

# Rapport de Stage

Loris DROUHOT

16 juin 2025

# Remerciements

Je souhaite remercier M. François LALAY, développeur web full-stack, qui fut maître de stage pendant ces 10 semaines, pour sa patience, sa confiance mais surtout sa pédagogie. Apprendre à ses côtés fut un réel plaisir. Il a su me guider afin de progresser sans pour autant tout m'expliquer à chaque fois, ce qui m'a forcé à rechercher par moi-même afin d'obtenir des réponses à mes questions techniques.

Je tiens aussi à remercier M. Ronan MÉVELLEC, Directeur, qui a pris le temps, une après-midi, dans son programme chargé, de me rencontrer afin que l'on puisse échanger sur ma situation, le stage, ainsi que de m'expliquer précisément ce qu'était l'ATD 16.

Je remercie aussi l'ensemble de l'équipe de l'ATD 16 qui m'a très bien accueilli, pris le temps de m'écouter lors de réunions, mais aussi de m'expliquer leur travail, ce qui fut très enrichissant. Plus précisément : Lionel CLERCQ, Responsable du pôle numérique, Charlotte CIARDULLI et Perrine MADIOT, toutes deux chargées du support administration numérique et logiciels métiers, ainsi que Pierre SAUZE, expert en numérique, pour leur temps et leurs conseils lors des réunions hebdomadaires du projet de développement.

Enfin, je remercie aussi mon professeur référent, M. Sébastien FAUCOU.

# Résumé

Dans le cadre de ma formation de Bachelor Universitaire Technologique deuxième année à l'Institut Universitaire Technologique de Nantes, j'ai réalisé un stage de 10 semaines au sein de l'Agence Technique du Département de la Charente. Lors de ce stage, j'ai mis en pratique ainsi que développé mes connaissances et compétences en développement web. J'ai donc développé une nouvelle page au site interne de l'agence, page de simulation permettant à un agent de calculer le prix qu'aurait à payer un adhérent pour les politiques auxquelles il souhaite adhérer en rentrant les variables nécessaires (nombre d'habitants, voirie, etc.)

To complete my second year at the l'Institut Universitaire Technologique de Nantes, I had to validate a 10 week internship at ATD16. During this internship, I used and developed my knowledge and skills in web development. I created a new page on the intern website of the agency. This is a simulating page that allows an agent to calculate a price that a possible member could pay for the services this possible member would like to have by entering the necessary variables (number of habitants, total roads kilometer, etc.)

# Table des matières

<b>Remerciements</b>	<b>1</b>
<b>Résumé</b>	<b>2</b>
<b>1 Introduction</b>	<b>5</b>
<b>2 Développement</b>	<b>7</b>
2.1 L'Agence Technique Départementale de la Charente . . .	7
2.1.1 Fonctionnement économique . . . . .	7
2.1.2 Environnement de Travail . . . . .	8
2.2 Explication de la mission . . . . .	8
2.3 Les outils . . . . .	9
2.3.1 Git . . . . .	9
2.3.2 Gestion de Projet . . . . .	10
2.3.3 Les documentations et l'IA . . . . .	10
2.4 Le simulateur . . . . .	11
2.4.1 L'API Numérobis . . . . .	11
2.4.2 Le controller simulateur . . . . .	12
2.4.3 Le site Numérobis . . . . .	14
2.5 Refactoring avec Zustand . . . . .	19
2.6 Autres petites missions . . . . .	21
<b>3 Conclusion</b>	<b>22</b>

Tous les mots suivis d'un astérisque sont définis dans le glossaire

# Chapitre 1

## Introduction

Étudiant en deuxième année de BUT Informatique, je dois réaliser un stage en entreprise d'une durée de 10 semaines afin de valider mon année. Ce stage se déroule donc entre le 22 avril 2025 et le 27 juin 2025, le but de ce stage est de mettre en pratique les compétences acquises dans notre cursus, ainsi que nous faire découvrir le monde professionnel en informatique.

J'ai eu la chance de pouvoir réaliser mon stage à l'Agence Technique Départementale de la Charente (ATD 16), j'ai donc été intégré à son pôle numérique en compagnie de François LALAY, mon maître de stage, unique développeur du site de l'agence. Étant donné des problèmes avec le GitLab de l'agence la semaine de mon arrivée, on m'a proposé de réaliser un mini-projet avec les technologies utilisées dans le site de l'agence, afin de me familiariser avec ces technologies, sont le framework JavaScript Next.js pour le front-end, et le framework PHP Symfony ainsi que Api-Platform pour le back-end. Une fois les problèmes résolus, j'ai pu commencer ma première mission, qui était le développement d'une page de simulateur sur le site interne à l'agence, et donc développer aussi bien la partie front-end que back-end

Afin de réaliser cette mission de la meilleure manière et le plus efficacement possible, on m'a demandé de réaliser moi-même un cahier des charges et des wireframe\*. Lors des réunions hebdomadaires avec certains agents du pôle numérique, j'ai présenté ce cahier des charges, on m'a alors précisé l'attendue de la mission. Une fois les bases et besoins établis, j'ai pu commencer le développement du simulateur. J'ai naturellement commencé par le back-end, où j'ai donc dû comprendre le fonctionnement de l'API déjà en place, toutes ses entités et ses différentes fonctionnalités. J'ai, dans le même temps, pris connaissance de la grille tarifaire des adhésions à l'ATD 16, ainsi que toutes ses exceptions qui sont de vrais problèmes à implémenter.

Dans le but de compléter la mission qui m'a été confiée, j'ai utilisé de nombreux outils, allant de la connaissance théorique aux documentations

des différents langages de programmation. Le concept le plus important quand j'ai travaillé sur la partie back-end de ma mission a été la serialisation et donc aussi la deserialization, ce sont les processus centraux du fonctionnement d'une API rest. Ce sont eux qui assurent que les données soient converties au bon format pour l'envoi, puis convertissent les données reçues au format du langage utilisé. Ces concepts sont intégrés à Symfony et Api-Platform nativement, il suffit de bien les utiliser. Il m'a aussi fallu me pencher sur d'autres concepts plus spécifiques aux outils utilisés, comme le type de route API, les différents événements attachés à ces routes. Pour la partie front-end, la théorie est moins présente car les données et le site sont spécifiques, j'ai donc été amené à utiliser des outils spécifiques aux frameworks avec lesquels on travaille, comme les hooks React ou les stores Zustand.

Pour trouver les solutions à la plupart de mes problèmes, un type d'outil fut primordial : les documentations. Quand elles sont bien renseignées, les documentations ont souvent la réponse à tous nos problèmes généraux pour un langage ou un framework. Lorsque j'avais des problèmes plus précis, alors les forums ou les IA sont aussi des outils intéressants, même s'ils ont des limites naturelles posées par le fait qu'ils ne connaissent pas notre projet, il s'agit alors de trouver un cas général donnant une idée de la solution à notre problème.

Dans un premier temps, je présenterai l'agence, son statut spécifique, son fonctionnement, ainsi que son intérêt pour les services publics charrentais.

Dans une seconde partie, je parlerai des outils qui m'ont été utiles ainsi que la réalisation de ma mission de développement

# Chapitre 2

## Développement

### 2.1 L'Agence Technique Départementale de la Charente

L'Agence Technique Départementale de la Charente est un service public. Fondée en 2014, elle regroupait déjà les compétences de l'AMO et du Juridique, puis a intégré les compétences du SDITEC (numérique et cartographique) en 2018. L'agence propose un nombre conséquent de compétences permettant d'aider, de conseiller ou d'assister les différentes collectivités territoriales de Charente. Pour l'organigramme structurel voir annexe

M. Ronan MÉVELLEC, directeur et fondateur, explique cela comme une mutualisation des moyens des collectivités, chaque collectivité n'a pas forcément les moyens de payer des agents pouvant réaliser tous genres de tâches, qu'elles soient numériques, juridiques ou cartographiques. C'est là où l'ATD 16 prend tout son intérêt, elle permet de centraliser toutes compétences (Aménagement, Numérique, Juridique et Cartographique) au sein d'un seul service public, auquel les collectivités peuvent demander de l'aide, à condition d'adhérer aux politiques adéquates.

#### 2.1.1 Fonctionnement économique

L'ATD 16 fonctionne sur un modèle de volets principaux, d'adhésions optionnelles et d'appuis ponctuels. En premier lieu, l'ATD propose deux volets principaux, le volet AMO, incluant l'assistance à maîtrise d'ouvrage et l'assistance juridique, et le volet Numérique, incluant la maintenance du parc informatique, l'administration numérique, un système de convocations électroniques et un profil acheteur sur les marchés publics. Une fois adhérents à l'un ou les deux volets, les collectivités ont le droit d'adhérer aux adhésions optionnelles, comptant par exemple l'entretien de la voirie, le RGPD, un parcours cybersécurité, l'assistance sur logiciels métiers, ou encore la mise à disposition de logiciels de cartographie permettant plein de choses comme la gestion des cimetières par exemple. Un adhérent peut cumuler autant d'adhésions optionnelles qu'il le souhaite,



le prix des volets et des adhésions optionnelles est calculé en général par strates de nombre d'habitants ou bien au nombre d'habitants, aussi pour la part variable du volet numérique ou le volet AMO par exemple. Il existe d'autres variables comme le nombre de boîtes de messagerie pour l'adhésion aux boîtes de messagerie ou encore la voirie pour l'entretien des routes. Pour les détails de la grille tarifaire voir annexe.

Enfin, l'ATD propose aussi des appuis ponctuels, services rendus irrégulièrement sur demande d'une collectivité ; il peut s'agir de suivi d'opérations après la mise en route d'un projet par l'AMO, de prêt de matériel, ou encore d'adressage pour les communes. Il existe plusieurs types de collectivités territoriales pouvant accéder aux services de l'ATD 16, la majorité sont des communes, mais il y a aussi des communautés de communes, des syndicats ou encore le département lui-même.

Étant un service public, l'argent injecté et récolté est surveillé de près, c'est pourquoi il y'a un Conseil d'Administration, présidé par M. Michel Carteret, ce conseil prend les décisions pour l'ATD, qu'elles soient budgétaires, salariale ou toute autre chose. Ce conseil est peuplé de représentants des adhérents à l'ATD 16. Monsieur le directeur ne peut qu'appliquer les décisions du conseil, néanmoins la plupart des idées sont suggérées par l'ATD puis validées par le conseil.

### **2.1.2 Environnement de Travail**

En ce qui est de l'environnement de travail, chaque pôle de l'agence possède son bureau sur le site de la Combe à Angoulême, Charente. Je travaille sur un HP ProBook 450 G7, les 8 Go de RAM sont peu afin de faire tourner le serveur du site web et de l'API pour le développement, mais je m'en sors plutôt bien, j'ai en plus deux écrans iiyama. Pour ce qui est du software, on m'a laissé le choix de l'IDE, j'ai choisi Visual Studio Code, c'est adapté au PC pas très performant que j'avais, quelques extensions comme le visualiseur de base de données sont nécessaires afin de limiter le nombre d'outils. Pour le back-end, comme expliqué dans l'introduction, c'est le framework Symfony qui est utilisé avec Api-Platform et donc le langage PHP. Afin de tester les routes API, j'ai utilisé Postman, logiciel permettant, entre autres, de faire des requêtes API. Maria-Db est utilisé pour avoir une copie de la base de données de production en local. Pour le front-end, comme précédemment dans l'introduction, c'est Next.js qui est utilisé en framework JavaScript. C'est le framework le plus populaire sur le web de nos jours. Tailwind CSS est utilisé pour le style des pages.

## **2.2 Explication de la mission**

La mission principale de mon stage est donc la réalisation d'un simulateur de prix d'adhésions sur le site Numérobis, site interne à l'ATD

16. Ce simulateur, doit pouvoir donner un prix pour une structure adhérente, en tenant compte des variables renseignées. Une simulation existait déjà, mais elle se limitait à une seule nouvelle adhésion et ne prenait en compte que les variables déjà rentré. Cette nouvelle version doit dépasser ces limites, il faut pouvoir calculer une ou plusieurs nouvelles adhésion pour une structure déjà adhérente mais aussi pouvoir simulé de zéro une adhésion d'une nouvelle structure. Une contrainte exprimé dès le début du stage est aussi primordiale, ce simulateur doit pouvoir être facilement utilisable sur mobile.

En résumé, il faut une page permettant de :

1. Simuler une nouvelle adhésion en partant de zéro
2. Simuler une nouvelle adhésion en utilisant une structure déjà existante
3. Être utilisable sur téléphone

## 2.3 Les outils

Au cours du stage, j'ai utilisé et découvert de nombreux outils, que je connaissait et utilisait déjà comme Git et GitLab, ou de nouveaux comme Jira pour faire de la gestion de projet. Cependant les outils que j'ai le plus utilisé de loin, ce sont les documentations des langages et technologies utilisés.

### 2.3.1 Git

Git est certainement le software le plus important dans le monde de l'informatique avec Linux. C'est un gestionnaire de versions, c'est-à-dire qu'il permet de stocker un ensemble de fichiers, d'y apporter des modifications et de conserver ainsi l'historique de ces. Ici, je n'ai pas découvert Git, mais son utilisation dans le monde professionnel, la logique de "production" et de "développement" et donc le fonctionnement des branches. Avant ce stage, mon utilisation de Git était assez rudimentaire, je me contentais de la simple branche "main", puis ajouter mes modifications une fois que cela fonctionnait en local. Grâce à ce projet, mon utilisation a évolué, le dépôt contenant l'API et l'application possède deux branches majeures "main" et "develop", ces deux branches sont protégées. Afin de développer, je crée une nouvelle branche à partir de develop, puis ajoute mes modifications au fur et à mesure, et enfin crée une "merge request" afin que François LALAY puisse voir mes modifications et les accepter dans la branche develop. Puis une fois qu'il juge les avancées satisfaisantes, il merge la branche develop avec la branche main avant de passer au déploiement.

### 2.3.2 Gestion de Projet

Le développement de ce simulateur étant mon plus gros et surtout long projet à ce jour, je ne pouvais pas négliger la gestion du projet, même si je travaillais seul dessus. Lors de mon arrivée, j'ai réalisé un cahier des charges, qui devait présenter ma vision du projet, que j'ai présenté lors de ma première réunion hebdomadaire. Effectivement, deux fois par semaine, le lundi puis le jeudi, il y a des réunions avec quelques personnes du pôle numérique, où François LALAY présente ses avancées sur le projet Numérobis. J'ai donc aussi pris part à ces réunions sur ma période de présence, où je présentais donc les évolutions du simulateur. Lors de ces réunions, chacun peut donner ses impressions et proposer des changements ou ajouts, et donc afin de ne pas me perdre dans les demandes, j'ai utilisé Jira et son tableau Kanban\*, ici je place dans la colonne "TO DO" les tâches à développer, "IN PROGRESS" la tâche en cours et dans "DONE" les tâches terminées.

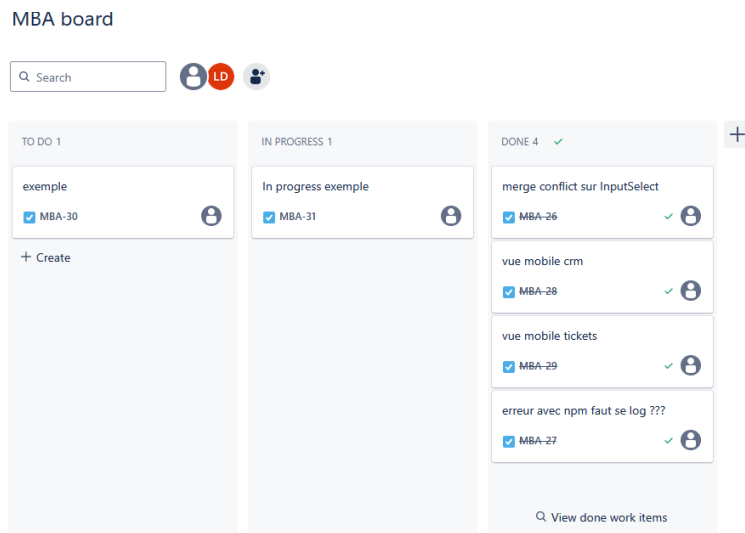


FIGURE 2.1 – Tableau Kanban de Jira

### 2.3.3 Les documentations et l'IA

Certainement le type d'outils que j'ai le plus utilisé, je les ai bien longtemps ignorés au profit d'outils plus "performants", notamment l'IA, mais lors de mon stage je me suis rendu compte de leur intérêt. Documentation d'API-Platform, Tailwind CSS, MDN, autant de documentation que de langages et technologies utilisés. Une fois que j'ai compris comment les utiliser, elles se sont avérées être des outils très puissants. J'ai eu, cependant, quelques soucis avec la documentation Symfony, dont je trouve que la recherche n'est pas très précise, et qui manque parfois d'explications et d'exemples. Ces différentes documentations sont à la base de mon projet, ont répondu à nombre de mes questionnements et recherches.

Un autre outil m'a aussi parfois aidé, Chat GPT, que j'ai beaucoup utilisé pendant le BUT, j'ai donc voulu faire l'effort de moins l'utiliser pendant le stage, ce que j'ai réussi. De plus, il a montré très vite ses limites, car s'il est effectivement très performant pour aider dans des petits projets universitaires, ici la masse conséquente de contexte le limite dans l'exactitude de ses réponses. Je l'ai donc utilisé dans deux types de cas, le premier, quand je ne savais absolument pas comment faire quelque chose, même pas l'idée de comment le faire, j'ai alors utilisé l'IA afin d'avoir un exemple pour commencer, le deuxième, quand j'avais besoin d'explications dont je ne trouvais pas la réponse dans la documentation.

## 2.4 Le simulateur

Le développement du simulateur s'est naturellement divisé en deux parties majeures, le front-end et le back-end. Pour la partie back-end, on m'a expliqué que le simulateur ne devait en aucun cas écrire en base de données, il devait se contenter de renvoyer le résultat. Il fallait aussi réutiliser les calculs déjà présents dans l'API, ce qui nécessite de comprendre ce dont ces calculs ont besoin comme paramètres. Pour le Front-end, pendant les réunions hebdomadaires, on a décidé que le simulateur serait une page à part entière du site, et que certaines informations étaient nécessaires sur la page, comme les champs pour renseigner les variables ainsi que les informations sur les différentes politiques ajoutées.

### 2.4.1 L'API Numérobis

Le back-end du site est une API Rest, développée avec Symfony et API Platform. Afin de développer le simulateur, j'ai dû comprendre la logique des entités Symfony, les relations entre elles ainsi que le fonctionnement des routes API Platform. Cette période de compréhension du projet a été grandement accélérée par le fait qu'au 3ème semestre j'avais réalisé un site en utilisant Symfony pour la SAE, j'étais donc déjà familier avec le fonctionnement de beaucoup d'aspects, comme les entités, les contrôleurs et Doctrine\* mais aussi plus globalement du langage PHP en lui-même.

Cependant, il fallait que je comprenne le fonctionnement d'API Platform, c'est un framework PHP basé sur Symfony, le site développé au 3ème semestre n'utilisait pas d'API, tout était fait dans le même projet, back et front. API Platform permet de créer des API REST avec Symfony. En créant les routes classiques automatiquement (GET, GET Collection, PUT, POST, DELETE), on peut aussi attacher des événements à ces routes comme "validate", ce qui va faire qu'un objet arrivant par cette route POST va être regardé afin de savoir s'il colle aux contraintes de la base de données. De plus, API Platform utilise le format Hydra JSON, ce qui permet de mieux comprendre ce qui est envoyé mais aussi de plus

simplement retrouver des objets avec l'IRI (Internationalized Resource Identifier), c'est le champ "@id" dans "hydra:member".

```
"@context": "/api/contexts/Adhesions",
"@id": "/api/adhesions",
"@type": "hydra:Collection",
"hydra:totalItems": 3781,
"hydra:member": [
  {
    "@id": "/api/adhesions/1776",
    "@type": "Adhesions",
```

FIGURE 2.2 – Exemple de hydra JSON

### 2.4.2 Le controller simulateur

Afin de réaliser le back-end du simulateur, je me suis grandement inspiré de la route de simulation déjà existante. En premier lieu j'ai analysé les paramètres nécessaires au calcul, puis j'ai créé des groupes de dénormalisation, c'est en fait une annotation qui permet de dire à l'API les données attendues pour la route POST. Ensuite, la fonction de calcul des prix a besoin de trois paramètres, une structure, une adhésion et une année, il faut donc que le json envoyé dans le body de la requête contienne les éléments nécessaires à la création de ces trois objets.

```
{
  "adhesion": {
    "structures": {
      "typesOfStructure": "/api/types_of_structure/4",
      "adhesionVariables": [
        {
          "value": 10,
          "years": "/api/years/2024",
          "priceUnit": "/api/price_units/2"
        },
        {
          "value": 1500,
          "years": "/api/years/2024",
          "priceUnit": "/api/price_units/1"
        },
        {
          "value": 7,
          "years": "/api/years/2025",
          "priceUnit": "/api/price_units/6"
        }
      ]
    },
    "formules": ["/api/formules/1"]
  },
  "years": 2025,
  "adheringRgpd": true
}
```

FIGURE 2.3 – Exemple de json pour le simulateur

On remarque donc ici que la Structure est imbriquée dans l'Adhésion, ces deux objets sont donc créés à la dénormalisation quand l'objet arrive avec la requête, on retrouve aussi dans ce json tous les champs du groupe

de dénormalisation, et qui suffisent donc à créer les deux objets. L'objet Année est un objet qui existe déjà en base de données, on transmet donc ici seulement l'année, qui va nous permettre, grâce aux repository de Symfony, de récupérer l'objet Years correspondant. Voici un graphique expliquant les principes de sérialisation et désérialisation.

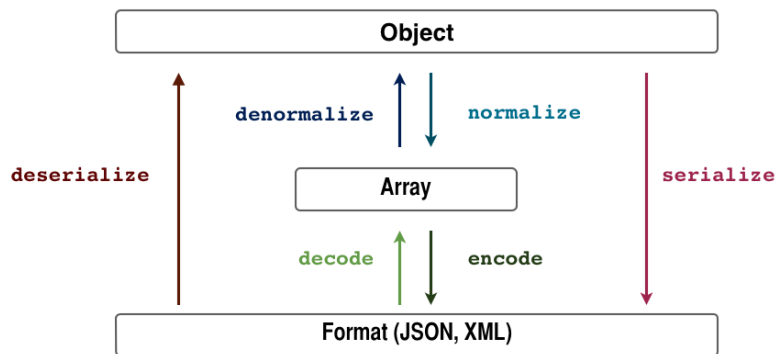


FIGURE 2.4 – La serialization

Ensuite, il y a deux exceptions majeures dans le calcul, la première n'est pas vraiment une exception mais une contrainte, la politique de maintenance de parc informatique multi-site est une augmentation de 25% du prix du volet numérique, il faut donc en plus de la formule multi-site, les formules du volet numérique, puis calculer leur prix, puis recalculer avec le multi-site afin d'obtenir le prix d'augmentation causé par le multi-site. La deuxième arrive dans le cadre de la politique "Parcours cybersécurité", si l'adhérent adhère déjà à la politique RGPD alors le prix de la politique est réduit de 30% supplémentaire, et donc pour gérer cette exception, le json possède un champ "adheringRgpd", qui donne l'information sur si cette structure adhère ou pas au RGPD, si c'est le cas, alors le controller ajoute la formule de RGPD avant le calcul, grâce au repository de Symfony, afin que la formule de calcul le prenne en compte. Afin de toujours avoir l'information sur l'ID des formules problématiques, leur ID est stocké dans un fichier .ENV, qui n'est pas sauvegardé sur Git, mais qui évite ces "nombres magiques", il y'a juste à changer l'ID dans le fichier .ENV et tous les endroits où c'est importé récupèrent alors le nouvel ID.

## Documentation

API Platform supporte nativement le format de documentation d'openAPI (anciennement Swagger), et donc cela crée une documentation à l'adresse [adresse-api/api](#). Sur cette page, on retrouve donc la documentation de toutes les routes de l'API, avec une documentation par défaut pour les routes standards, cependant ma route n'est pas standard, et donc pour la documenter correctement, j'ai utilisé des DTO (Data Transfer Object). J'ai donc créé deux DTO, un "input" (entrée) et "output" (sortie), grâce au groupe de dénormalisation et au DTO, la documentation

décrit parfaitement ce qui est nécessaire dans le body de la requête. Ces DTO décrivent donc ce qui est attendu en entrée et en sortie de la route.

```
{
  "adhesion": {
    "structures": {
      "name": "string",
      "typesOfStructure": "https://example.com/",
      "adhesionVariables": [
        {
          "value": 0,
          "years": "https://example.com/",
          "priceUnit": "https://example.com/"
        }
      ]
    },
    "formules": [
      "https://example.com/"
    ]
  },
  "years": 0,
  "adheringRgpd": {}
}
```

FIGURE 2.5 – Documentation de la route API

```
{
  "total": "string"
}
```

FIGURE 2.6 – Documentation de la route API

## Tests

Ayant écrit du code dans l'API, il faut bien évidemment le tester. J'ai donc créé une batterie de tests assurant une couverture de cas. Ce sont des tests fonctionnels, c'est-à-dire que l'on va tester les réponses de l'API. Dans les tests déjà écrits dans le projet, François LALAY utilise Zenstruck, qui permet de générer facilement des objets aléatoires pour le testing, mais aussi de réinitialiser la base de test entre chaque test, je l'ai donc aussi utilisé. J'ai donc testé mon contrôleur en lançant aussi la totalité des tests afin de vérifier si mes modifications n'ont pas cassé quelque chose autre part dans le code.

### 2.4.3 Le site Numérobis

Le site interne de l'ATD 16, Numérobis, est un ERP (Entreprise Resource Planning), le choix du développement d'un site interne a été conditionné par la complexité du système tarifaire de l'ATD.

## Fonctionnement de React

React fonctionne sur le modèle d'un arbre, chaque composant possède des enfants qui peuvent être d'autres composants React ou bien du JSX (c'est le code qui ressemble à du HTML dans le retour des composants).

Dans les composants rendus côté client, on peut utiliser des "hooks" react, ce sont des fonctions qui permettent de stocker des états ou encore d'utiliser le cycle de vie des composants. Par exemple "useState", ce hook permet de garder une valeur entre les différents cycles de vie, et donc chaque fois que cet état est mis à jour, le composant se rend à nouveau, ainsi que tous ses enfants. Afin d'optimiser les performances du site, il faut viser le minimum de rendu possible.

## **Le simulateur**

Le simulateur se découpe en cinq zones majeures : un header, la liste des adhésions, le détail de l'adhésion sélectionné, l'importation d'une structure ainsi que les variables d'adhésions.

Le header est plutôt simple, il contient le prix total des adhésions, simulé et importé s'il y a, ainsi que le sélecteur de type de structure, sélectionner un type de structure est nécessaire pour le calcul.

La liste des adhésions, c'est l'endroit où on affiche la liste qui contient les données envoyées à l'API. On retrouve dans cette liste toutes les données importantes relatives aux adhésions dans le simulateur : le titre de la politique, le prix simulé, ainsi que les formules. Le nom de chaque politique est cliquable pour afficher le détail des formules et pouvoir décocher ou cocher celles que l'on veut. Sur le côté gauche, on retrouve un groupe de boutons, le premier (ajouter), ouvre une modale qui permet de choisir une politique à ajouter, le deuxième supprime la politique courante et le dernier efface toute la liste. Les composants affichés sont inspirés de composants déjà développés par François LALAY, que j'ai modifiés afin de faire apparaître les informations voulues.



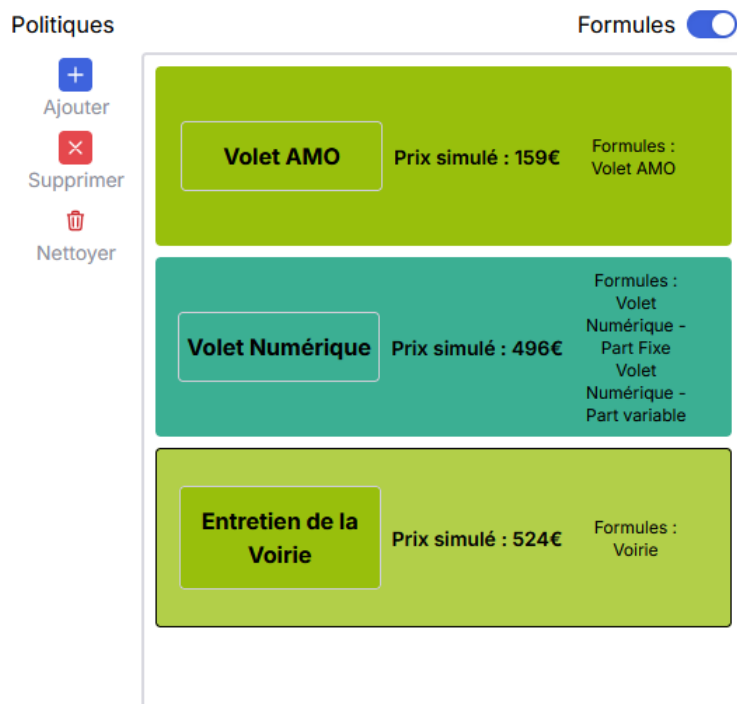


FIGURE 2.7 – Liste des adhésions

Le composant de détail consiste essentiellement en un groupe de checkbox pour choisir les formules. À l'origine, les formules étaient récupérées par un appel API, mais par souci d'optimisation, j'injecte désormais dans le composant les formules à afficher, ce qui réduit le nombre d'appels à l'API. Ce composant apparaît aussi dans une modale quand on sélectionne une nouvelle politique, ce qui permet de faire disparaître l'affichage classique de ce composant pour la vue mobile.

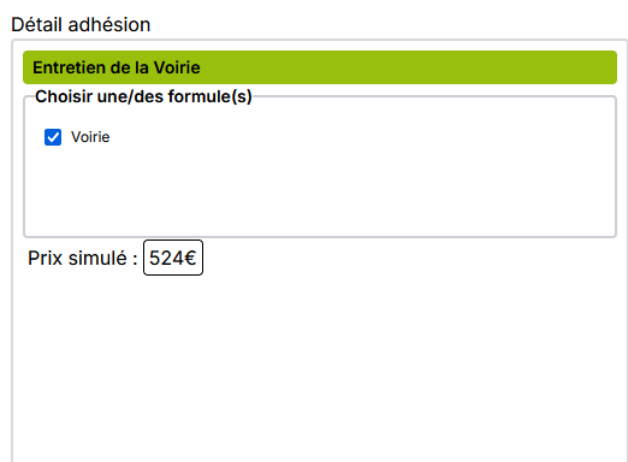
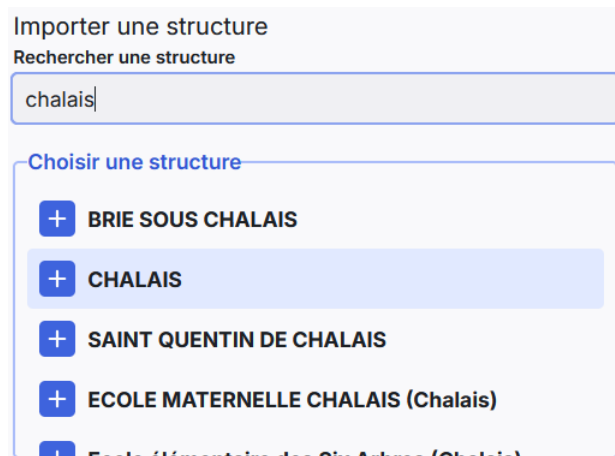


FIGURE 2.8 – Détail adhésion

Le composant d'import ne comporte visuellement qu'une seule chose, la barre de recherche de structure. Une fois qu'on sélectionne une struc-

ture, on récupère les données avec l'API, puis je les traite afin de correspondre au format nécessaire à l'affichage et au calcul de la simulation. Une fois la structure importée, on affiche ses adhésions ainsi que ses variables connues. De la même manière que pour le détail d'adhésion, ce composant disparaît en vue mobile et s'affiche dans une modale quand on clique sur le bouton "importer" (présenté plus tard avec la vue mobile).



Importer une structure

Rechercher une structure

chalais

Choisir une structure

- + BRIE SOUS CHALAIS
- + CHALAIS
- + SAINT QUENTIN DE CHALAIS
- + ECOLE MATERNELLE CHALAIS (Chalais)
- + Ecole (Municipale) des Saint-Amand (Chalais)

FIGURE 2.9 – L'importation de structures

Ce composant est une liste d'"input contrôlés", un input contrôlé en React, c'est un input où l'on va stocker sa valeur à chaque modification, pas de bouton de validation, ça permet de tout le temps connaître la valeur de l'input. Ici, c'est une liste d'input contrôle, c'est-à-dire qu'avec un seul "useState" on garde la trace de toutes les variables. La liste de variables est quant à elle récupérée par un appel API au chargement de la page.



Variables d'adhésions

Calculer

Habitants

0

Voirie

150

ETP

0

Utilisateurs

0

logiciels

0

Boîtes de messagerie

0

FIGURE 2.10 – Les variables d'adhésion

Comme expliqué précédemment, la vue mobile faisait partie des de-

mandes dès le début du développement. La question de la vue mobile tombe plutôt dans le domaine du CSS que du React, même si quelques changements ont été nécessaires de côté, comme la création des deux modalités de détail et d'import. Après avoir discuté avec mon maître de stage du design mobile, il m'a fait remarquer que dans la plupart des applications, notamment de scroll comme TikTok et Instagram réels, ont les actions placées sur le côté droit et en bas, les zones d'accessibilité du pouce.

Tailwind CSS possède un système très intéressant d'application conditionnelle des classes à certains paliers, par exemple avec le tag "md :" précédant une classe, celle-ci devient effective qu'à partir d'un affichage d'une largeur minimum de 768 pixels. Comme expliqué dans la documentation de Tailwind, ces paliers s'activent quand l'écran est supérieur à la largeur, il faut donc comprendre que l'affichage mobile doit être développé en premier, car on changera le comportement avec les marqueurs pour les écrans plus grands. Cependant, le site Numérobis n'a pas été pensé pour le mobile, et moi-même, m'y étant intéressé qu'une fois le simulateur fonctionnel, j'ai dû réécrire les classes Tailwind avec en configuration par défaut le mobile, et l'affichage bureau derrière des tags "md :". Ici, on a un contenant flex, avec une disposition en colonne pour la vue mobile (flex-col) puis en ligne si affichage plus large (md:flex-row).

```
<div className="m-1 flex h-full flex-col gap-2 p-1 md:flex-row">
```

FIGURE 2.11 – Exemple de responsive

J'ai donc transformé les boutons d'actions à gauche de la liste d'adhésions en une barre d'actions qui reste en bas de l'écran, et ajouté un bouton "importé" qui permet d'afficher une modale avec le composant d'import de structure. J'ai aussi inversé l'affichage des prix et du nom dans la liste des adhésions afin d'avoir la zone cliquable à droite.



FIGURE 2.12 – Vue mobile

## 2.5 Refactoring avec Zustand

Dans la première version du simulateur, j'ai utilisé les contextes de React pour pouvoir transmettre des infos en profondeur dans l'arbre React et aussi pour les modifier. Les contextes permettent d'éviter de faire des cascades de props\*.

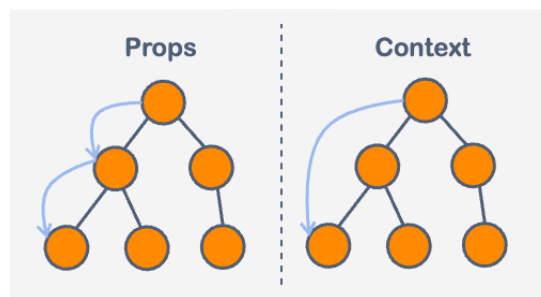


FIGURE 2.13 – Contextes React

Ils fonctionnent sur le principe suivant, dans un composant parent, on crée un contexte et un état avec "useState", puis dans le "return" du composant, on crée un "Provider" du contexte avec en valeur juste la valeur de l'état pour de la lecture seule, ou bien un objet avec la valeur de l'état ainsi que la fonction de modification si on souhaite le modifier plus bas dans l'arbre.

Dans le simulateur, j'ai eu besoin de transmettre beaucoup d'informations dans les composants, je me suis donc très vite retrouvé avec

beaucoup de contextes et de providers dans la section de ma page. Ce qui rend, d'une, le code très peu lisible, car tous les états sont créés dans ce composant et de deux un souci d'optimisation, comme expliqué précédemment, lorsqu'un état change, le composant se rend à nouveau ainsi que tous ses enfants, combiné ça avec plusieurs contextes qui peuvent être modifiés et on obtient une page qui se rend entièrement à chaque modification.

```
<VariablesAdhesionContextComponent adheVariables={adheVariables}>
  <PoliciesContextComponent>
    <FormulesContextComponent>
      <ResultsArrContextComponent>
        <SelectedIndexContextComponent>
          <StructureTypesContextComponent>
            <ImportedPricesContextComponent>
```

FIGURE 2.14 – Le "Provider Hell"

Dans un premier temps, j'avais déplacé la création des états et des contextes dans des composants à part, afin d'améliorer la lisibilité, mais cela ne règle pas les problèmes d'optimisation. Mon tuteur, François LALAY, m'a orienté vers une librairie qu'il utilise dans le site Numérobis, Zustand. Zustand permet de créer un "store", qui va stocker les données que l'on veut partager entre différents composants. Ce store est indépendant de React, les composants vont donc pouvoir interagir avec pour récupérer les infos nécessaires. De plus, grâce aux sélecteurs de Zustand, seulement sont mis à jour les composants qui utilisent une valeur du store qui a été modifiée, ce qui réduit grandement le nombre de rendus. On retrouve donc dans ce store les données stockées ainsi que les méthodes de modification, cependant, tout comme les états React, les valeurs sont immutables, c'est-à-dire qu'on va "remplacer" par la nouvelle valeur, plutôt que de modifier l'ancienne.

```
const initialState = {
  adhesions: [],
  adhesionVariables: [],
  selectedAdhesionIndex: 0,
  typesOfStructure: "",
  allFormules: null,
  allPolicies: null,
  typesOfStructureGroups: null,
  error: "",
  isStructureImported: false,
};

export const useSimulatorStore = createSelectors(
  create((set) => ({
    ...initialState,
    setAdhesions: (newAdhe) => set({ adhesions: newAdhe }),
    setAdhesionVariables: (newVariables) =>
      set({ adhesionVariables: newVariables }),
    setSelectedAdhesionIndex: (newIndex) =>
      set({ selectedAdhesionIndex: newIndex }),
    setTypesOfStructure: (newType) => set({ typesOfStructure: newType }),
    setAllFormules: (allFormu) => set({ allFormules: allFormu }),
    setAllPolicies: (allPol) => set({ allPolicies: allPol }),
    setTypesOfStructureGroups: (typesOfStrucGroups) =>
      set({ typesOfStructureGroups: typesOfStrucGroups }),
    resetSimulatorStore: () =>
      set({
        adhesions: [],
        selectedAdhesionIndex: 0,
        typesOfStructure: "",
      }),
    setError: (newErr) => set({ error: newErr }),
    setIsStructureImported: (newBool) => set({ isStructureImported: newBool }),
  })),
);
```

FIGURE 2.15 – Mon store Zustand

Le deuxième point important que m'a permis de faire Zustand, c'est la refactorisation de mon code. Du fait d'avoir déplacé les données de mon composant section au store, j'ai pu mieux isoler les logiques de chaque composant. Le fait de tout traiter dans la section m'avait contraint à y centraliser la logique, maintenant, la logique est séparée et donc plus lisible et compréhensible pour chaque composant. Par exemple, à la fin du développement de la première version, donc avec les contextes et sans aucun refactoring, le fichier de la section de la page a atteint environ 350 lignes de codes, maintenant ce fichier ne fait que 21 lignes, car plus aucune logique ne s'y trouve, j'y organise simplement la disposition de la page.

## 2.6 Autres petites missions

Ayant terminé ma mission principale plutôt rapidement, j'ai réalisé d'autres petites missions,

Chapitre 3

Conclusion