maintenabilité du code. Nous avons choisi la version 5.10.1, car elle est la seule à pouvoir prendre en charge **.NET7** en cas de migration vers un serveur plus récent, envisageant ainsi la pérennité de notre API. (Cette version a été obtenue via le chemin <\\srv-it\IT Sources\Tools\BDD\LLBLGen\Versions>).

Dans le cadre de la migration de **cACTus** vers **Atlas**, il a fallu faire une fonction qui va copier les données de l'atelier RA (garage) vers Atlas, et une autre qui va vérifier si l'atelier RA a déjà été migré ou non.

Arrive donc les premiers tests unitaires avec **Telerik JustMock** (c'est un outil de mocking pour les tests unitaires en développement logiciel, permettant de simuler le comportement des dépendances). Le but des tests unitaires est simple, on vérifie si la fonction fait ce qu'elle est censée faire. Telerik JustMock est un outil qui facilite la création de ces tests unitaires en permettant de "simuler" ou de "se moquer" de certaines parties du code. Pour ça, on va lui envoyer des données correctes, et de fausses données pour voir comment va réagir la fonction. Si le test est concluant, alors la fonction fonctionne bien, indépendamment du reste du code et peut donc partir en recette /prod. Sinon, cela veut dire que la fonction est incomplète/fausse et à revoir. (**C19**)

J'ai appris l'approche "**Arrange, Act, Assert**" (**AAA**). C'est une structure couramment utilisée dans la rédaction de tests unitaires pour maintenir une organisation claire et cohérente. J'explique :

1. **Arrange** : C'est la première étape où on configure l'environnement de test. Il faut mettre en place tous les prérequis et les conditions initiales nécessaires pour le test. Cela peut inclure la création d'objets, l'initialisation de variables, la configuration de l'état initial, ou la préparation de données de test.

2. **Act** : C'est la deuxième étape où on effectue l'action ou l'appel de la méthode que l'on souhaite tester. C'est le moment où on exécute le code à évaluer.
3. **Assert** : C'est la dernière étape où il faut vérifier les résultats de l'action entrepris à l'étape précédente. On compare les résultats obtenus avec les résultats attendus pour s'assurer que le comportement du code testé est correct. Si les résultats correspondent aux attentes, le test est considéré comme réussi.

Sur l'image qui suit, nous avons testé la fonction **MigrateWorkshopGroupInBDD** qui va chercher si le regroupement de garage (ce sont des groupes de garages, je prends l'exemple de Ford, s'il y'a 5 garages sur Lyon, le groupe contiendra 5 garages Ford) à bien envoyé les données à Atlas. (**3 réussites ici**).

Dans les deux premiers tests, on envoie de bonnes réponses, dans le troisième, des fausses.

```

    public void MigrateWorkshopInBDD_migrationDateNotExist()
    {
        // Arrange
        DealerInfoEntity mockWorkshopEntity = Mock.Create<DealerInfoEntity>();
        mockWorkshopEntity.MigrationDate = null;
        mockWorkshopEntity.IsActive = true;

        FordMigrationManager mockManager = Mock.Create<FordMigrationManager>();

        // Act
        var inst = new PrivateAccessor(mockManager);
        bool result = (bool)inst.CallMethod("MigrateWorkshopInBDD", new object[] { mockWorkshopEntity });

        // Assert
        Assert.IsTrue(result, "La valeur de retour doit être vrai.");
        Assert.IsTrue(mockWorkshopEntity.MigrationDate.HasValue, "MigrationDate doit avoir une valeur.");
        Assert.IsFalse(mockWorkshopEntity.IsActive, "IsActive doit être faux.");
    }
}

```

Compléments

Dans un souci de vous montrer au maximum le fait que j'ai vu l'ensemble du référentiel DFS, je vais vous expliquer ce que j'ai fait sur différentes cellules du référentiel de l'école (**C XX**) sans qu'il y est forcément de suite logique, mais juste confirmer le fait que j'ai vu ou fait ces activités.

Back API node express

The screenshot shows a code editor interface with a sidebar on the left displaying the project structure and a main panel on the right showing the code content. The sidebar shows the following structure:

- main
- + Go to file
- api
- controllers
- routes.js (selected)
- core
- models
- node_modules
- .DS_Store
- .env
- app.js
- package-lock.json
- package.json
- web

The main panel shows the content of the routes.js file:

```
1 const express = require("express");
2 const app = express();
3 const router = express.Router();
4 const Users = require("../models/users");
5 const axios = require('axios');

6 // On ecoute avec la requête GET, recuperer tout les users
7 router.route("/users").get((req, res) => {
8     res.status(200).json(Users);
9 });

10 //On ajoute avec le post un nouvel user
11 router.route("/users").post((req, res) => {
12     const newUser = {
13         id: Users.length + 1,
14         name: req.body.name,
15         age: req.body.age,
16         salary: req.body.salary
17     };
18     Users.push(newUser);
19     res.redirect("/users");
20 });

21 //On cible un user en particulier
22 router.route("/user/:id").get((req, res) => {
23     const id = parseInt(req.params.id);
24     const user = Users.find(user => user.id === id);
25     res.status(200).json(user);
26 });

27 // On delete un user en particulier
28 router.route("/users/:id").delete((req, res) => {
29     const id = parseInt(req.params.id);
30     let userIndex = Users.findIndex(user => user.id === id);
31     if (userIndex === -1) {
32         return res.status(404).json({ message: "User not found" });
33     }
34     let deletedUser = Users.splice(userIndex, 1);
35     res.status(200).json({ message: "User deleted", deletedUser });
36 });

37 
```

Dans le cadre de notre apprentissage, nous avons vu les API et le backend qui les gère, il nous a donc fallu faire, grâce à **node.js** et **express**, faire les redirections, l'enregistrement des données et la logique qui gérera les données. Petite explication de ce qu'est **Node.js** et **Express** : (**C16**)

Node.js : C'est un environnement d'exécution JavaScript côté serveur (Back) qui permet d'exécuter du code JavaScript en dehors d'un navigateur web (Chrome, Firefox ...). Node.js est largement utilisé pour créer des applications serveur rapides et évolutives.

Express : C'est un **Framework** web pour Node.js. Il simplifie la création d'applications web en fournissant des fonctionnalités prêtées à l'emploi, comme la gestion des routes et des requêtes HTTP. Express est populaire pour le développement du backend d'API en raison de sa simplicité et de sa flexibilité.

Framework : Il faut voir les Framework comme des infrastructures prêtées à l'emploi pour le développement logiciel. Ils sont conçus pour simplifier et accélérer le processus de création d'applications en fournissant une structure et des outils standardisés. J'ai moi-même beaucoup utilisé **Bootstrap**, qui est très pratique pour le **responsive** design (l'adaptation de l'apparence en fonction de la taille de l'écran : le rendu visuel d'une application ne sera pas obligatoirement le même si on le regarde sur un téléphone portable ou sur un écran d'ordinateur).

Pour expliquer rapidement ce que fait ce code, en premier j'importe les modules nécessaires, ensuite je crée un objet Express en utilisant **express()**, et un routeur est créé en utilisant **express.Router()** pour gérer les différentes routes de l'API. Enfin, le code définit des routes pour différentes actions liées aux utilisateurs. J'ai suivi les conventions REST :

- Pour la route /users en méthode GET, il renvoie tous les utilisateurs stockés sous forme de JSON.
- Pour la route /users en méthode POST, il ajoute un nouvel utilisateur en extrayant les données du corps de la requête.
- Pour la route /user/:id en méthode GET, il récupère un utilisateur spécifique en fonction de l'ID fourni dans les paramètres de la requête.
- Pour la route /users/:id en méthode DELETE, il supprime un utilisateur spécifique en fonction de l'ID fourni dans les paramètres de la requête. Si l'utilisateur n'est pas trouvé, il renvoie un statut 404 avec un message "User not found".

On renvoie sous format JSON des réponses **res.status()** qui définissent le statut http approprié. Les codes HTTP 200 indiquent le succès, les codes 300 sont liés aux redirections, les codes 400 sont des erreurs attribuées au client (Front), et les codes 500 sont des erreurs attribuées au serveur (Back).

Vous devez certainement connaître le plus connu d'entre eux :



Comme vous pouvez le voir sur l'image, il y'a également un fichier «.env » dans notre API, prenons quelques instants pour expliquer son utilité.

.env

Un fichier ` `.env` (environnement) est un fichier de configuration utilisé pour stocker des variables d'environnement dans une application, y compris dans le contexte d'une API .

Ces variables d'environnement sont essentiellement des paramètres ou des valeurs de configuration que l'application peut utiliser pour fonctionner correctement. Ici, nous l'utilisons par exemple pour définir les endPoints (adresses) qui seront utilisées/écoutes par l'API. (**C23**)

```
1  REACT_APP_URL_API=http://localhost:3001
2  # Pour l'appel, ecrire le code suivant :
3  # process.env.REACT_APP_URL_API +
```

Le .env de notre API permet de remplir quelques points importants : (sécurité référentiel)

- **Stockage des configurations sensibles** : Les fichiers ` `.env` sont généralement utilisés pour stocker des informations sensibles ou des configurations spécifiques à l'environnement, telles que des clés d'API, des identifiants de base de données, des jetons d'accès ou des URL de services tiers. Plutôt que d'inclure ces informations directement dans le code source de notre application, elles sont stockées de manière sécurisée dans un fichier ` `.env` .
- **Séparation des configurations et du code** : L'utilisation d'un fichier ` `.env` permet de séparer les configurations de votre code source. Donc on peut changer les paramètres de configuration sans avoir à modifier directement le code de l'API, ce qui la rend bien plus facile à gérer ou à moduler au fil du temps et des différentes mises à jour possibles.
- **Protection des données sensibles** : En utilisant un fichier ` `.env` , on réduit le risque d'exposer des informations sensibles, car ce fichier est généralement ajouté à la liste des fichiers ignorés (exemple : « ` .gitignore` » en bas de la page). En gros le fichier ` `.env` ne sera pas partagé avec le code source lorsque vous utilisez des systèmes de contrôle de version comme Git.
- **Lecture des variables d'environnement** : Dans notre code source, on peut lire les variables d'environnement à partir du fichier ` `.env` ` en utilisant une bibliothèque ou un package qui permet de charger ces variables. Elles peuvent ensuite être utilisées pour configurer l'API, y compris pour se connecter à des bases de données ou gérer l'authentification.

.gitignore

Le fichier ` .gitignore` est utilisé dans les systèmes de contrôle de version Git pour spécifier des fichiers et des répertoires qui ne doivent pas être suivis ni inclus dans le dépôt Git. Cela permet d'ignorer les fichiers temporaires, les fichiers de configuration locaux et d'autres fichiers non pertinents pour le développement.

```
1  # See https://help.github.com/articles/ignoring-files/ for more about ignoring files.  
2  
3  # dependencies  
4  /node_modules  
5  /.pnpm  
6  .pnpm.js  
7  
8  # testing  
9  /coverage  
10  
11 # production  
12 /build  
13  
14 # misc  
15 .DS_Store  
16 .env.local  
17 .env.development.local  
18 .env.test.local  
19 .env.production.local  
20
```

L'avantage ?

- **Protection des données sensibles** : En spécifiant les fichiers et répertoires sensibles dans le fichier `.gitignore`, on les empêchent de se retrouver dans le dépôt Git.

Donc les fichiers contenant des clés d'API, des mots de passe ou d'autres données confidentielles ne seront pas visibles en dehors du code voulu (exemple : « `.env` »). Et donc on réduit le risque de divulgation accidentelle de ces données.

- **Exclusion des fichiers générés** : Les fichiers générés automatiquement (par exemple, les fichiers binaires, les fichiers de build ou les journaux d'erreurs) peuvent être exclus du suivi Git en les spécifiant dans `.gitignore`. Cela évite que des fichiers volumineux ou inutiles soient ajoutés au dépôt (exemple : `node-modules`, plusieurs milliers de fichiers inutiles à stocker dans Git).

- **Prévention d'inclusions malveillantes** : En ayant un fichier `.gitignore` bien configuré, on réduit le risque d'inclusion accidentelle de fichiers malveillants ou non autorisés dans le dépôt. Donc on peut se protéger contre des attaques potentielles liées à l'injection de code ou à l'ajout de fichiers malveillants.

Dans le cadre de notre formation, nous avons abordé un point extrêmement important concernant le droit et les lois qui entourent notre métier. Imaginons que l'on me demande de concevoir un algorithme destiné à piloter automatiquement des voitures électriques, et qu'un jour cette même voiture soit impliquée dans un accident. Qui en serait responsable ? La société qui a fabriqué la voiture ? Le passager ? Le développeur qui a écrit le code ? Toutes ces questions suscitent des débats complexes. Par conséquent, nous avons eu la chance de rencontrer une avocate au cours qui a résumé le contexte. (**C25**)

RGPD

RGPD : Le Règlement Général sur la Protection des Données est une loi européenne qui concerne la manière dont les données personnelles sont traitées. En gros, il garantit que les entreprises doivent prendre des mesures pour protéger les informations personnelles de leurs utilisateurs et que donc elles doivent demander la permission de collecter ces données, les protéger de manière sécurisée et permettre aux gens de les supprimer s'ils le souhaitent. C'est la base de la protection de la vie privée des gens en ligne.

Dans notre application **Purrrpaws**, nous avons donc ajouté un bouton d'acceptation des **CGU** (Conditions Générales d'Utilisation) qui nous autorise à agir conformément à ce qui est stipulé dans notre règlement. Tant que nous ne transgressons pas la loi, nous avons la possibilité (techniquement) de gérer les données enregistrées dans notre base de données comme bon nous semble. Il faut savoir que ces données ont une valeur, et que nous pouvons parfaitement les partager avec des organismes qui les utiliseront pour personnaliser leurs publicités de manière plus ciblée (chose très fréquente).

En partant de ce principe, nous avons décidé de ne pas être trop excessif et de nous catonner aux données qui nous intéressent à savoir :

- Le nom / prénom de l'utilisateur,
- Les coordonnées (mail, tel),
- Les données de l'animal (nom, poids, sexe, castration, photo, âge),
- Le lieu de résidence,
- Le type de croquette que va recevoir la machine

Sachant que tout ceci a une valeur, nous avons pris la décision de tout vendre. Après tout, pourquoi pas ?

Plateforme d'hébergement

Toujours dans notre formation, on nous a montré **AWS : (C 22)**

Amazon Web Services est une plateforme de cloud computing. Elle permet de louer des ressources informatiques à la demande, comme des serveurs ou du stockage, via internet. AWS offre un large éventail de services pour héberger des applications et des données. Les utilisateurs paient en fonction de leur utilisation pour de nombreux besoins informatiques.

Je pensais que c'était vraiment génial comme plateforme, jusqu'à ce que je vois le système économique. Le meilleur avantage à mes yeux c'est la sécurité, ça évolue en permanence et c'est très complet. Par contre l'inconvénient principal, l'argent ! Si on ne fait pas attention à l'optimisation des couts de manière permanente, il peut y avoir des factures inattendues (vous remarquerez le pluriel). Si bien que quand on travaille avec eux, avant toute chose, il faut regarder combien ça va coûter à la fin, et ensuite seulement de quoi a-t-on réellement besoin. Donc non, hors de question d'utiliser cette plateforme.

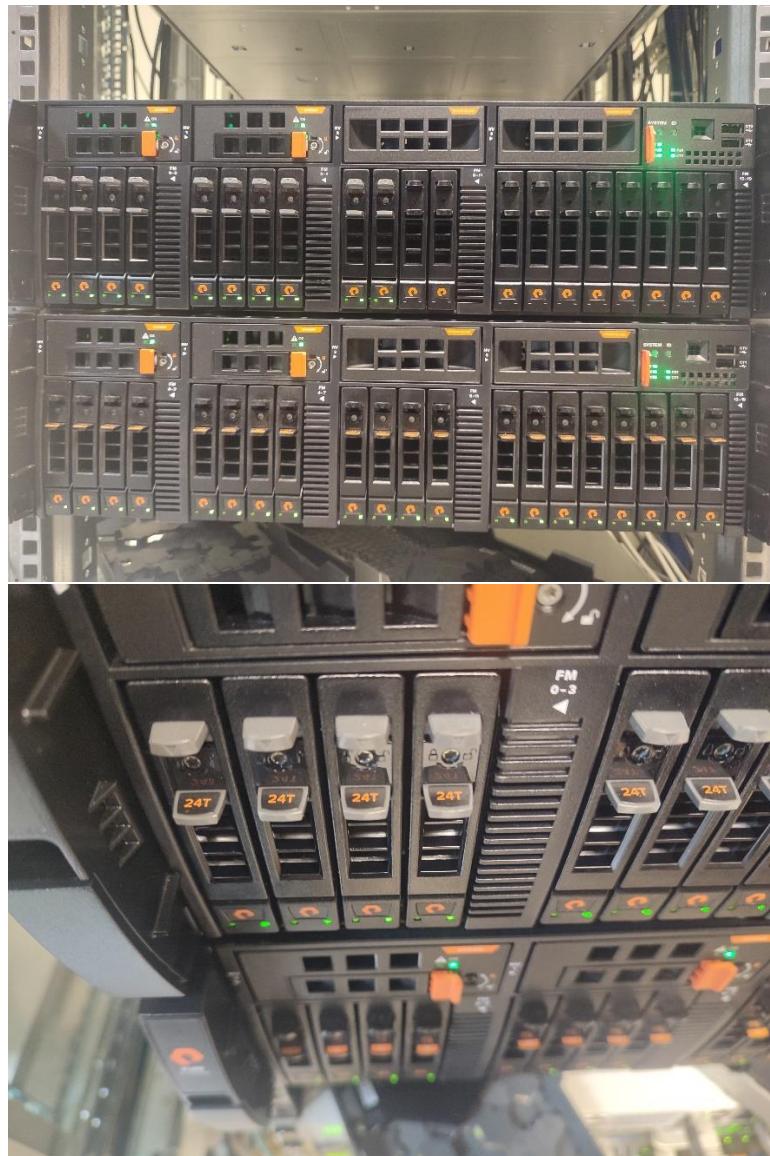
Pour rebondir sur ce sujet, mon entreprise **AEF** détient son propre data center. Le budget alloué cette année dépasse les six chiffres. Leur ancienne base de données était à 99,9 % de charge cette année en septembre. Il a fallu intervenir en supprimant des données stockées. Le choix s'est porté sur des données vieilles de plusieurs années et devenues obsolètes avec le temps, ce qui a permis de gagner quelques giga-octets. Précisons rapidement quand même ce qu'est un octet :

Un **bit** est l'unité de base de l'informatique, il peut avoir 2 valeurs : 0 ou 1. Il constitue le fondement du langage binaire.

Un **octet** est composé de 8 bits. C'est grâce à cette unité que nous mesurons la plupart des données de stockage. Et enfin, les multiples pour faciliter les quantités qui sont :

Le **Méga-octet** (1 million), le **Giga-octet** (1 milliard) et le **Téraoctet** (1000Go ou 1 trillion d'octet).

Mais quand le nouveau matériel est arrivé, tout a évolué très vite. Pour la même quantité de données, les serveurs sont passés de 98 % à 4 % !! Et voici une partie des éléments qui composent actuellement le Datacenter de mon entreprise.



Comme vous pouvez le voir, chaque partie peut contenir **24 To** de données, alignés sur plusieurs étages. Redoutable d'efficacité. Si j'avais le choix, je ferai exactement comme eux, pas de latence car proximité évidente, sécurité maximum, pas de surprises au niveau des factures, quantité de stockage suffisante pour plusieurs années. Je signale également le fait qu'il y'a un **backup** (si tout disparait, les données sont sauvegardées dans un autre Datacenter en toute sécurité).

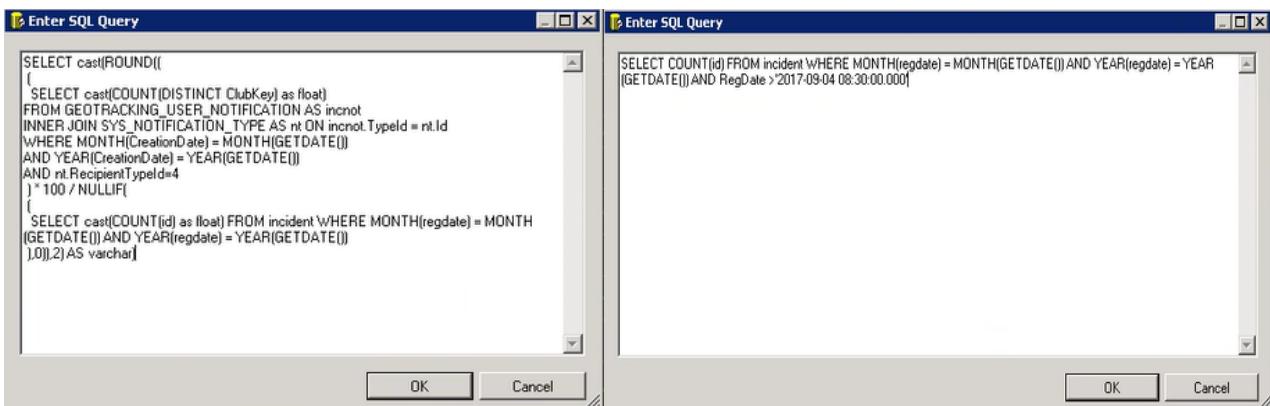
SSIS

SSIS : ou **SQL Server Intégration Services**, c'est un outil de **Microsoft SQL Server** qui permet de gérer et d'automatiser le processus **ETL**. Pour le résumé rapidement, c'est un script **SQL** qui va exécuter automatiquement une série d'action dans un système (Dans l'exemple suivant ce sera une Base de

données) et rendre quelque chose en sortie (Ici un mail qui va faire une synthèse en pourcentage du nombre de dossiers traités ou pas, ainsi qu'un compteur de notifications) (**C27**)

ETL : ou **Extraction** (récupération des données), **Transformation** (modification des données si nécessaire) et **Load** (chargement des données dans une base de données). C'est un ensemble de processus utilisés pour collecter des données à partir de sources variées (essentiellement tout ce qui peut contenir de la donnée), les transformer et les charger dans une base de données.

Dans les screenshots suivant, je vous montre des SSIS qui vont rassembler, modifier et afficher, dans un mail, un suivi des performances de l'application « **Espace Commun** » chez AEF (C'est une application de mise à disposition des données d'incidents pour les clients).



```
SELECT cast(ROUND((  
    ( SELECT cast(COUNT(DISTINCT ClubKey) as float)  
        FROM GEOTRACKING_USER_NOTIFICATION AS incnot  
        INNER JOIN SYS_NOTIFICATION_TYPE AS nt ON incnot.TypeId = nt.Id  
        WHERE MONTH(CreationDate) = MONTH(GETDATE())  
        AND YEAR(CreationDate) = YEAR(GETDATE())  
        AND nt.RecipientTypeId=4  
    ) * 100 / NULLIF(  
    ( SELECT cast(COUNT(id) as float) FROM incident WHERE MONTH(regdate) = MONTH  
        (GETDATE()) AND YEAR(regdate) = YEAR(GETDATE())  
        ,0),2) AS varchar)  
  
SELECT COUNT(id) FROM incident WHERE MONTH(regdate) = MONTH(GETDATE()) AND YEAR(regdate) = YEAR  
(GETDATE()) AND RegDate > '2017-09-04 08:30:00.000'
```

Et donc le visuel final dans le mail (Les données sont réelles, merci de ne pas prendre de notes)

Bonjour,

Nombre de Dossiers Ouverts : 361.

Nombre de Notifications MyAssist créées : 789.
Nombre de Notifications MyAssist vues : 440.
Pourcentage de Notification MyAssist vues : 55.77 %.

Nombre de dossiers avec sollicitation d'une géolocalisation : 4.
Pourcentage de dossiers avec sollicitation d'une géolocalisation : 1.11 %.
Nombre de dossiers avec géolocalisation effectuée par le bénéficiaire : 4.
Pourcentage de dossiers consultés avec géolocalisation effectuée par le bénéficiaire : 100 %.
Pourcentage de dossiers avec géolocalisation refusée : 50 %.
Pourcentage de dossiers avec géolocalisation expirée : 0 %.
Pourcentage de dossiers avec géolocalisation impossible : 0 %.
Pourcentage de dossiers avec géolocalisation trop longue (abandonnée) : 0 %.
Pourcentage de dossiers Géolocalisés : 50 %.
Temps de géolocalisation moyen d'un dossier : 77 secondes.

Nombre de Notifications Client créées : 120.
Nombre de Notifications Client vues : 12.
Pourcentage de Notification Client vues : 10 %.

Nombre de Notifications Acta créées : 88.
Nombre de Notifications Acta vues : 10.
Pourcentage de Notification Acta vues : 11.36 %.

Nombre de SMS envoyés : 870.
Coût des SMS envoyés : 52.2 euros.

Nombre d'enquêtes sollicitées : 220.
Nombre de réponses aux enquêtes : 0.
Pourcentage de réponses aux enquêtes : 0 %.

Transactions PayPal

Le code qui suit est une base pour intégrer des paiements **PayPal** dans une application web en collectant des données du formulaire et en utilisant **l'API PayPal** pour effectuer la transaction. Le

mode « **sandbox** » est utilisé pour effectuer des tests sans effectuer de transactions réelles (Le code réel est un peu trop long, je vais en expliquer le principal). (**C29**)

```
1  <?php
2  require_once 'vendor/autoload.php';
3
4  // On call les API
5  use PayPal\ApiItem;
6  use PayPal\ApiItemList;
7
8  $apiContext = new PayPalRestApiContext(
9      new PayPal\Auth\OAuthTokenCredential(
10          '18C713N7',           // ClientID
11          '53C83780085'        // ClientSecret
12      )
13 );
14
15 $apiContext->setConfig(
16     array(
17         'log.LogEnabled' => true,
18         'log.FileName' => 'PayPal.log',
19         'log.LogLevel' => 'DEBUG',
20         'mode' => 'sandbox', // 'live' or 'sandbox' Pour tests
21     )
22 );
23
24 // Création client
25 $payer = new PayPal\ApiPayer();
26 $payer->setPaymentMethod('paypal');
27
28 // Création de l'objet vendu
29 $itemSold = new Item();
30 $itemSold->setName($_POST['item'])
31     ->setCurrency('USD')
32     ->setQuantity(1)
33     ->setPrice($_POST['amount']);
34
35 // Création de la liste qui contiendra les objets vendu
36 $itemList = new ItemList();
37 $itemList->setItems(array($itemSold));
38
39 // Création de la somme totale
40 $amount = new PayPal\ApiAmount();
41 $amount->setTotal($_POST['amount']);
42 $amount->setCurrency('USD');
43
44 // Regroupement de la liste et de la somme
45 $transaction = new PayPal\ApiTransaction();
46 $transaction->setDescription("Payment For Service")
47     ->setItemList($itemList)
48     ->setAmount($amount);
```

1. Inclusion des bibliothèques :

- Le code commence par inclure les bibliothèques nécessaires pour appeler avec l'API PayPal.

2. Configuration du contexte PayPal :

- On crée un objet **\$apiContext** et on le configure avec les informations d'identification du client PayPal (Client ID et Client Secret).
- Le mode "sandbox" est spécifié pour effectuer des tests.

3. Définition du payeur :

- Un objet **\$payer** est créé pour spécifier la méthode de paiement, qui est "**paypal**".

4. Définition de l'article :

- On crée l'objet **\$itemSold** en récupérant des données depuis un formulaire **POST** (nom de l'article et montant).
- Cet article sera en dollars (**USD : Dollar United State**) et la quantité est fixée à 1 (ligne 31 => 33).

5. Crédit d'une liste d'articles :

- Un objet **\$itemList** est créé et contient l'article qu'on vient de créer.

6. Définition du montant de la transaction :

- L'objet **\$amount** est créé avec le montant total récupéré depuis le formulaire **POST** à la ligne 30 (USD)

7. Création de la transaction :

- On rassemble tout ce qui a été créé plus tôt en le mettant dans l'objet **\$transaction** et on attribue la description de la transaction (Le paiement), la liste d'articles et le montant.

8. Définition des URL de redirection :

- On ne le voit pas sur le screenshot mais, on crée l'objet **\$redirectUrls** pour spécifier les URL de retour après le paiement, comme la page de succès et d'annulation.

Ensuite, on fait la **Création du paiement** : **\$payment** en définissant l'intention « vendu », le payeur, la transaction, et les URL de redirection.

Pour finir, on gère les cas **d'erreur** lors de la création du paiement, le code gère l'exception « **PayPalExceptionPayPalConnectionException \$ex** » en affichant des informations de débogage.
(echo \$ex->getData();

CI/CD (Partie 2)

Comme mentionné précédemment ([Page 28](#)), le concept de **CI/CD (Continuous Integration/Continuous Deployment)** est une approche de développement logiciel visant à automatiser la construction, les tests et le déploiement des applications de manière continue. Cette automatisation permet de simplifier les mises en production ou d'optimiser les opérations répétitives, notamment en utilisant des scripts en ligne de commande pour accélérer le processus.

Imaginons que pour créer un nouveau client souhaitant acheter notre produit, la personne en charge doit accéder à la **base de données**, puis créer une nouvelle ligne contenant le nom, le prénom, l'ID, le produit acheté et le montant de la transaction manuellement. Une fois cela fait, nous devons accéder à **un tableau Excel** où nous conservons l'ID, les transactions et le prix. Ensuite, nous devons accéder à **un autre tableau** où nous enregistrons le nom, le prénom et l'ID. Enfin, nous devons accéder à **un dernier tableau** pour enregistrer toutes les données concernant la personne.

Mettre bout à bout toutes ces étapes impliquent plusieurs détails intéressants : le risque d'erreur de frappe est important, tout comme le risque d'oublier de remplir l'un des tableaux.

En automatisant l'ensemble de ce processus, nous pourrions créer un script qui récupérerait toutes ces données et les déployerait dans les bons tableaux et dans la base de données en une seule fois et de manière automatique. Pour illustrer cela, le code pourrait être contenu dans un fichier nommé « **CDbash.sh** » et ressemblerait à ceci :

```

1  #!/bin/bash
2
3  # Informations de connexion à la base de données
4  DB_USER="utilisateur"
5  DB_PASSWORD="mot_de_passe"
6  DB_NAME="base_de_données_Y"
7
8  # Informations sur le nouveau client
9  NOM="Bart"
10 PRENOM="Haba"
11 ID="13340"
12 PRODUIT_ACHETE="Cigarettes"
13 MONTANT_TRANSACTION="777.00"
14
15 # Insère un nouveau client dans la base de données
16 mysql -u $DB_USER -p$DB_PASSWORD $DB_NAME << EOF
17 INSERT INTO clients (nom, prénom, ID, produit_acheté, montant_transaction)
18 VALUES ('$NOM', '$PRENOM', '$ID', '$PRODUIT_ACHETE', '$MONTANT_TRANSACTION');
19 EOF
20
21 # Accède au fichier Excel (ceci est un exemple bien sûr)
22 FICHIER_EXCEL="parsskonvieng2Loains.ods"
23
24 # Étape 1 : Ajouter les données du nouveau client à la feuille "Transactions"
25 echo -e "$ID\t$PRODUIT_ACHETE\t$MONTANT_TRANSACTION" >> transactions_temp.txt
26
27 # Étape 2 : Ajouter les données du nouveau client à la feuille "Identité"
28 echo -e "$NOM\t$PRENOM\t$ID" >> identite_temp.txt
29
30 # Étape 3 : Ajouter les données du nouveau client à la feuille "Toutes Données"
31 echo -e "$NOM\t$PRENOM\t$ID\t$PRODUIT_ACHETE\t$MONTANT_TRANSACTION" >> toutes_donnees_temp.txt
32
33 # Ajoute la logique pour manipuler les fichiers Excel|
34 echo "Nouveau client ajouté à la base de données et aux feuilles Excel. Bienvenue $NOM $PRENOM"

```

Le fait de remplir ce « **bash** » fera gagner du temps à la personne qui gère les nouvelles inscriptions et évitera au passage les erreurs potentielles (Sans parler bien sûr du bien que cela fera à l'utilisateur).

Sécurité

1. HTTPS : le protocole HTTPS (*Hyper Text Transfer Protocol Secure*) est une extension sécurisée du protocole HTTP, le « S » pour « Secured » signifie que les données échangées entre le navigateur de l'internaute et le site web sont chiffrées et ne peuvent en aucun cas être espionnées (confidentialité) ou modifiées (intégrité). Pour obtenir le fameux S, il faut passer par l'acquisition et l'installation d'un certificat SSL/TLS auprès d'une Autorité de Certification reconnue.

Ce certificat garantit l'authenticité du site Web et assure la confidentialité des données pendant la transmission. Cela affichera ainsi le HTTPS, le cadenas vert et le mot « Sécurisé » dans la barre d'adresse du navigateur.

A noter qu'il existe plusieurs certificats avec plusieurs niveaux d'authentifications :

- Le DV (**Domain Validation**) considéré comme de l'authentification faible, il valide le nom de domaine pour lequel le certificat SSL a été demandé avec l'approbation du demandeur.
- Le OV (**Organization Validation**) qui elle serait considérée à authentification forte. Comme le DV mais avec en plus un examen de l'organisation. Des informations complémentaires sur l'organisation sont affichées sur le sceau du certificat SSL.
- Et le EV (**Extended Validation**) qui est une authentification renforcée. Comme le OV, avec un contrôle plus complet de l'organisation. Les règles pour l'attribution d'un certificat SSL EV sont définies par le forum CA/Browser et strictement contrôlées.

2. Les tokens : Les tokens sont en fait des clés numériques. Ils sont générés de manières aléatoires et utilisés pour sécuriser les transactions financières, accéder à des ressources ou authentifier des utilisateurs. Ils peuvent être des nombres, des lettres, ou une combinaison des deux, et ils sont souvent générés de manière à être uniques et difficiles à prédire (comme des clés pour des portes dans la vie réelle).

Par exemple, pour l'authentification, lorsqu'un utilisateur se connecte à un service en ligne, on génère un Token et ensuite, il est utilisé pour vérifier l'identité de l'utilisateur.

Pareil pour les **API**, on les utilise pour autoriser l'accès à des bases de données par exemple ou tout autre ressources protégées.

Pour les transactions financières, on utilise les Tokens pour sécuriser les transactions de manières unique, ce qui permet d'empêcher les fraudes plus facilement.

3. Les Mots de passe en base de données : Afin d'empêcher toute lecture possible de qui que ce soit, on nous a appris à « **hacher** » le mot de passe pour ne pas pouvoir se connecter à la place d'un utilisateur. Je précise que le hachage est un processus à sens unique et qu'il est très compliqué de revenir au mot de passe d'origine. Pour notre projet, on a utilisé le **SHA256**.

4.RGPD : Le **Règlement Général sur la Protection des Données** est un ensemble de règles de l'Union européenne visant à renforcer la protection des données personnelles des citoyens de l'UE. Ce règlement s'applique à tous ceux qui vivent en Europe mais aussi à ceux qui n'y sont pas basés. Le traitement des données personnelles doit se faire de manière transparente et loyale sous peine de grosses pénalités (financières bien sûr).

5.OWASP : C'est l'acronyme de **I'Open Web Application Security Project**, c'est une communauté internationale axée sur la sécurité des applications Web. Elle propose des ressources, des outils, des guides et des normes destinées à aider les organisations à améliorer la sécurité de leurs applications et services Web.

L'objectif principal d'OWASP est d'améliorer la sécurité des logiciels en fournissant des ressources gratuites et ouvertes à la communauté mondiale. L'organisation se concentre particulièrement sur la sécurité des applications Web, qui sont souvent la cible d'attaques en raison de leur exposition publique.

L'un des aspects les plus connus d'OWASP est le **Projet Top 10**, qui liste les 10 principales vulnérabilités de sécurité courantes dans les applications Web. Ce document est régulièrement mis à jour pour refléter les nouvelles menaces et les nouvelles techniques d'attaque.

J'ai compris ce que sont les injections SQL et leur impact potentiel sur la sécurité des applications. L'une des méthodes que j'ai utilisées pour renforcer la sécurité des applications faites en cours a été de les protéger au niveau des champs de saisie, notamment l'utilisation des expressions régulières (**regex**). Ces dernières me permettent de valider et de filtrer les entrées utilisateur, en empêchant l'ouverture de balises ou de parenthèses par exemple réduisant ainsi considérablement le risque d'injections SQL. J'ai aussi été surpris de voir que des outils comme « **Burp suite** » sont gratuites et peuvent te permettre de voir ce qu'il se passe entre les « requêtes/réponses » et potentiellement modifier leur contenu.

En parallèle, j'ai aussi pratiqué le site « **Rootme** » et vu comment fonctionnait la base du hack. J'avoue que d'avoir vu et utilisé ce site m'a fait comprendre l'importance de la sécurité dans le code. Si on ne fait pas un minimum attention, les back doors peuvent vraiment être fatale.

Je suis conscient que la sécurité dans les applications et sites web demandent énormément de temps d'apprentissage et même si je ne suis pas encore expert en la matière, j'ai vu et me renseigne régulièrement sur les diverses avancées technologiques qui protègent ou à l'inverse pénètrent les défenses les plus utilisées.

6.Double authentification : La double authentification (2FA), également connue sous le nom **d'authentification à deux facteurs**, est un mécanisme de sécurité qui renforce le processus d'authentification en exigeant deux méthodes distinctes pour vérifier l'identité d'un utilisateur. Ces méthodes peuvent être classées en trois catégories principales : quelque chose que vous savez, quelque chose que vous avez, et quelque chose que vous êtes.

Quelque chose que vous **avez** :

Il s'agit généralement d'un mot de passe ou d'un code PIN, qui est une information secrète que l'utilisateur doit connaître pour accéder à un compte ou à un système.

Quelque chose que vous **avez** :

Il s'agit d'un élément physique que l'utilisateur possède, tel qu'un téléphone mobile, une carte à puce, ou un jeton matériel. Les codes générés par des applications d'authentification mobiles ou les codes envoyés par SMS entrent également dans cette catégorie.

Quelque chose que vous **êtes** :

Il s'agit d'une caractéristique biométrique, telle que l'empreinte digitale, la reconnaissance faciale, ou la reconnaissance de l'iris. Cependant, la biométrie est moins courante dans les systèmes de double authentification en comparaison avec les deux premières catégories.

La double authentification ajoute une couche de sécurité significative en rendant plus difficile pour les attaquants d'accéder à un compte même s'ils ont réussi à compromettre le mot de passe. Même si le mot de passe est divulgué, l'attaquant aurait également besoin du deuxième facteur pour accéder au compte.

7. Chat Gpt : Pourquoi je vous parle de chat GPT (ou tout autres IA performantes à venir) ?

Il se trouve que Chat GPT est maintenant capable de « repérer » les emplacements des boutons via un lien qu'on lui donne en temps réel. Concrètement donc, il serait possible de faire un **batch** (script de code qui exécute une suite d'instructions) ou on crée (par exemple) un nouveau compte, puis au moment de la vérification via un click bouton « **Vous n'êtes pas un robot** » demander au batch de se connecter a Chat GPT, puis de lui demander via un script préétabli et testé « Donne-moi la localisation de la fenêtre à cliquer vous n'êtes pas un robot en me donnant l'endroit horizontal et vertical en pixels» puis de récupérer cette réponse et d'appuyer à cet endroit via un event pour créer un nouveau compte de manière totalement automatisée. Cette utilisation pourrait être une grosse faille de sécurité future pour les sites utilisant cette méthode pour empêcher toute utilisation frauduleuse de leur site ou application.

Je précise que la requête ainsi que le but de cette opération sont volontairement inutiles en l'état.

Une solution à ce problème serait d'utiliser un drag and drop puzzle.

En ce qui concerne les puzzles, en ajustant le script pour qu'il identifie non pas un, mais deux endroits (l'emplacement de la pièce du puzzle à déplacer et le point d'arrivée dans le dessin à compléter avec la pièce), et en modifiant l'événement pour déplacer la pièce du puzzle du point "A" au point "B" sans relâcher le clic, cette méthode de sécurité est déjà plus difficile à vaincre qu'un simple bouton « **Vous n'êtes pas un robot** » (même si avec le temps et des efforts supplémentaires, cette méthode aussi deviendra obsolète).

Chat GPT peut à la fois être un outil formidable et mais également une menace non négligeable pour la sécurité des entreprises (selon moi).

DEVOPS

L'objectif du DEVOPS est d'automatiser, optimiser et accélérer le déploiement des logiciels, favorisant ainsi une collaboration fluide entre les développeurs et l'infra, des mises en production rapides et une amélioration continue des processus.

Mise en Production : La "Mise en Prod", est la phase finale du cycle de développement d'un logiciel. Elle consiste à déployer la version finale d'une application ou d'un service sur l'environnement de production, c'est-à-dire l'infrastructure réelle utilisée par les utilisateurs finaux. Avant cette étape, il y'a la mise en **recette** qui est fondamentalement la même chose, mais dans un environnement de **test** pour diminuer le risque d'échec de mise en production.

Accès serveurs : Les serveurs hébergeant les bases de données d'Arc Europe France requièrent une sécurité renforcée pour prévenir tout accès non autorisé. Cette sécurité s'articule autour de conventions de nommage strictes. L'étiquète se doit de correspondre au produit (exemple, **BDDTest** s'occupe exclusivement des données en environnement de test, et **APPTest** s'occupe uniquement des applications dans le même environnement).

De plus, les droits d'accès aux données sont configurés, en fonction de notre poste au sein de l'entreprise, garantissant que seules les personnes autorisées peuvent interagir avec les informations stockées. Cette approche vise à assurer l'intégrité et la confidentialité des données sensibles, tout en limitant les risques liés à d'éventuelles intrusions externes.

Entente avec l'infra : Actuellement, AEF n'est pas à 100% « **DEVOPS** » mais essai vraiment de le devenir. Il y'a des points à améliorer, toujours selon moi, pour devenir plus proche de l'idée du DEVOPS. Je pense notamment à l'automatisation du déploiement ainsi qu'à un environnement supplémentaire de PROD pour du load balancing.

Le but étant de pouvoir automatiser au maximum les déploiements, en basculant d'un environnement sain et vérifié à l'environnement de PROD à traiter en surveillant toute déviance potentielle. Cela assure également de ne pas travailler dans l'urgence à chaque Mise en Prod.

Le fait de ne pas couper la PROD, et de prendre le temps de vérifier l'environnement complet, l'utilisation des sites et applications concernées, serait un vrai plus pour vérifier à chaque itération que le déploiement automatique a bien été exécuté sans l'aide de l'infra. Ce qui serait un pas de plus dans l'objectif DEVOPS d'AEF.

Conclusion

En conclusion, je dirai qu'il est extrêmement compliqué de compiler deux années de ma vie en quelques lignes. J'ai réellement essayé de vous représenter dans l'ordre dans lequel s'est déroulé ma formation et mon stage tout en détaillant au mieux les demandes du référentiel.

J'ai appris énormément de choses ces deux dernières années, sur pleins de sujets différents, que ce soit l'infrastructure, les langages, la logique de code, les systèmes. Cette formation full stack aura été une aubaine pour moi.

Avec tout ce que l'on a vu cette année (qui est passée à une allure ahurissante), je me suis rendu compte que certains élèves avaient des affinités pour certaines activités et moins pour d'autres. Tout comme moi d'ailleurs. Je pense être un « Fronteur », avoir une mission claire ainsi qu'une ou plusieurs images de références et me demander de me rapprocher le plus possible de celle-ci, voire du « pixel perfect » me challenge au plus haut point. Jouer avec le nom des variables et le CSS pour atteindre mon but est plaisant pour moi.

Je ne boude pas le Backend mais, forcé de constater que je prends moins de plaisir en back qu'en front. Postman, Mockoon et les API est très captivant également. Echanger avec le client pour bien comprendre son besoin, demander de répéter aux autres pour être sûr que tout le monde a bien compris la même chose avant de commencer à travailler est une étape plaisante à mes yeux.

Nous avons également vu tout un pan de la sécurité informatique (OWASP, rootme) et même si le sujet est primordial pour la protection des données de l'utilisateur ou de l'ordinateur lui-même, ce sujet est long et fastidieux pour moi.

Hors contexte du code et du développement, cette année a surtout été une période de rencontres que ce soit l'administration de l'IT-Akademy ou de ces élèves (avec qui je resterai en contact même après cette expérience a n'en pas douter), les professeurs avec qui j'ai pu sentir la passion de ce métier on ne peut plus gigantesque qu'est le développement. Et enfin mes collègues de travail à Arc Europe France qui sont tous ce que je cherche dans une équipe avec laquelle je travaille : bienveillance, rires, entraide et conscience professionnelle (Jihène, Aurélien et Damien).

J'ai dû quitter mon ancien emploi dans lequel j'étais respecté, avais une équipe et un poste à responsabilité, un travail dans lequel j'étais compétent (même si je le suis toujours à n'en pas douter), parce que le futur que me promettait mon entreprise ne correspondait pas avec mes attentes. J'ai tout risqué pour cette nouvelle vie qui se déroule depuis deux ans maintenant avec tous les challenges et difficultés qui l'accompagnent.

Faire une reconversion professionnelle à 38 ans s'est révélé être un véritable défi pour moi. Je n'ai jamais eu peur de me dépasser ou de m'améliorer, peu importe la difficulté, peu importe le temps que cela peut prendre, et je ne doute pas une seconde du fait que mon niveau dans le futur sera bien supérieur à l'actuel. Le passé est important mais le futur est celui pour quoi je passe cette épreuve ainsi que celles qui suivront.

Pour finir, je tiens à remercier chaleureusement toutes les personnes qui ont cru en moi et m'ont soutenu tout au long de ce périple, toutes celles et ceux qui m'ont aidé dans les temps de doutes que j'ai pu avoir et tous ceux qui auront pris le temps de lire cette documentation sur ma soutenance.

Un grand merci à ma femme que j'aime, ma famille et mes amis qui m'aide sur tellement de sujets.

Je ferai tout pour devenir meilleur sans jamais perdre de vue celui que je suis.

Merci à tous !

CV



Contact

Téléphone
0663916563

Email
marcdelarrea@gmail.com

Linked In
www.linkedin.com/in/marc-de-larrea

Adresse
58 av Général LECLERC
69100 VILLEURBANNE

Formations

2023
Formation développeur Full Stack
Actuellement en reconversion professionnelle

2018
Diplôme d'ingénieur
Conducteur d'équipements industriels

2001
CAP Techniverrier
Fabrication et manipulation du verre

Langues

Français / Langue maternelle

Anglais / Intermédiaire

Loisirs

- Roman Héroïque Fantasy
- Jeux vidéos
- Manga
- Maitre de Jeu de rôle

Déplacements

Titulaire permis B

Marc DE LARREA

Développeur Web Front-end

PROFIL PROFESSIONNEL

Actuellement en formation de développeur full stack en reconversion, je suis passionné par le front-end et suis à la recherche d'un Emploi.

J'ai acquis des connaissances en HTML, CSS, JS, SQL, ASP.net et Bootstrap, je me suis également familiarisé avec des technologies telles que Node.js, Express, React, TFS, SonarQube, LLBL GEN Pro 5, GIT et les API.

Fort d'une expérience variée en industrie, mon ambition est de contribuer au sein d'une équipe agréable et de mettre mes compétences au service d'un groupe motivant.

J'aime travailler en équipe, participer à des projets stimulants et acquérir de nouvelles compétences pour continuer à évoluer dans ce domaine captivant.

Compétences

- Fortes compétences de communication orale et écrites
- Ouverture d'esprit
- Connaissances des technologies de développement web telles que :
 - **HTML, CSS, Bootstrap**
 - **JavaScript, Jquery (node, express)**
 - **SQL (MySQL, mongoDB) LLBL Gen Pro 5**
 - **ASP.net, React**

Parcours Professionnel

2023 - 2023

Arc Europe France / Limonest

○ Développeur Web, Support informatique

- Gestion des bugs de l'application Cactus.
- Contribution à la résolution des problèmes en apportant les solutions adaptées et en formulant des propositions d'amélioration.
- Application d'un code eco responsable et moderne (Green IT)
- Identification des besoins en écoutant attentivement les demandes formulées et les spécificités de l'environnement.
- Sprint Agile et Scrum quotidien.
- Adaptation à la situation en fonction des problèmes retournés par les clients.

2016 - 2021

Aldes aérolithe / Vénissieux

○ Formateur en entreprise

- Construction et suivi des parcours individualisés des apprenants.
- Application d'une pédagogie différenciée en fonction du profil d'apprentissage.
- Évaluation de la progression et des acquis des apprenants, mise en place d'actions correctives dans le cadre des évaluations en cours de formation.
- Gestion des feuilles de présence.