Rapport de stage
Stage de M1
Université d'Angers

Encadrant : M. Da Mota
Collaborateur : M. Cauchy

Yamis MANFALOTI
2024 - 2025

1. Introduction

Je tiens tout d'abord à remercier M. Da Mota pour m'avoir proposé ce sujet de stage dans un délai restreint, ainsi que pour son encadrement, sa disponibilité et la qualité de ses retours tout au long du stage. Je souhaite également remercier M. Cauchy pour ses nombreux retours et pour m'avoir vulgarisé clairement de nombreux concepts de chimie, ce qui m'a permis de mieux comprendre les enjeux du projet.

1.1. Contexte

Ce stage a été réalisé dans le cadre de ma première année de Master Informatique à l'Université d'Angers. Il s'est déroulé au sein du laboratoire LERIA (Laboratoire d'Études et de Recherche en Informatique d'Angers), du 7 avril au 13 juin 2025.

Il a été encadré par M. Da Mota et est en collaboration avec M. Cauchy, un chimiste du laboratoire MOLTECH-Anjou. Ce stage portait sur un sujet interdisciplinaire mêlant la chimie et le développement d'outils numériques à destination de chimistes.

1.2. Objectif du stage

L'objectif principal du stage était de rentrer en contact avec le chimiste afin de concevoir, puis de développer une application ergonomique, permettant la visualisation interactive de spectres scientifiques, prédits à partir d'une molécule dessinée. De plus, il était question de mettre en avant les liens entre les pics du spectre et les structures moléculaires correspondantes. L'application devait donc répondre à 3 besoins principaux : permettre aux chimistes de dessiner une molécule, de visualiser le spectre associé. Ainsi que de lire visuellement les liens entre les atomes de la molécule et les signaux du spectre.

De plus, une attention particulière devait être portée à l'ergonomie, à la fluidité de l'interface et à la qualité de l'expérience utilisateur. Cela est dû au fait que l'application finale est destinée à des chimistes, et que ceux-ci ne sont pas forcément adeptes des outils numériques.

En outre, si le temps le permettait, une extension du projet envisageait l'étude voire la mise en place de modèles de prédictions servant à obtenir le spectre à partir d'une structure moléculaire.

Au global, ce stage constituait également une opportunité pédagogique pour mettre en pratique une démarche complète de conception logicielle. Allant du recueil des besoins auprès de l'utilisateur final, des choix technologiques, de la conception de l'application, jusqu'au développement final. De plus, celui-ci permettait d'approfondir les aspects de communication avec un client (ici le chimiste) et d'appliquer des techniques de conception UML et C4[1], de prototypage et d'implémentation d'une interface ergonomique Web ou client lourd.

1.3. Contexte scientifique

Le projet de ce stage s'appuie sur des concepts issus du domaine de la chimie. Et ceux qui nous intéressent en particulier sont la représentation des molécules (SMILES[2]) et l'analyse spectroscopique (RMN[3]).

SMILES :

Le format SMILES (Simplified Molecular Input Line Entry System) est une manière textuelle de représenter une molécule chimique en encodant sa structure sous la forme d'une chaîne de caractères. Il s'agit d'une notation destinée à être lisible à la fois par l'humain et par des logiciels, tout en restant compacte.

La notation SMILES repose sur plusieurs conventions. La première étant que la représentation textuelle suit généralement la plus longue chaîne carbonée possible dans la molécule, à laquelle sont greffés les autres éléments chimiques. De plus, les atomes d'hydrogène ne sont pas explicitement notés, car ils peuvent être déduits implicitement. Cela est dû au fait que chaque atome a un nombre maximal de liaisons possibles (appelée valence), et en connaissant les autres atomes auxquels il est lié, on peut calculer combien d'hydrogènes sont nécessaires pour compléter sa structure. Enfin, certaines structures chimiques spécifiques comme les liaisons multiples, les cycles, et les branches sont représentées par des symboles ou des parenthèses, ce qui permet de décrire des structures chimiques complexes. On peut voir un exemple de molécule et de son SMILES au travers de la vanilline ($C_8H_8O_3$), un composé de la vanille, dans la **Figure 1** ci-dessous.

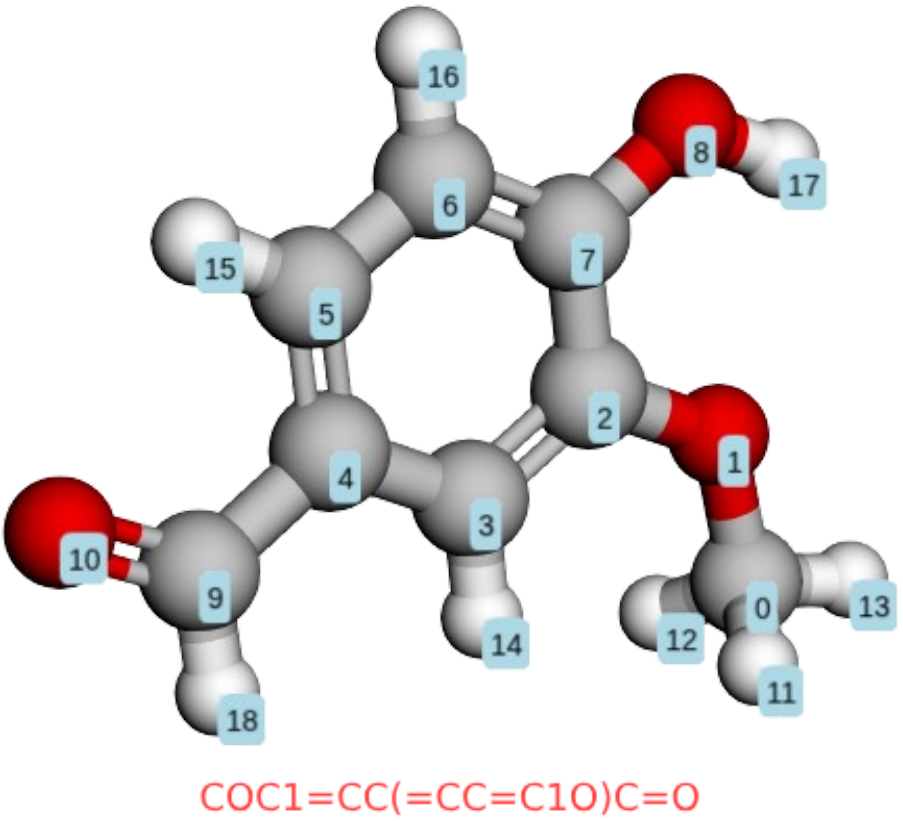

Figure 1 – Illustration de la vanilline et de son SMILES

Cependant, malgré sa simplicité apparente, le format SMILES laisse une certaine liberté d'interprétation. Une même molécule peut être décrite par plusieurs chaînes SMILES différentes, selon le point de départ choisi, l'ordre d'exploration des atomes ou la gestion des cycles. Cela implique que différents logiciels peuvent générer des versions légèrement différentes d'un même SMILES, notamment dans l'ordre des atomes ou la façon de fermer les cycles, même si ces représentations sont chimiquement équivalentes. Cela peut s'apparenter au domaine des graphes ou la molécule correspond ici a un graphe avec comme noeux les atomes, et pour les arêtes, les liaisons atomique. Et dans un tel exemple, les différentes versions des SMILES s'apparente au différentes façons de parcourir le graphe.

Pour pallier ces ambiguïtés, certains outils utilisent un format appelé Canonical SMILES, où un algorithme impose un ordre standardisé pour garantir une unique représentation d'une molécule donnée. D'autres outils appliquent également une kekulisation, c'est-à-dire une conversion de certaines structures (comme le benzène) en une forme avec des liaisons simples et doubles explicites, afin de représenter les cycles aromatiques de manière plus précise et de simplifier leur interprétation.

RMN :

La Résonance Magnétique Nucléaire (RMN), ou Nuclear Magnetic Resonance (NMR) en anglais, est une technique d'analyse très utilisée en chimie pour déterminer la structure des molécules. Elle repose sur l'interaction de certains noyaux atomiques avec un champ magnétique, ce qui permet d'obtenir un spectre RMN, un graphique riche en informations sur la structure d'une molécule.

Il existe différents types de RMN selon les noyaux observés. Les plus courants sont la RMN ^{1}H, qui s'intéresse aux noyaux d'hydrogène (protons). Ainsi que la RMN ^{13}C, qui analyse les noyaux de carbone 13 (un isotope du carbone).

Ce qu'on appelle un spectre RMN, est une courbe où l'on retrouve généralement deux axes. Le premier est l'axe horizontal (abscisse) et est gradué en ppm (parties par million), indiquant le déplacement chimique. Et ce déplacement reflète l'environnement électronique du noyau étudié, c'est-à-dire comment les électrons autour de cet atome influencent sa réponse au champ magnétique. Le second, correspond à l'axe vertical (ordonnée) qui représente l'intensité du signal, qui est lui-même proportionnelle au nombre de noyaux identiques dans un même environnement chimique. Par exemple, si plusieurs atomes d'hydrogène se trouvent dans une même configuration, ils produiront un pic plus intense. Un exemple d'un tel spectre est donné plus bas dans la **Figure 2**.

Une autre chose à savoir est que les pics du spectre ne sont pas placés au hasard. Ils suivent des motifs caractéristiques dépendant de la structure de la molécule. Par exemple, dans le cas de la vanilline, on retrouve des signaux caractéristiques des différents groupes fonctionnels présents dans la molécule, comme le groupe aldéhyde (-CHO) ou encore le groupement méthoxy ($-OCH_3$). Chacun de ces groupes donne un ou plusieurs pics situés à des positions spécifiques sur l'axe des ppm, ce qui permet de les identifier. La forme et le nombre de ces pics peuvent également être influencés par les interactions entre noyaux voisins.

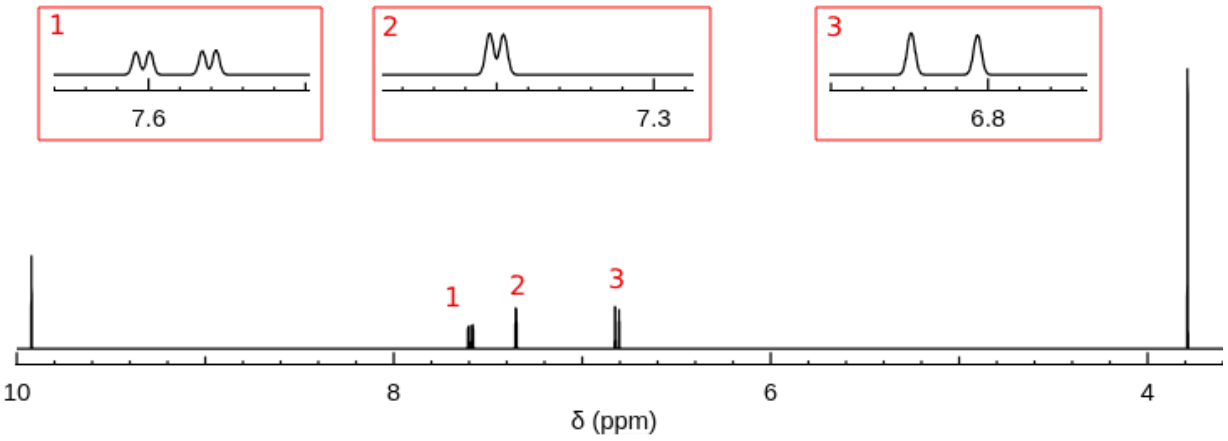

Figure 2 – Spectre RMN 1H de la vanilline
(source : prediction de nmrdb.org + mise en page)

L'intérêt principal de la RMN expérimentale est d'obtenir un spectre précis à partir d'un échantillon réel. Ce spectre fournit des informations essentielles pour identifier la composition chimique, déduire la structure moléculaire, et analyser des substances, y compris inconnues. Grâce à la RMN, les chimistes peuvent étudier les interactions atomiques dans une molécule directement issue d'un matériau analysé en laboratoire. En reprenant l'exemple des graphes vu précédemment, on peut assimiler les différents pics du spectre RMN à des sortes de sous-graphes. Et l'analyse de ces différents sous-graphes permet de reconstruire la structure complète du graphe, ce qui correspond, dans notre cas, à la structure de la molécule

En parallèle, il est possible de simuler un spectre RMN à partir d'une structure moléculaire connue grâce à des outils informatiques. Cette simulation permet de prédire le spectre attendu, ce qui aide à interpréter les signaux expérimentaux et à vérifier la cohérence entre la structure théorique et les données réelles. La simulation joue un rôle clé dans le développement d'outils numériques pour assister les chimistes dans l'analyse spectrale, car elle est beaucoup plus rapide que la RMN expérimentale, qui peut parfois prendre plusieurs jours.

2. Organisation du stage

Ce stage s'est structuré autour de deux grandes phases. La première étant la partie correspondant au développement de l'outil (frontend, backend, interactions, packaging, documentation), et qui était la partie principale du projet. Tandis que la seconde correspondait à l'étude des modèles de prédiction, une phase prévue en fin de stage, comme une extension de celui-ci si le temps le permettait.

Dès le début du stage, j'ai commencé par établir un planning détaillé couvrant l'ensemble des semaines à venir, et découpé en phases correspondant aux différents objectifs. Vous trouverez ce planning et la répartition des tâches au travers du tableau en **Figure 3**. Cela m'aura permis de structurer ma progression semaine par semaine, voire jour par jour. D'ajuster les priorités selon les retours client et la réalité du développement. Ainsi que de planifier le déroulement à l'avance afin d'avoir une vision claire de la composition du projet, et donc de faciliter sa mise en place par la suite.

De plus, j'ai tenu ce planning à jour quotidiennement, en marquant les tâches effectuées et en réajustant les tâches à compléter si nécessaire. Cette approche m'a permis d'avoir une vision claire de l'avancement du projet, tout en garantissant la flexibilité nécessaire à un développement itératif. Cette rigueur dans la planification m'a permis de maintenir un bon rythme de développement tout en documentant précisément chaque étape du projet.

Phase	Période	Objectif
Phase 1	Jours 1 à 8	Besoin et conception (reformulation des besoins, étude de l'existant, architecture, UX, formats de données, maquettes)
Phase 2	Jours 9 à 18	Développement du frontend (structure générale, intégration de Ketcher[6] et de Plotly[7], menu de configuration, fonctionnalités js)
Phase 3	Jours 19 à 26	Développement du backend et des logiques d'interaction avancée
Phase 4	Jours 27 à 29	Finitions et déploiement (packaging, refactoring, tests multi-OS, documentation technique et utilisateur)
Phase 5	Jours 30 à 33	Rédaction du rapport et préparation à la soutenance sur la partie élaboration de l'application
Phase 6	Jours 34 à 40	Étude du modèle de prédiction, Finalisation du rapport
Phase 7	Jours 41 à 44	Phase finale, Préparation soutenance

Figure 3 – Planification et répartition *des tâches*

3. Analyse du besoin

3.1. Contexte et objectifs du projet

Le projet visait à développer une application destinée aux chimistes permettant de dessiner des molécules, de prédire leurs spectres RMN, puis de visualiser ces spectres de façon interactive.

L'objectif principal était de faciliter le processus de dessin d'une molécule, de prédiction du spectre ainsi que de l'analyse des résultats. Et pour cela, il était question de faire une interface ergonomique et pratique afin de permettre une bonne prise en main pour des chimistes, pas forcément adeptes du numérique. De plus, l'analyse des résultats devait être facilitée grâce à une interface qui relie directement la structure moléculaire et les parties du spectre RMN correspondantes.

La mission consistait à créer une application hybride combinant un dessin moléculaire et un affichage de spectres RMN prédits, avec une interaction dynamique entre les deux. La solution devait aussi intégrer un prédicteur RMN (factice dans un premier temps) prenant en entrée une molécule au format SMILES et produisant un spectre RMN simulé.

Les enjeux de cette application, consiste à offrir un outil pédagogique et scientifique simplifiant la prédiction des spectres, ainsi que leur compréhension et leur interprétation. Il s'agit également de permettre aux chimistes d'identifier et d'analyser rapidement les relations entre les atomes d'une molécule et les signaux RMN correspondants. Enfin, un grand enjeu est de réduire le temps d'analyse en automatisant la génération et la visualisation des spectres à partir de la structure moléculaire dessinée.

3.2. Recueil du besoin avec les utilisateurs

Une collaboration étroite avec le chimiste a permis de définir les fonctionnalités clés, via des échanges par messages ou des discussions vocales sur Discord. Ses retours ont orienté le choix des fonctionnalités et des outils, ainsi que les contraintes ergonomiques spécifiques au domaine.

Dans un premier temps, les fonctionnalités principales qui ont été identifiées étaient les trois suivantes : Tout d'abord, la possibilité d'entrer une molécule au format SMILES dans un prédicteur (considéré comme une boîte noire) qui générerait un spectre RMN simulé. La fonctionnalité suivante correspondait à l'affichage simultané du dessin moléculaire et du spectre RMN, avec des interactions entre les deux. Telle qu'un clic sur un pic qui mettrait en brillance les atomes correspondants. Et enfin, la dernière fonctionnalité, était de pouvoir dessiner une molécule via un éditeur intégré et obtenir son SMILES. Celui-ci étant utilisable par le prédicteur.

De plus, il y avait plusieurs contraintes ergonomiques et scientifiques à respecter afin de produire une solution adaptée aux chimistes : La première contrainte était de produire une interface claire et fluide, adaptée aux usages scientifiques, et facile à prendre en main. Puis il était demandé une séparation nette entre la zone de dessin et d'affichage de spectre. Des fonctionnalités interactives facilitant l'analyse du spectre (surbrillance, zoom, labels). Des performances suffisantes pour manipuler des molécules ou des spectres complexes. Une application totalement open source pour un usage à l'université. Ainsi qu'une compatibilité multiplateforme.

3.3. Étude des solutions existantes

Avant de concevoir une application sur mesure, une analyse des solutions déjà existantes dans le domaine de la prédiction et de la visualisation de spectres RMN a été menée. Deux outils se sont démarqués : Mnova NMR Predict[4] ainsi que NMRium[5].

Pour Mnova NMR Predict[4] de Mestrelab, il s'agit d'un logiciel propriétaire très complet, permettant à la fois le dessin de la molécule, la prédiction RMN à partir de la structure chimique, ou encore l'affichage et l'interprétation des spectres (multiplicité, couplages, attribution des signaux). Cependant, ce logiciel est performant, mais propriétaire et payant, ce qui ne correspond pas aux contraintes de notre projet (open source, à usage académique).

Quant à NMRium[5], il s'agit d'un visualiseur open source dédié à la prédiction et à l'analyse de spectres RMN. Il propose une interface web moderne permettant de charger, afficher, annoter et analyser des spectres 1D et 2D. Il intègre désormais un éditeur moléculaire natif, permettant de dessiner ou d'importer une molécule, puis de prédire automatiquement les spectres RMN correspondants (^{1}H, ^{13}C, COSY, HSQC, HMBC). Bien que centré sur l'analyse spectroscopique, il offre ainsi une solution complète allant de la structure chimique à l'interprétation des données RMN. Le logiciel est librement accessible, et des services professionnels (hébergement, support, intégration ELN) sont proposés par l'entreprise Zakodium.

En résumé, NMR Predict[4] est une solution complète, mais propriétaire et payante, ce qui ne correspond pas aux exigences open source et académiques de notre projet. Tandis que NMRium[5] est un outil open source proposant un visualiseur RMN avec éditeur moléculaire intégré et prédiction spectrale. Mais celui-ci reste principalement centré sur l'analyse spectroscopique et son modèle économique inclut également des services payants, limitant sa flexibilité pour une personnalisation complète.

Ces constats justifient le développement d'une solution sur mesure, tirant parti des fonctionnalités clés de ces outils tout en garantissant une modularité, une ouverture et une adaptation aux besoins spécifiques de notre projet.

Par la suite, afin de concevoir notre propre application, les solutions qui existaient déjà, ainsi que celles qui correspondaient à nos exigences, ont été étudiées. Soit dans le cadre de leur intégration à l'application finale, soit simplement pour les fonctionnalités qu'ils proposent.

Parmi ces solutions, on peut les classer en 3 groupes. Le premier étant les solutions qui permettent de gérer le dessin des structures moléculaires. Le second correspond aux solutions qui nous permettraient d'afficher et d'analyser les spectres RMN prédit. Et le dernier groupe correspondant au système d'affichage qui nous servira à faire le reste de notre application.

Nous avons donc d'abord analysé les solutions qui permettent à l'utilisateur de dessiner une molécule via une interface visuelle, et d'extraire des formats comme SMILES. Cette analyse est synthétisée dans le tableau ci-dessous en **Figure 4**.

Outil	Technologie	Licence	Avantages / Limitations
JSME Editor	JavaScript	BSD	🟩 Léger, simple à intégrer dans une page web 🟧 Système de dessin plus primaire
Ketcher[6]	JavaScript	Apache 2	🟩 Moderne, ergonomique, supporte de nombreux formats, API JS puissante 🟧 Plus lourd car plus complet
RDKit[8]	Python	BSD	🟩 Très léger et efficace 🟧 Permet seulement l'affichage des molécules

Figure 4 – Tableau comparatif des solutions pour l'édition de structure moléculaire

De même pour les solutions qui permettent d'afficher des graphiques correspondant au spectre RMN et qui sont présentés dans le tableau en **Figure 5**.

Outil	Technologie	Licence	Commentaire
Plotly.js / Plotly.py[7]	JS / Python	MIT	🟩 Super pour les graphes interactifs, disponible en JS et Python avec bonne intégration, et supporté par une large communauté bien documentée 🟧 Un peu plus lent sur de grand ensemble de donnée
D3.js	JavaScript	BSD	🟩 Grande flexibilité et contrôle total sur les graphes, très performant sur des données complexes et volumineuses, avec une grande communauté active 🟧 Plutôt bas niveau, et courbe d'apprentissage plus élevé
Matplotlib	Python	Python Software Foundation License	🟩 Très facile à prendre en main, bien documenté, et performant sur des données simples. 🟧 Graphe statique et peu interactif

Figure 5 – Tableau comparatif des solutions pour l'affichage de spectres

Enfin, c'est au tour des frameworks et bibliothèques qui servent à construire le reste de l'interface (menus, barres d'outils, panneaux interactifs, etc.). Le tableau en **Figure 6** propose de nouveaux une vue d'ensemble de ces solutions, en soulignant les points clés à considérer pour le choix des technologie.

Outil	Technologie	Licence	Commentaire
React / Vue / Svelte	JavaScript	MIT	🟩 Moderne, très populaire pour créer des UI web,large communauté, excellent support d'intégration avec d'autres outils. 🟧 Peut nécessiter un temps d'apprentissage pour les débutants.
Tkinter	Python	Libre	🟩 Facile à utiliser et livré avec Python par défaut. 🟧 Pas adapté pour les applications web. Et a une interface minimale, basique et peu moderne.
PyQt	Python	GPL ou commercial	🟩 Interface très riche et puissante, documentation complète et un bon support communautaire. 🟧 Licence GPL ou commerciale, ce qui peut poser un problème.
Kivy	Python	MIT	🟩 Idéal pour applications mobiles et desktop multi plateformes, bon support pour interfaces tactiles et réactives. 🟧 Moins populaire et peut nécessiter des adaptations pour des interfaces plus classiques.
PyWebView	Python + HTML	BSD	🟩 Permet une interface web dans une application desktop Python, léger et simple à utiliser. 🟧 Moins performant pour des applications très interactives ou complexes.

Figure 6 – Tableau comparatif des solutions pour construire l'interface

4. Conception de l'application

4.1. Méthodologie de conception

La conception du projet n'a pas suivi un plan linéaire ou fixé dès le départ. Une approche itérative a été privilégiée, en tenant compte des retours réguliers du chimiste tout au long de l'avancement du projet.

À chaque itération, les retours permettaient d'ajuster ou de redéfinir certains besoins fonctionnels ou techniques. Ensuite, une nouvelle phase de conception était engagée avant de passer au développement. Et cette phase de conception se basait sur de la documentation et des outils de modélisation pour garantir une structure claire et cohérente.

Parmi les outils de modélisation qui ont été utilisés, on retrouve : Les diagrammes de modélisation C4[1] pour représenter l'architecture logicielle à différents niveaux d'abstraction (contexte, conteneurs, composants). Ainsi que les diagrammes UML, notamment avec les diagrammes de composants pour clarifier l'organisation logicielle. Ainsi que les diagrammes de séquence pour modéliser les interactions entre les différents composants.

Cette méthodologie de travail a permis de maintenir une conception toujours alignée avec les besoins réels, tout en garantissant une bonne structuration technique du projet.

4.2. Architecture générale

L'architecture logicielle qui a été choisie repose sur une structure modulaire de type client-serveur en local. Elle est organisée autour d'une API REST facilitant la communication entre le frontend (interface utilisateur) et le backend (traitement et prédiction).

Et ce découpage est directement issu de l'analyse des besoins, afin d'avoir une interface intuitive pour le dessin de molécules, des traitements de données chimiques, ainsi que la génération et l'affichage de spectres RMN, tout en permettant une évolutivité du système.

Pour le backend, il a été choisi d'utiliser Python avec Flask[9] pour l'API, et de le combiner à des bibliothèques scientifiques telles que RDKit[8] ou SciPY. Python est idéal pour la manipulation scientifique, notamment avec RDKit[8] pour le traitement des structures chimiques. Tandis que Flask[9], est léger et simple, et permet de créer une API rapide à mettre en place, adaptée à un projet sans logique serveur complexe.

L'architecture du backend est représentée sous forme d'un diagramme C4[1] dans la **Figure 7**, qui illustre les composants du container backend et leurs interactions.

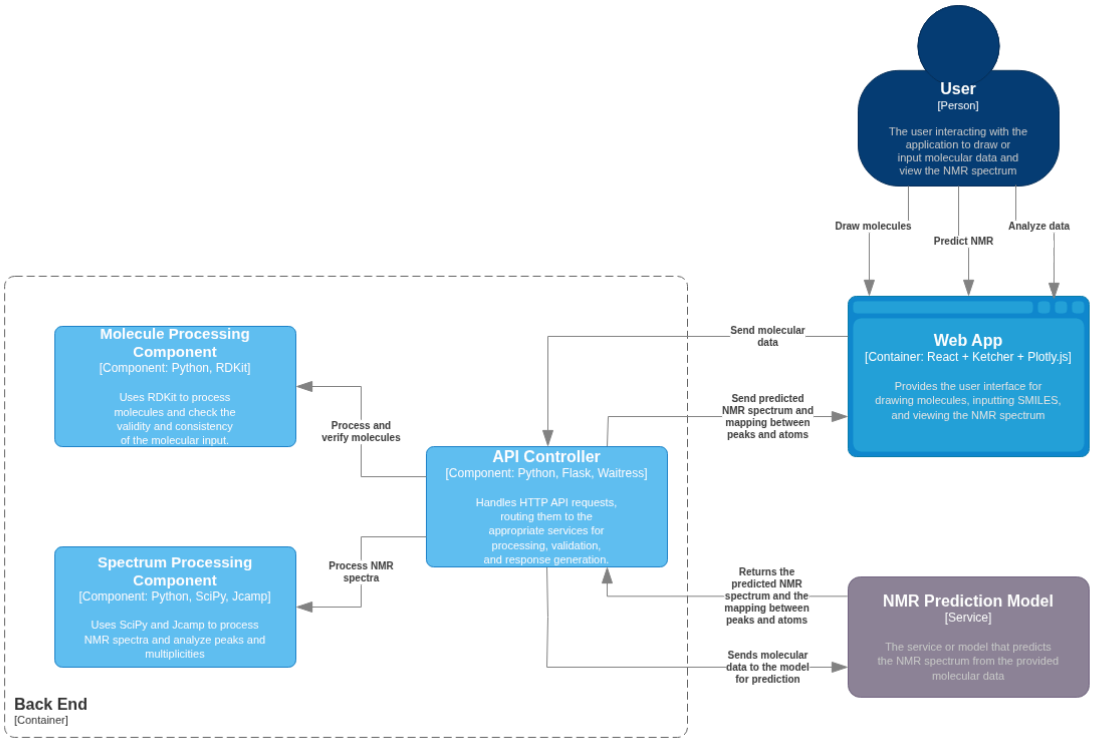

Figure 7 – Diagramme C4 pour le conteneur Backend

Pour le Frontend, c'est React qui a été utilisé. Avec, on retrouve Ketcher[6] pour l'éditeur de structure moléculaire, ainsi que Plotly.js[7] pour l'affichage de spectres. Ces choix ont été faits car React offre une interface moderne et réactive, en plus d'être facilement extensible. Ketcher[6] est une solution open source très complète et permet de dessiner des molécules et de les exporter en SMILES, ce qui est essentiel pour notre application. Et pour Plotly.js[7], il permet l'affichage interactif des spectres RMN, avec des fonctionnalités telles que le zoom, la détection de survol du spectre, etc.

Cette architecture, simple mais modulaire, garantit une bonne intégration des outils scientifiques, une interface fluide pour l'utilisateur et une évolutivité du système à long terme. Elle est illustrée dans le diagramme de modélisation C4[1] du conteneur Web App, présenté en **Figure 8**.

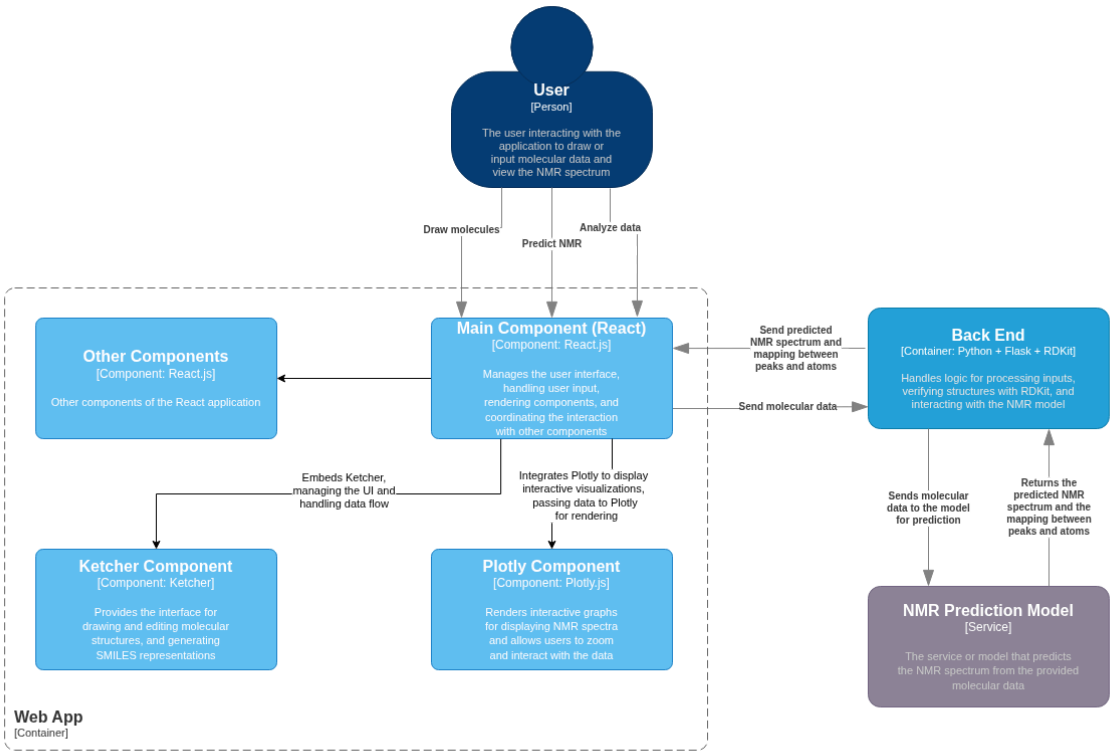

Figure 8 – Diagramme C4 pour le conteneur Web App (Frontend)

4.3. Conception UI / UX

L'interface de l'application a été pensée pour répondre aux besoins spécifiques des chimistes en matière de prédiction et d'analyse de spectres RMN. Une interface à deux panneaux latéraux (molécule à gauche, spectre à droite) a été privilégiée, avec une barre d'outils en haut regroupant les actions principales (prédiction, chargement, export, paramètres, aide, onglet, etc.).

Chaque élément a été conçu pour offrir une expérience fluide et interactive : zoom sur les spectres, interaction directe entre molécules et pics, minimisation et redimensionnement des panneaux.

La disposition de l'interface, ainsi que les différents éléments interactifs, ont été représentés sous la forme d'une maquette afin de visualiser concrètement l'organisation générale de l'application. Cette même maquette est illustrée au travers de la **Figure 9**.

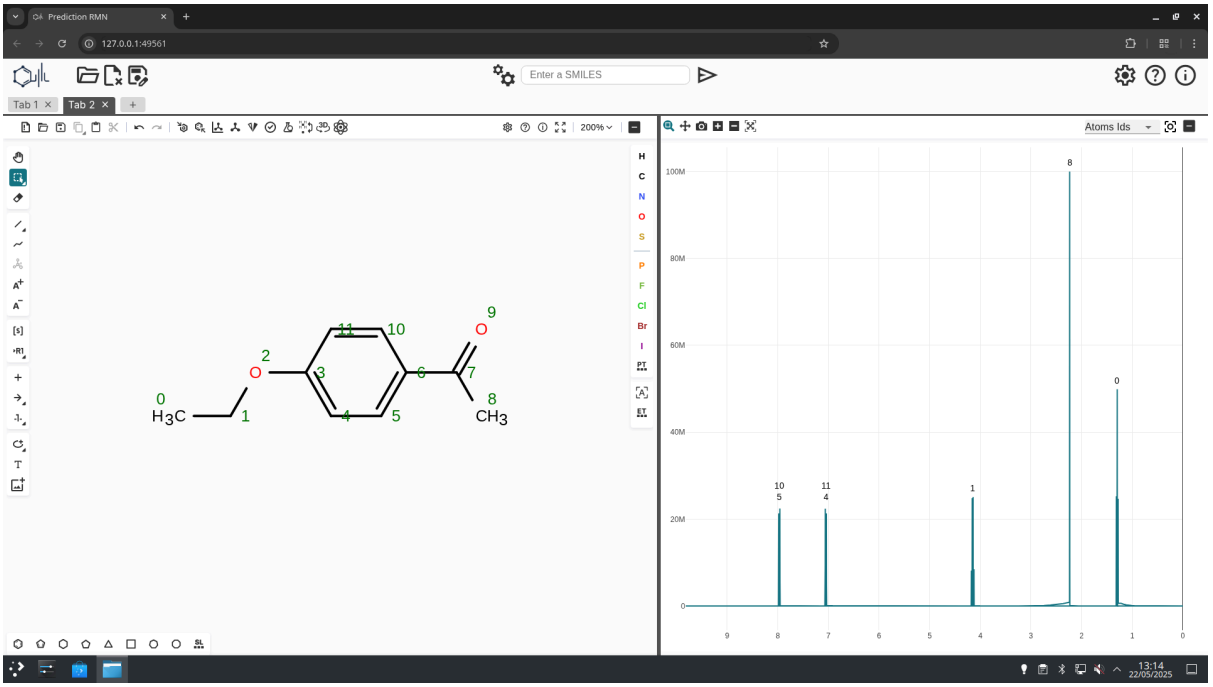

Figure 9 – Maquette de l'interface de l'application

En plus des maquettes, plusieurs parcours utilisateurs ont été modélisés et validés par le chimiste. Ceux-ci ont permis de définir l'enchaînement d'actions que doit effectuer l'utilisateur final afin d'arriver à ses objectifs. Et parmi les parcours utilisateurs les plus importants dans notre application, on retrouve :

- **Dessiner une molécule**
  Ouvrir l'application > Accéder au panneau gauche (Ketcher) > Ajouter/modifier des atomes et des liaisons

- **Prédire un spectre à partir d'une molécule dessinée**
  Dessiner la molécule dans le panneau gauche (Ketcher) > Laisser le champ de SMILES vide > Cliquer sur "Lancer la prédiction" > Attendre la réponse de l'API > Affichage automatique du spectre

- **Prédire un spectre à partir d'un SMILES texte**
  Accéder au champ de SMILES > Saisir ou coller un SMILES > Lancer la prédiction > Attendre la réponse de l'API > Affichage du spectre

- **Observer un spectre RMN**
  Lancer une prédiction (ou charger un spectre) > Affichage du spectre dans le panneau droit (Plotly) > Interagir avec le graphique (zoom, survol, déplacement, annotation)

- **Mettre en surbrillance les atomes associés à un pic**
  Cliquer sur un pic dans le spectre > Les atomes liés sont mis en surbrillance dans la molécule

- **Enregistrer un projet**
  Cliquer sur le bouton “Exporter le projet” > Sélectionner le format JSON > Génération du fichier avec la molécule, le spectre et le mapping atomes–pics

- **Charger un projet**
  Cliquer sur le bouton “Charger le projet” > Choisir le fichier “.json” (projet complet) ou “.jdx” (spectre seulement) > Chargement du fichier contenant la molécule, le spectre et le mapping atomes–pics

Côté design, une charte graphique claire et sobre a été définie pour favoriser la lisibilité et l’ergonomie. En couleurs principales on retrouve : le blanc , le bleu canard foncé (#167782), le gris foncée (#525252), ainsi que le gris claire (#D9D9D9). Pour la typographie, c’est la police “Inter”, qui à été choisie pour sa modernité et sa lisibilité. Pour les différentes icônes de l’application, Google Material Symbols a été retenue pour sa cohérence, ainsi que la facilité avec laquelle on reconnaît la signification de chaque icônes. Et pour le design global, c’est un style épuré qui a été choisi et le style graphique est inspiré de l’éditeur Ketcher[6] afin d’assurer une intégration visuelle harmonieuse.

Enfin, l’interface a été conçue selon une approche UX centrée sur l'utilisateur, en collaboration avec le chimiste, et ajustée selon les retours. L’objectif a toujours été de fournir un outil intuitif, fonctionnel et agréable à utiliser, facilitant la manipulation pour des chimistes pas forcément adeptes des outils numériques.

4.4. Itérations et ajustements

Le projet a été mené de façon itérative afin de rester flexible et réactif face aux retours du chimiste. À la fin de chaque cycle, un prototype était livré, testé, puis ajusté en fonction des remarques reçues. Cette méthode a permis d’améliorer progressivement l’ergonomie et la pertinence des fonctionnalités.
Par ailleurs, plusieurs fonctionnalités non prévues initialement ont été ajoutées au fil du développement, en réponse aux besoins exprimés par l’utilisateur ou identifiés en cours de test. Une communication continue avec le chimiste a renforcé cette dynamique d'amélioration.

02/05/2025 – Prototype initial (Itération 1)

- Interface statique testable localement.
- Chargement / enregistrement de projets .json (molécule + spectre)
- Export en JSON ou image du spectre
- Simulation de prédiction à partir d’un SMILES ou dessin
- Affichage interactif du spectre (zoom, survol, clic → atomes surlignés)
- Panneaux redimensionnables / minimisables
- Premiers réglages accessibles (modèle, aide, paramètres généraux)

09/05/2025 – Intégration backend & UX (Itération 2)

- Connexion au backend pour la gestion des paramètres locaux
- Conversion au format kékulisée des SMILES via RDKit[8]
- Overlay Snackbar pour les retours utilisateur
- Sélecteur de modèle de prédiction (dans les paramètres de prédiction)
- Ajout d'un modèle de prédiction factice
- Effacer le projet en cours
- Début du système d'onglets

19/05/2025 – Version fonctionnelle complète (Itération 3)

- Fin du système d'onglet
- Gestion de l'attente en arrière-plan des résultats de la prédiction
- Ajout du menu de configuration des modèles de prédiction
- Bouton pour créer un nouvel onglet directement dans la barre d'onglets
- Export au format ZIP (JSON + spectre + molécule numérotée via RDKit[8])
- Sélecteur d'annotations sur le spectre (aucun / IDs atomes / ppm des multiplicité)
- Prise en charge des fichiers JCAMPDX (.jdx)
- Détection de pics améliorée avec SciPy[10] (find_peaks[12]) et gestion des multiplicités
- Affichage / masquage des IDs d'atomes via un bouton dans Ketcher[6]
- Confirmation à la fermeture d'un onglet qui contient des données

4.5. Conception détaillée

La conception détaillée a permis d'anticiper l'organisation technique du projet, en définissant les structures internes, les flux de données, et les interactions entre les composants. Bien qu'une base ait été posée en amont, cette conception a également évolué tout au long des itérations, en réponse aux retours du chimiste et aux ajustements fonctionnels.

Un découpage clair avait été initialement prévu. Avec le frontend (React + Ketcher[6] + Plotly.js[7]) qui correspond à l'interface utilisateur interactive, la gestion des états visuelle, ainsi que l'interactions molécule–spectre. Pour le backend (Flask[9] + bibliothèques scientifiques), c'est une API REST qui gère le traitement des SMILES, la prédiction, les conversions chimiques, ainsi que le reste de la communication entre les différentes parties de l'application. Enfin, pour le modèle de prédiction, celui-ci est géré tel service externe ou local qui retourne un spectre simulé à partir d'une structure moléculaire.

Au sein de cette conception détaillée, plusieurs points clés ont été étudiés et détaillés afin de fournir une structure claire. On retrouve parmi eux :

- Un contrat d'API interne clair (/predict, /convertToKekuleSmiles, /detectSpectrumRegions, etc.) pour assurer une communication structurée entre le frontend et le backend.

- Des modèles de données JSON structurés, prévoyant les paramètres, les liens entre atomes et pics, les multiplicités, et les informations nécessaires aux interactions.

- Des comportements UX définis à l'avance sous forme de scénarios (prédiction, interaction, sauvegarde), parfois formalisés en diagrammes de séquence.

- Les adaptations prévues de Ketcher[6] (affichage d'IDs d'atomes, intégration de contrôles externes, compatibilité avec les flux de l'app).

- Ainsi qu'une organisation modulaire du frontend, pensée pour être maintenable et extensible (panneaux latéraux, gestion d'onglets, overlays, etc.).

Au global, cette phase a servi de fondation technique pour le développement, tout en restant flexible afin de suivre les changements. À chaque cycle d'itération, certains éléments (API, formats de données, UI, etc) ont été ajustés pour mieux répondre aux usages concrets observés pendant les tests.

5. Développement de l'application

5.1. Mise en place du projet

L'organisation du projet pour la partie développement de l'application s'est appuyée sur deux **dépôts Git** distincts, chacun dédié à une responsabilité claire du système.

Le premier dépôt se nomme CustomKetcher[6] et est un fork de la version stable v3.1.0 de Ketcher[6]. Ce dépôt a été utilisé pour intégrer et personnaliser l'éditeur moléculaire, et contient deux branche principles :

- Une branche dédiée aux modifications du cœur de Ketcher[6] (composants React, intégration dans une structure modulaire, amélioration de l'UI, ajout de fonctionnalités spécifiques comme le contrôle externe et l'affichage des IDs).

- Une branche de développement frontend dans laquelle les modifications précédentes sont intégrées via des merges, et qui héberge le développement de l'interface utilisateur finale de l'application. Cette interface est ensuite compilée en fichiers statiques pour être intégrée à l'application finale.

Tandis que le deuxième dépôt, appelé PredictionRMN, centralise tous les composants backend et les éléments nécessaires à la distribution finale de l'application. Il contient donc : Le serveur Flask[9] et l'API interne. L'intégration des fichiers statiques générés à partir du frontend. Les scripts de packaging (PyInstaller[11]) pour générer un exécutable autonome (Windows, Linux, etc.) Ainsi que la documentation technique, utilisateur et les fichiers de configuration.

Cette séparation en deux dépôts à permis une meilleure modularité, facilitant l'intégration à venir des mises à jour indépendantes de Ketcher[6]. Ainsi que l'évolution de l'application sans empiéter sur les bases du package principal.

5.2. Méthodologie de développement

Comme vu durant la partie conception, le développement s'est déroulé de façon itérative, avec une intégration progressive des fonctionnalités. Chaque ajout était testé manuellement, validé en conditions réelles et ajusté en fonction des retours, notamment lors des échanges réguliers avec le chimiste.

Afin d'assurer la qualité et la cohérence du code, plusieurs outils de développement ont été utilisés dès l'initialisation du projet :

- Prettier et ESLint ont permis de standardiser le style de code et de repérer automatiquement les erreurs courantes.
- Le typage strict TypeScript a été activé pour mieux détecter les incohérences à la compilation.
- Le projet a été structuré à partir de l'environnement de développement du dépôt original de Ketcher[6], incluant un système de build, des tests unitaires (via Jest) et un environnement de production.

Des hooks Git étaient aussi configurés pour automatiser certaines vérifications à chaque étape. Avant chaque commit, les fichiers sont formatés automatiquement par Prettier et ESLint, et toute erreur de format empêche l'action. De plus, avant chaque push sur le dépôt git, une série de tests est exécutée. Parmi ces tests on retrouve le formatage du code, les tests unitaires de Ketcher[6], la vérification du typage TypeScript, ainsi que l'analyse statique (tsc). Si un test échoue, le push est également bloqué.

GitHub a été un outil central pendant le développement, en facilitant le travail entre plusieurs appareils, le suivi des modifications et la gestion des versions.

Une attention particulière a été portée à la rédaction des messages de commit selon la convention Conventional Commits, ce qui a permis de garder un historique clair et structuré (ex : « feat(ketcher-react): Add minimize button to toolbar »).

Enfin, à la fin de chaque itération importante, des tests utilisateurs ont été réalisés par le chimiste afin de valider les fonctionnalités mises en place. Ces retours ont permis d'ajuster l'interface, de corriger certains comportements inattendus et d'améliorer l'ergonomie globale.

5.3. Défis rencontrés durant le développement

Le développement s'est effectué de manière progressive, avec chaque étape qui apportait de nouveaux défis à relever. Trois problèmes techniques sont illustrés dans cette partie ainsi que leur solution. Ceux-ci illustrent la complexité de l'intégration entre les outils, la gestion des états dans React, ainsi que le traitement des données de spectres RMN.

Maintien de la cohérence entre les IDs des atomes de Ketcher et de RDKit :

Pour assurer l'association précise entre les pics du spectre RMN et les atomes de la molécule, il était indispensable que les identifiants des atomes restent stables et cohérents entre le frontend (éditeur Ketcher[6]) et le prédicteur (basé sur RDKit[8]).

Lorsqu'une molécule est importée dans Ketcher[6] via un SMILES généré par RDKit[8], les IDs d'atomes restent les mêmes. Cependant, si on crée ou modifie une molécule dans Ketcher[6], le SMILES retourné n'est pas kekulisé ni canonique. Cela rend les IDs instables et une même molécule pouvait être interprétée différemment. Et donc ne pas avoir les mêmes IDs d'atomes comparés à RDKit[8], ce qui cassait l'association entre les atomes et les pics du spectre.

Pour résoudre cela, dès qu'un SMILES côté frontend est utilisé en dehors de Ketcher[6], il est envoyé au backend pour une conversion. Le SMILES est transformé en molécule dans RDKit[8], puis reconverti en SMILES kekulisé et canonique. Ce traitement garantit une représentation cohérente entre les deux outils, maintenant la correspondance entre les IDs des atomes.

Gestion des onglets et synchronisation des états dans React :

L'application permet de travailler sur plusieurs projets simultanément grâce à un système d'onglets, chaque onglet contenant ses propres données (molécule, spectre et associations).

Au départ, ces données étaient gérées via des useState React. Ceux-ci permettent à un composant de mémoriser une valeur propre à son contexte et de déclencher un nouveau rendu lorsque celles-ci changent. Cependant, lors de modifications rapides ou successives, des problèmes de synchronisation sont apparus, dus au fait que les mises à jour des useState sont asynchrones et ne reflètent pas immédiatement les changements. Cela entraînait des incohérences et des erreurs dans l'affichage et la manipulation des données.

La solution a donc été de remplacer ces états par des useRef, qui permettent de manipuler les données directement comme une variable mutable classique, garantissant ainsi des mises à jour immédiates. En contrepartie, comme useRef ne déclenche pas de re-render automatique, il a été nécessaire d'implémenter une gestion manuelle des rafraîchissements de l'interface pour certains composants, afin que l'affichage reste cohérent avec les données modifiées.

Extraction automatique des pics dans le spectre RMN :

Un des besoins clés était d'automatiser la lecture des spectres RMN pour extraire les pics, leurs positions (ppm), leurs intensités maximales et leurs multiplicités, ainsi que d'associer les données du spectre aux atomes correspondants dans la molécule.

Malheureusement, aucune solution open source simple ne permettait d'analyser directement des spectres RMN et d'en extraire ces informations.

Il a donc fallu développer un algorithme Python basé sur SciPy[10], et qui combine plusieurs techniques. On commence d'abord par un lissage du spectre pour réduire le bruit. Suivie d'une détection des pics grâce à la fonction find_peaks[12] de SciPy[10]. Puis un regroupement des différents pics proches en un groupe de multiplets appartement à un même pic selon des seuils définis expérimentalement. Enfin, on finit par une liaison entre ces pics établis et les atomes identifiés au travers de leurs IDs, lorsque ces données sont disponibles.

5.4. Déploiement et documentation

Une fois les fonctionnalités achevées, l'étape suivante a été le déploiement de l'application.

Dans ce processus de déploiement, on retrouve tout d'abord la compilation du frontend React en fichiers statiques (HTML, CSS et JS). Puis, la mise à jour de ces fichiers statiques dans le dépôt principal de l'application. Suivie d'une dernière phase de tests en conditions réelles pour valider le bon fonctionnement de l'application. Enfin, on fait le packaging de l'application via PyInstaller[11], afin de permettre la création d'un exécutable unique pour Linux et Windows (la version Mac est également possible).

Suite à cela, c'est le moment de mettre à jour la documentation technique pour refléter les dernières évolutions du projet. Puis, on s'occupe de rédigé le guide utilisateur, qui explique l'installation, le lancement de l'application, ainsi qu'une description détaillée des différentes fonctionnalités et de l'interface utilisateur.

Cette étape finale assure à la fois la stabilité du logiciel et facilite sa prise en main, que ce soit pour les utilisateurs finaux qui s'en serviront ou pour les futurs développeurs qui reprendront le projet.

6. Résultat final de l’application

À l’issue de la phase de développement logiciel, une version complète et fonctionnelle de l’application a été finalisée. Cette version intègre l’ensemble des fonctionnalités prévues initialement, ainsi que plusieurs évolutions issues des échanges avec l’utilisateur final.

L'application finale permet donc de dessiner une molécule via un éditeur intégré (version modifiée de Ketcher[6]). D’importer ou générer automatiquement un SMILES à partir du dessin. De lancer une prédiction de spectre RMN (avec un prédicteur factice dans un premier temps). Visualiser le spectre RMN de manière interactive (zoom, survol, clics, etc.). D’annoter le spectre de différentes manières (aucune annotation, IDs des atomes, ppm des multiplicités). De lier les pics du spectre aux atomes de la molécule. Gérer plusieurs projets simultanément via un système d’onglets. D’importer et d’exporter des projets, y compris sous forme de fichiers .json, .jdx, ou .zip (molécule + spectre + images annotées). Et enfin, de configurer les modèles de prédiction et d’ajuster leurs paramètres.

L’interface a été pensée pour rester claire, fluide et adaptée à un usage scientifique, tout en restant accessible pour des utilisateurs non techniques. L’application est distribuée sous forme d’exécutables pour Linux et Windows (version Mac possible), accompagnée d’un guide utilisateur et d’une documentation technique.

7. Limites et perspectives de l’application

La version actuelle de l’application a plusieurs limites. Tout d’abord, l'algorithme de détection de pics est perfectible. car l’analyse des spectres RMN repose sur un algorithme basé sur SciPy[10] (lissage + détection de pics) et fonctionne correctement, mais des cas limites ou bruits expérimentaux peuvent le mettre en défaut. Il y a aussi le cas du support partiel du format JCAMP-DX. Certains fichiers utilisent des variantes ou une compression avancée et ceux-ci ne sont pas interprétés entièrement par l’application. Une autre limite, est que le logiciel est adapté seulement pour la RMN à une dimension de type ^{1}H ou ^{13}C, mais ne prend pas en charge les spectres 2D, comme les spectres COSY, ou encore HMBC. Enfin, à ce moment-là, l’application ne dispose pas encore d’un véritable modèle de prédiction. Il s'agit d'un modèle de prédiction factice qui retourne toujours les mêmes données de prédiction. Le système est prêt pour une intégration réelle, mais la prédiction scientifique n’a pas encore été implémentée.

Bien que l’application ait plusieurs limitations, elle est aussi pleine de perspectives. Parmi ceux-ci, on peut retrouver l’amélioration de la détection de pics. C'est-à-dire, utiliser des valeurs expérimentales plus fiables ou des techniques d’apprentissage automatique pour fiabiliser l’analyse spectrale. On pourrait aussi ajouter une extension des fonctionnalités. Tel que le support de nouveaux types de spectres, une meilleure prise en charge des formats, et une visualisation scientifique plus avancée. A cela, on pourrait prévoir un déploiement multiplateforme facilité, avec une version en ligne qui pourrait étendre l’usage de l’outil, notamment dans contexte pédagogique ou collaboratif. Et pour le dernier point, il s’agit de l’intégration d’un modèle fonctionnel de prédiction RMN. Et ceci sera le point que nous allons aborder dans les parties suivantes, car nous arrivons à la partie bonus du stage qui consistait à l’étude des modèles de prédiction.

8. Vue d’ensemble du système de prédiction

8.1. Objectif du modèle

L’objectif est de faire un ou plusieur système de prédiction, qui permet de prédire les spectres RMN 1H et 13C a partir d’une molécule donnée au format SMILES.

Ce système prendra donc en entrée une molécule au format SMILES, ou tout autre informations pouvant être inférées sur la molécules à partir de son SMILES (ex: vecteurs de features atomiques obtenue par RDKIT).

En sortie, il retourne un spectre RMN sous la forme d’une liste de triplet avec le format (ppm, intensité, atomsIDs) où atomsIDs est une liste des IDs des atomes correspondant au pics du spectre.

8.2. Présentation générale de la stratégie

Il n’est pas possible de demander directement à un modèle de prédiction de prendre un SMILES en entrée et de produire tous les points d’un spectre RMN. Cela reviendrait à lui faire apprendre des relations chimiques complexes ainsi que la distribution des points du spectre RMN. Cela serait trop complexe pour qu’un modèle le fasse tout d’un coup.

Pour simplifier le problème, il a été choisi de le découper en plusieurs étapes. À partir du SMILES, on commence par extraire des informations sur les atomes de la molécule, appelées "features", qui décrivent leur environnement chimique. Ces données sont ensuite utilisées pour prédire les caractéristiques des pics RMN que la molécule devrait présenter.

Une fois ces informations prédites, on génère les courbes de chaque pic à l'aide de fonctions mathématiques. Et ces courbes sont ensuite fusionnées pour former le spectre complet.

Enfin, le spectre sera compressé en supprimant les segments où l'intensité est nulle afin d'en réduire la taille sans perdre d'informations utiles.

Cette méthode permet donc de passer d'une molécule (SMILES) à un spectre RMN de manière progressive, réaliste et plus facile à améliorer étape par étape.

8.3. Pipeline fonctionnel

Le pipeline fonctionnel correspond aux étapes qui seront mises en place pour prédire un spectre RMN à partir d'un SMILES une fois le modèle de prédiction entraînée.

On commence donc par la réception d'un SMILES en entrée. Et à partir de cette représentation moléculaire, les ensembles de noyaux équivalents (les protons ou les carbones selon le type du spectre) seront identifiés à l'aide d'outils chimiques comme RDKit[8]. Une fois ces ensembles définis, des features atomiques sont extraites pour chacun d'eux. Ces features traduisent des propriétés structurelles et électroniques, qui servent de base à la prédiction des caractéristiques des pics RMN.

Les vecteurs de features extraits sont ensuite soumis à un prétraitement, qui permet à la fois d'adapter leur format aux exigences de la suite du traitement, et d'en déduire certaines informations de la sortie lorsque cela est possible. En effet, toutes les caractéristiques nécessaires à la génération des pics du spectre ne requièrent pas forcément une prédiction. Certaines peuvent être obtenues directement à partir des features disponibles, tandis que d'autres, plus complexes, nécessitent une étape de prédiction dédiée. Une fois ces informations réunies, chaque ensemble de noyaux équivalents permet de générer une courbe représentant des pics, qui seront ensuite combinées pour former le spectre complet.

Ces courbes sont ensuite fusionnées en un spectre complet, représenté comme une séquence de triplets (ppm, intensité, atomsIDs), où chaque point est associé aux atomes responsables du signal. Pour réduire la taille des données, les segments du spectre où l'intensité reste constamment à zéro sont détectés automatiquement, et seuls les points de début et de fin de ces segments sont conservés. Ce processus permet une représentation efficace et compacte du spectre final, tout en conservant les données les plus utiles du spectre.

Ce pipeline fonctionnel permet donc de passer d'une simple chaîne SMILES à une simulation fidèle d'un spectre RMN, de manière entièrement automatique, reproductible, et scientifiquement interprétable.

9. Préparation des données

Cette partie va expliquer la préparation des données, en passant tout d'abord par la création du dataset qui recueille les informations de RMN sur plusieurs molécules. Puis, le format du dataset. Ainsi que les features qui sont extraites du dataset afin d'entraîner les modèles de prédiction.

Les éléments et données apportés dans cette partie correspondent à la première version des modèles. Nous verrons pas la suite dans la partie entraînement du modèles les améliorations apportées progressivement aux modèles dans le but de l'améliorer.

9.1. Génération du dataset

Pour entraîner les modèles de prédiction de spectres RMN, nous avons donc dû constituer notre propre dataset, faute de disposer d'une base de données publique adaptée, contenant à la fois des structures moléculaires, des spectres RMN, ainsi que les informations et l'association entres les pics du spectre et les atomes de la molécule.

Il a donc été décidé de créer un premier dataset en se servant d'un modèle de prédiction existant. Pour cela, on se repose sur les prédictions obtenues à l'aide du simulateur de spectres RMN proposé par le site nmrdb.org[13].

Pour le premier jeu de données, l'ensemble sélectionné compte 500 molécules obtenues depuis la base de données moléculaire PubChem[14]. Parmi ceux-ci, 475 molécules organiques ont été choisies en ne retenant que celles contenant les éléments hydrogène, carbone, azote, oxygène et soufre, afin de cibler la chimie organique. Et les molécules retenues devaient aussi compter entre 5 et 30 atomes. Pour améliorer la généralisation du modèle, 25 molécules non organiques, soit environ 5 % du dataset, ont été ajoutées. Au final, il y a aura deux dataset différents. Un pour la RMN 1H et un autre pour la RMN 13C, afin d'entraîner deux modèles spécifiques à ces types de RMN.

Du au manque d'accès via une API, la génération des données s'est appuyé sur un scraper utilisant Selenium[15] et développé spécifiquement pour interagir avec l'interface web de nmrdb.org[13]. Ce dernier récupère pour chaque molécule la prédiction des spectres RMN, la molécule au format molfile pour garder l'ordre des atomes, les informations des pics, ainsi que la table d'association pics-atomes correspondants dans la structure moléculaire.

Les données extraites sont ensuite traitées pour être compatibles avec notre modèle. La représentation des molécules est standardisée au format SMILES kékulisé via la bibliothèque RDKit[8] et en supprimant les hydrogènes explicites. Les spectres simulés sont récupérés au format JCAMP-DX. Puis, le même algorithme que celui utilisé dans notre application détecte les pics dans ces spectres et associe les IDs des atomes correspondant à tous les points du spectre qui appartiennent au pic, assurant ainsi la conformité avec le format attendu pour le modèle de prédiction. On récupère aussi telle quelle la table d'informations sur les pics du spectre.

Chaque molécule est stockée dans un fichier individuel au sein du dataset, reprenant en partie le format des projets utilisés dans l'application PredictionRMN, ce qui permet de visualiser directement les données dans l'interface associée. Chaque fichier contient donc la structure au format SMILES, les données du spectre au format d'une liste de triplet (ppm, intensité, atomsIDs), ainsi que la table d'informations de chaque pics du spectre.

En revanche, certaines limitations ont été remarquées, et ce sont partiellement les mêmes que dans l'application. Tout d'abord, pour les spectres 13C nmrdb.org[13] utilise un format compressé dans le JCAMP-DX, donc on retrouve à nouveaux le problème où moins de points existe dans le spectre, ce qui réduit la résolution des pics et peut affecter la qualité de la détection. Deuxièmement, quelques divergences avec les données de nmrdb on été remarquées lors de la détection de pics, comme la détection de deux pics au lieu d'un seul, notamment pour les structures moléculaires de type benzène. Si le temps le permet, ces imperfections seront corrigées dans les futures itérations.

Ces deux datasets ont été conçus comme une base de travail pour entraîner un premier modèle de prédiction des spectres RMN 1H et 13C. Les données, ainsi que les codes de génération et d'entraînement, seront aussi publiés sur le **dépôt GitHub** spécifique au modèle de prédiction du projet PredictionRMN afin de garantir la transparence et la reproductibilité des travaux.

9.2. Format des données du dataset

Le dataset final est organisé sous la forme de plusieurs fichiers JSON, avec un fichier par molécule. Chaque fichier contient trois parties principales : la représentation moléculaire en SMILES, la table d'informations sur les pics et le spectre RMN simulé.

La structure moléculaire est stockée dans un objet json "molecules", qui précise que le format utilisé est un SMILES (kékulisé et obtenu via RDKit[8]).

On retrouve aussi les informations sur les différents pics des spectres. Et parmi ces informations, on retrouve des données comme le ppm du pic, les atomes correspondants, le nombre d'atomes parmi les noyaux équivalents (ex : atomes d'hydrogène pour la 1H), la multiplicité du pic ainsi que les valeurs de couplings.

Le spectre RMN lui est enregistré sous la clé "spectrum", qui contient une liste ordonnée de points du spectre au même format que celui utilisé dans l'application vue précédemment. Chaque point est un triplet associant la position chimique en ppm ("ppm"), l'intensité correspondante ("intensity") et la liste des identifiants des atomes associés à ce point ("atomID").

Pour illustrer cette organisation, un exemple simplifié du format JSON utilisé dans le dataset est présenté en **Figure 10**.

```
{
  "molecules": {
    "format": "SMILES",
    "data": "OC1C=C(Cl)C(O)C=C1Cl"
  },
  "association": [
        ...
     {
        "ppm": 3.348,
        "atoms": [4],
        "nb_atoms": 2,
        "multiplicity": "t",
        "couplings": [5.443],
     },
        ...
  ]
  "spectrum": [
        ...
    {
      "ppm": 4.9294,
      "intensity": 2742325.05,
      "atomID": [1, 5]
    },
        ...
  ]
}
```

Figure 10 – Format utilisé dans le dataset

9.3. Extraction des features 1H

L'extraction des features est une étape fondamentale permettant de convertir la structure moléculaire en une représentation numérique exploitable pour l'apprentissage. Pour chaque molécule, la structure est d'abord convertie en un objet RDKit[8], ce qui permet d'accéder à une multitude d'informations sur les atomes, leurs liaisons et leur environnement chimique.

Les features extraites concernent principalement les atomes d'hydrogène et leur environnement immédiat. Elles incluent des propriétés atomiques individuelles telles que le type d'atome (par exemple carbone, hydrogène, azote, oxygène), l'hybridation (sp, sp^2, sp^3), la charge formelle, et le nombre de voisins directs. Ces caractéristiques sont essentielles pour comprendre la nature électronique locale autour des atomes.

En complément, les features prennent en compte l'environnement local en résumant la nature des atomes voisins par des statistiques (moyenne, écart-type, nombre total), ainsi que la présence de cycles aromatiques ou non-aromatiques auxquels l'atome pourrait appartenir. Ces informations aident à modéliser les effets de champ magnétique local qui influencent fortement le déplacement chimique.

Enfin, des propriétés globales des groupes d'hydrogènes équivalents sont extraites, telles que le nombre de protons équivalents, leur symétrie et leur position dans la molécule, car ces paramètres impactent directement les signatures RMN observées.

Ces features sont sélectionnées car elles capturent les facteurs structuraux et électroniques influençant directement les propriétés spectrales RMN, notamment le déplacement chimique, la multiplicité et les couplages entre protons. Grâce à ces informations, le modèle peut apprendre à associer des signatures spectrales précises aux environnements chimiques des hydrogènes.

9.4. Extraction des features 13C

Pour le modèle 13C, l'extraction des features cible principalement les atomes de carbone dans la molécule, afin de comprendre leur environnement chimique spécifique, qui influence directement les déplacements chimiques 13C en RMN.

Les caractéristiques extraites incluent des propriétés atomiques telles que : son degré de connexion (nombre de liaisons avec des atomes voisins), son hybridation (sp, sp^2, sp^3), la présence dans un cycle (aromatique ou non),ainsi que la charge formelle et la charge partielle calculée.

En complément, le nombre et la nature des voisins directs sont détaillés, en particulier le nombre d'atomes d'hydrogène attachés, mais aussi les différents types d'atomes lourds environnants (carbone, oxygène, azote, soufre, halogènes). Ces informations aident à modéliser les effets électroniques locaux autour du carbone.

Des distances topologiques spécifiques sont aussi mesurées, comme la distance des carbones au plus proche atomes d'oxygène, d'azote, ou à un cycle aromatique, capturant ainsi les influences chimiques à plus grande échelle.

Enfin, une indication sur la symétrie locale est donnée par une feature binaire, signalant si le carbone est dans un environnement chimique symétrique, ce qui peut affecter la multiplicité et la forme du signal RMN.

Ces features, en synthétisant la structure chimique locale et les influences électroniques autour du carbone, permettent au modèle de prédire précisément les déplacements chimiques des carbones.

10. Choix du modèle de prédiction

Une fois la structure globale du système de prédiction définie, il est nécessaire de déterminer quel type de modèle de prédiction est le plus adapté pour estimer certaines des propriétés des pics RMN à partir des features extraites. Cette étude comparative permet donc de sélectionner un algorithme adapté aux différentes contraintes du projet.

10.1. Étude comparative des modèles

Pour sélectionner le modèle le plus adapté à notre projet, nous avons comparé plusieurs approches. Le tableau récapitulatif de cette étude comparative est présenté en **Figure 11**.

Modèle	Type	Avantages principaux	Limites / Données nécessaires
Régression Linéaire	ML classique	Très simple, rapide, interprétable	Trop basique, peu adapté à des données complexes
k-NN Regression	Distance-based	Facile à implémenter, interprétation intuitive	Sensible au bruit, lent à l'inférence, peu robuste
Random Forest[16]	ML classique	Robuste, peu de réglages, bon pour des petits datasets	Fonctionne avec des vecteurs de features, ~1k+ échantillons
XGBoost LightGBM	ML classique	Très précis, gère bien les interactions complexes	Tuning nécessaire, l'idéal est d'avoir 3k échantillons
MLP (réseau de neurones)	Deep Learning	Gère bien les non-linéarités, évolutif	Risque d'overfitting, nécessite +10k échantillons
Graph Neural Network (GNN)	Deep Learning	Très adapté aux molécules (liaisons, atomes)	Mis en place plus longue, l'idéal est d'avoir 20k+ échantillons

Figure 11 – Tableau comparatif des différents *modèles de prédiction étudiés*

10.2. Choix du modèle

Compte tenu des contraintes spécifiques de ce projet, il est essentiel de choisir un modèle à la fois performant et réaliste à mettre en place dans un temps limité. La partie du stage qui est attribué à l'étude des modèles, est d'environ une semaine, et avec un niveau de départ peu expérimenté en intelligence artificielle. De plus, aucun jeu de données n'est directement disponible, ce qui implique de générer soi-même un dataset, ce qui peut être chronophage. Il faut donc favoriser l'utilisation sur des petits datasets.

Dans ce contexte, il est préférable d'opter pour un modèle qui fonctionne bien avec un petit volume de données, et qui ne demande pas de connaissances techniques avancées ou de configurations complexes. Il est également important de choisir une solution gratuite, sans dépendance à des outils propriétaires ou payants.

Parmi les différents modèles étudiés, le Random Forest[16] apparaît comme le plus adapté à ces exigences. Il s'agit d'un algorithme de machine learning classique, facile à mettre en œuvre, bien documenté, et largement utilisé pour des tâches de régression. Il offre de bonnes performances même sur des datasets de taille modérée (quelques centaines à milliers d'échantillons), et tolère assez bien des données bruitées ou incomplètes.

En plus de sa robustesse, le Random Forest[16] peut être directement utilisé avec des vecteurs de caractéristiques extraits via RDKit[8], ce qui permet d'exploiter les informations structurales et chimiques de la molécule sans devoir encoder directement le SMILES ou construire des graphes complexes.

En résumé, le Random Forest[16] offre un excellent compromis entre simplicité, efficacité et accessibilité, ce qui en fait le choix le plus pertinent pour ce projet d'étude à court terme.

11. Entraînement des modèles

Dans la première version des modèles, le pipeline d'entraînement commençait par l'exploitation de l'ensemble des résultats RMN issus du dataset contenant les structures moléculaires au format SMILES ainsi que les associations RMN (valeurs de déplacement chimique, multiplicités, couplages, etc.). Pour chaque molécule, la structure est convertie en objet RDKit[8] et enrichie avec des hydrogènes explicites. À partir de cette représentation, on identifie les noyaux équivalents (hydrogènes pour la 1H et carbones pour la 13C), puis on extrait les features pour chacun d'eux.

Une fois les features extraites, elles sont fusionnées avec les labels RMN disponibles (comme le déplacement chimique, le nombre de protons, la multiplicité et les constantes de couplage) en associant chaque ensemble de noyaux équivalents à la description spectrale correspondante à son pic. Le résultat est un jeu de données tabulaire où chaque ligne correspond à des noyaux équivalents, avec ses features chimiques en entrée et ses propriétés RMN en sortie.

Avant l'apprentissage, le jeu de données est prétraité. Les colonnes numériques sont utilisées telles quelles, tandis que les colonnes catégorielles comme l'hybridation sont transformées via un encodage one-hot. Des colonnes spécifiques telles que la liste des types atomiques voisins sont résumées par des statistiques (moyenne, écart-type, nombre d'éléments). Les constantes de couplage, représentées sous forme de liste variable, sont normalisées en vecteurs de longueur fixe grâce à un transformateur personnalisé.

Pour garantir une bonne évaluation des performances, le dataset est divisé en deux ensembles distincts : un ensemble d'entraînement (train) et un ensemble de test (test). La séparation est réalisée de manière aléatoire tout en veillant à conserver une distribution représentative des molécules dans chaque groupe, généralement selon un ratio de 80 % pour l'entraînement et 20 % pour le test. Cette séparation permet d'entraîner le modèle sur un sous-ensemble des données, puis de mesurer sa capacité à généraliser sur des données non vues durant l'entraînement.

Le pipeline d'entraînement est implémenté en Python, et s'appuie sur plusieurs bibliothèques clés. On retrouve d'abord RDKit[8] pour la manipulation des structures moléculaires, l'extraction des ensembles de noyaux équivalents et les features chimiques. Puis il y a aussi Scikit-Learn[17] pour le prétraitement des données (encodage, normalisation), la gestion de la séparation train/test, et l'implémentation du modèle Random Forest[16] ainsi que des modèles multi-sorties. Et des outils personnalisés sont utilisés pour gérer la normalisation des constantes de couplage et l'assemblage des vecteurs de features.

La v1 du modèle 1H utilise une approche multi-sortie. Quatre prédicteurs différents sont entraînés simultanément : Deux régressions pour prédire le déplacement chimique et le nombre d'atomes par groupe. Une classification pour prédire la multiplicité du pic (singulet, doublet, triplet, etc.). Et une régression multivariée pour prédire les constantes de couplage. Tandis que pour le modèle 13C, on utilise seulement une seule sortie avec une régression pour le déplacement chimique. Et les autres valeurs sont prédites via une logique métier.

Tous les prédicteurs reposent sur l'algorithme Random Forest[16], reconnu pour sa robustesse face aux données mixtes (numériques et catégorielles), sa tolérance au bruit et sa simplicité d'utilisation. L'ensemble du pipeline (préprocesseurs, modèles, outils de transformation) est encapsulé dans une structure modulaire, facilitant la réutilisation, la mise à jour et l'évaluation des modèles.

12. Résultats et évaluation des modèles

12.1. Méthodologie d'évaluation des modèles

L'évaluation des modèles repose sur un ensemble de métriques et des tests de visualisation qui permettent d'avoir un retour objectif sur les performances.

Pour les sorties numériques continues des modèles (déplacement chimique, nombre d'atomes équivalents, constantes de couplage), deux métriques principales sont utilisées.

La première est la MAE[18] (Mean Absolute Error) qui donne une indication directe de la précision moyenne des prédictions, en mesurant l'erreur absolue entre la valeur prédite et la valeur réelle. Et la seconde métrique est la RMSE[19] (Root Mean Squared Error), qui reflète la stabilité du prédicteur, en pénalisant davantage les erreurs importantes.

Tandis que pour la prédiction de la multiplicité, qui est un problème de classification, on utilise l'accuracy[20], qui mesure la proportion de prédictions exactes par rapport aux classes attendues (singulet, doublet, triplet, etc.).

En complément des métriques, des test de visualisations de spectres sont générées sur plusieurs exemples de test. Cela permet une validation plus qualitative des modèles, en observant par exemple la disposition globale des pics dans le spectre (cohérence sur l'axe du déplacement chimique). La taille relative des pics, reflétant le nombre de noyaux équivalents. La multiplicité correcte des signaux. Ainsi que l'association correcte des pics aux bons atomes de la molécule.

Ces visualisations offrent une perspective utile sur la fidélité des prédictions et mettent en évidence certains écarts subtils que les métriques globales ne capturent pas toujours.

L'objectif de cette méthodologie est double. Il s'agit d'abord d'avoir une mesure fiable des performances des modèles sur différents aspects du spectre RMN. Ainsi que d'identifier les forces et les faiblesses des modèles, afin d'orienter les améliorations dans les prochaines versions. Ces évaluations permettent ainsi de comparer objectivement différentes configurations de modèles et de guider les choix d'optimisation à venir.

12.2. Évolution et améliorations des modèles

Les modèles de prédiction ont été améliorés de façon itérative en se basant sur les métriques expliquées précédemment, utilisées comme indicateurs pour optimiser les modèles. L'objectif est d'obtenir des modèles aussi précis que possible tout en maintenant une bonne stabilité.

Le tableau ci-dessous en **Figure 12** présente une synthèse des améliorations apportées à chaque version pour le modèle 1H, ainsi que l'évolution attendue des résultats. Des analyses et commentaires sont également proposés pour chaque version. Le **tableau complet** des métriques et des features pour chaque version est disponible en annexe.

Légende :
🟢 Amélioration — 🔴 Baisse — 🔵 Référence / Stable

Version 1H	Modifications principales	Évolution des métriques	Analyse et commentaires
v1	- Pipeline initial - Features initial - Dataset de 500 échantillons	🔵 AtomsIDs 🔵 ppm 🔵 nbAtoms 🔵 multiplicité 🔵 couplages 🔵 Visualisation	- Bon point de départ - Précision très correcte sur les ppm, les atomsIDs et le nombre d'atomes - Multiplicités et couplage à améliorer - Résultats en visualisation très correcte
v2	- Dataset augmenté à 1000 échantillons	🔵 AtomsIDs 🟢 ppm 🔴 nbAtoms 🟢 multiplicité 🟢 couplages 🔴 Visualisation	- Amélioration des multiplicités et du couplage mais toujours pas assez précis - PPM plus précis mais moins stables dans certains cas - La MAE[18] du nombre d'atomes augmente mais reste raisonnable - Résultats en visualisation moins fidèles
v3	- Intégration de features spécifiques à la multiplicité - Estimation du nombre d'atomes via la fusion des pics et la feature ynum_H, au lieu du modèle de prédiction	🔵 AtomsIDs 🟢 ppm 🔴 nbAtoms 🟢 multiplicité 🟢 couplages 🟢 Visualisation	- Amélioration des multiplicités et du couplage - La MAE[18] du nombre d'atomes augmente encore mais cela reste toujours raisonnable - PPM plus précis et plus stable - Résultats en visualisation plus fidèles - Encore quelque erreur sur la multiplicité - Amélioration de la fusion des pics et de leur génération

Figure 12 – Synthèse des améliorations et résultats par version du modèle 1H

Pour ce qui est du modèle de prédiction pour la RMN 13C, celui-ci a été effectué après le modèle 1H, donc il profite d'entrée de jeux des améliorations apportées par les différentes versions de celui-ci.

Les différences principales entre les modèles 1H et 13C résident dans la sélection des atomes étudiés (les carbones au lieu des groupes d'hydrogènes) ainsi que dans les features extraites spécifiquement pour ces carbones. Par ailleurs, contrairement au modèle 1H qui prédit plusieurs paramètres spectraux, le modèle 13C se concentre uniquement sur la prédiction des déplacements chimiques (ppm). Cela est dû au fait que la multiplicité du signal 13C est généralement considérée comme un singulet par défaut, et les valeurs de couplage sont fixées à une valeur nulle, ce qui simplifie la modélisation des couplages dans ce cas. Et pour ce qui est du nombre d'atomes, cette valeur est déterminée via la fusion des pics comme pour la 1H.

On retrouve de nouveaux le tableau de synthèse des améliorations par version mais cette fois-ci pour la RMN 13C en **Figure 13**. Comme pour la 1H le **tableau complet** des métriques et des features par version du 13C est aussi disponible en annexe.

Légende :
🟢 Amélioration — 🔴 Baisse — 🔵 Référence / Stable

Version 13C	Modifications principales	Évolution des métriques	Analyse et commentaires
v1	- Pipeline initial - Features initial - Dataset de 500 échantillons	🔵 AtomsIDs 🔵 ppm 🔵 nbAtoms 🔵 multiplicité 🔵 couplages 🔵 Visualisation	- Très bon point de départ - Précision très élevé sur les ppm, les atomsIDs et le nombre d'atomes - Multiplicités et couplage correct car ils sont fixes - Résultats en visualisation très fidèle car plus simple que la 1H
v2	- Dataset augmenté à 1000 échantillons	🔵 AtomsIDs 🟢 ppm 🟢 nbAtoms 🔵 multiplicité 🔵 couplages 🟢 Visualisation	- atomsIDs, multiplicité et couplage toujours aussi bon - La MAE[18] du nombre d'atomes baisse un peu mais cela est négligeable - PPM plus précis que la v1 - Visualisation encore plus fidèles

Figure 13 – Synthèse des améliorations et résultats par version du modèle 13C

12.3. Résultat final des modèles de prédictions

Le modèle 1H (v3) présente de bonnes performances globales, avec notamment une MAE[18] ppm de 0.131 et une RMSE[19] ppm de 0.317, ainsi qu'une accuracy[20] en multiplicité de 85,4 %. De plus, les déplacements chimiques (ppm) et le nombre d'atomes sont prédits avec précision et sont plutôt stables. L'ajout de nouvelles features spécifiques aux environnements chimiques, telles que le nombre d'hydrogènes voisins ou la présence dans des groupes symétriques, a permis d'améliorer la qualité de la prédiction, en particulier pour les multiplicités et les couplages. En visualisation, le spectre généré est fidèle aux données attendues.

Cependant, certaines limites subsistent : des erreurs sur la multiplicité sont encore présentes, et l'algorithme actuel de fusion des pics utilise une valeur fixe de 0.15 ppm, qui peut être trop élevée et gagnerait à être affinée en tenant compte de la fréquence d'analyse RMN. Par ailleurs, bien que le temps de prédiction soit globalement satisfaisant, la fusion des pics et la détection des pics côté frontend de l'application restent des étapes relativement lentes, ce qui impacte la réactivité globale.

Pour le modèle 13C, les bases ont été posées en reprenant la structure et les évolutions du modèle 1H, avec une adaptation des features et de la sortie aux spécificités du carbone. Après quelques obstacles initiaux, notamment liés à l'absence d'assignation des atomes dans le prédicteur 13C de nmrd.org, un dataset d'entraînement a finalement pu être généré en grâce à un deuxième scrapper pour NMRium[5] cette fois-ci. Cela a permis d'entraîner et de tester efficacement un modèle Random Forest 13C, avec des performances prometteuses dès la première version, et d'encore les améliorées dans la version numéro 2.

Les performances obtenues sur le modèle 13C sont globalement satisfaisantes. La version 2 atteint une MAE[18] de 2.345 ppm et une RMSE de 5.557 ppm pour les déplacements chimiques. Si cette valeur peut sembler élevée comparée au modèle 1H (MAE[18] ~0.1 ppm), il faut noter que le spectre du carbone 13 s'étend sur une échelle bien plus large, allant de 0 à 250 ppm (contre 0 à 10 ppm pour le 1H), ce qui rend cette précision tout à fait acceptable. La MAE[18] sur le nombre d'atomes et l'accuracy[20] sur la multiplicité atteignent respectivement 0.1 et 100 %, ce qui témoigne de la robustesse du modèle sur ces aspects. Le jeu de features, largement inspiré de celui du modèle 1H mais adapté aux spécificités du carbone, s'est montré pertinent pour capturer les principales variations de l'environnement chimique.

En général, les modèles développés présentent des performances encourageantes. Toutefois, certaines limitations subsistent. Le traitement de la multiplicité et des couplages en 1H pourrait encore être amélioré à l'aide d'une logique métier plus avancée, notamment pour mieux gérer les cas ambigus ou complexes. De plus, les molécules sélectionnées pour l'entraînement ne sont sûrement pas assez nombreuses et ne sont peut-être pas suffisamment représentatives de la diversité chimique rencontrée en pratique. Avec plus de temps et de ressources, il aurait été possible de raffiner les algorithmes de post-traitement, d'enrichir les features, et d'améliorer la qualité du modèle et des données, ce qui aurait sans doute permis d'atteindre des performances encore meilleures.

13. Retours d'expérience global

En soi, ce stage m'a permis d'aborder un projet à l'intersection de l'informatique et de la chimie, un domaine scientifique que je connaissais très peu au départ. Cela a été à la fois un défi et une opportunité d'apprentissage.

Le défi principal a été d'aborder un projet dans un domaine scientifique, la chimie, alors que je n'avais plus pratiqué cette matière depuis le lycée. Je n'avais aucune connaissance en spectroscopie RMN ou sur les formats comme SMILES. J'ai donc dû rapidement me réapproprier certaines bases, et faire des recherches personnelles pour mieux comprendre les bases des atomes et des molécules, ainsi que les particularités des formats et représentations utilisés (kekulisation, canonisation, etc.).

Cela m'a amené à faire quelques erreurs liées à ces méconnaissance, mais cela à aussi été très formateurs. Ces erreurs m'ont poussé à faire davantage de veille technique et scientifique pour mieux comprendre ce que je manipule. Et au final, cela m'a permis de renouer avec le monde des sciences, qui est un domaine que j'ai toujours apprécié.

Un autre point important est que le projet nécessitait une collaboration avec un chimiste, afin de bien comprendre les différents besoins. J'appréhendais un peu cette communication, pensant que mon manque de connaissances scientifiques allait être un problème. Mais au final, cela s'est très bien passé. Grâce à une communication soignée et des échanges réguliers, la collaboration a été fluide et productive. Le fait que le chimiste soit lui-même familier avec le monde de l'informatique a également facilité les choses.

D'un point de vue purement développement logiciel, avant ce stage, j'avais déjà réalisé plusieurs projets en développement web avec JavaScript, mais ceux-ci étaient plutôt simples. Développés de zéro, et avec peu de contraintes d'intégration. Ce stage m'a confronté à une autre réalité, où il s'agissait ici de reprendre et adapter des composants existant (Ketcher[6]), en y intégrant des fonctionnalités nouvelles dans un environnement déjà structuré et techniquement exigeant.

Cela m'a forcé à réfléchir à la manière d'intégrer proprement des éléments dans un projet existant, à composer avec les choix techniques déjà faits, et à respecter une architecture hybride mêlant JavaScript/React pour le frontend et Python/Flask[9] pour le backend.

De plus, j'ai aussi découvert toute une organisation autour du développement de Ketcher[6], avec un système automatisé qui vérifie la qualité du code à chaque commit ou push. Cela inclut du linting (via Prettier et ESLint), des tests unitaires, des vérifications de typage TypeScript, et des conventions de nommage strictes pour les messages de commit (Conventional Commits). J'ai trouvé cette rigueur très formatrice, et j'ai pu en tirer des bonnes pratiques pour mes futurs projets.

Maintenant passons à mes retours sur la partie élaboration des modèles de prédiction. En réalité, ce stage a été ma première vraie expérience de développement d'un modèle de prédiction en partant de zéro, et dans un contexte plus réel et plus complexe qu'un simple environnement d'apprentissage comme un jeu ou un tutoriel. Au travers de cette expérience, j'ai rapidement découvert que certains préjugés que j'avais sur cette discipline étaient faux.

Au départ, je pensais que mon rôle serait principalement de générer des données d'entraînement pour alimenter un modèle déjà défini et de trouver les bons hyperparamètres. En réalité, j'ai dû concevoir toute une chaîne de traitement afin de choisir les bonnes features, structurer les données, construire le modèle de prédiction, puis générer les résultats sous la forme convenue. Je me suis vite rendu compte que la prédiction d'un spectre RMN était un problème complexe, et qu'il ne suffirait pas de donner les données à un modèle pour qu'il nous sorte un joli spectre à la fin.

Plutôt que de prédire directement un spectre, j'ai dû réfléchir à des méthodes alternatives afin de simplifier la tâche demandée au modèle de prédiction. J'ai donc mis en place des modules de pré-processing et post-processing pour donner les données les plus adaptées au modèle, et ensuite exploiter les données sorties du modèle pour générer le résultat attendu. Et ce découpage m'a permis de simplifier le travail du modèle tout en gardant un bon contrôle sur le résultat final.

Ce travail m'a aussi amené à approfondir mes connaissances scientifiques, notamment sur les liens entre caractéristiques chimiques et propriétés spectrales RMN, afin de concevoir des features pertinentes et essayer d'améliorer les résultats de la prédiction. Cela m'a aidé à mieux comprendre comment le spectre est influencé par certaines caractéristiques chimiques de la molécule. Et cela m'a aussi permis de mieux comprendre certains concepts que je ne comprenais pas vraiment, comme la façon dont la multiplicité et le coupling influencent la forme des pics dans le spectre.

Dans les faits, l'amélioration des modèles s'est faite de manière itérative où à chaque version j'évaluais les performances à l'aide de métriques, et c'est ce qui m'a permis d'identifier les points faibles et de proposer des solutions concrètes. Et j'ai donc bien fait de commencer par mettre en place une démarche d'analyse des modèles, car c'est ce qui m'a permis de prendre des décisions, et d'ajuster mes choix en conséquence.

Au final, cette partie du projet m'a montré que la création d'un modèle de prédiction va bien au-delà de ce que je pensais. Cela implique de comprendre en détail le problème traité, de concevoir tout un pipeline adapté, et de travailler en accord avec les éléments scientifiques sur lesquels repose le domaine étudié.

14. Annexe

14.1. Liens vers les dépôts GitHub

- Frontend (interface utilisateur, intégration de Ketcher, React, Plotly) : https://github.com/KreeZeG123/CustomKetcher
  (branch `custom-ketcher-v3.1.0-with-app`, dossier `example/`)

- Backend et application finale (Python Flask, API, RDKit, SciPy, modèle de prédiction factice, exécutables) : https://github.com/KreeZeG123/PredictionRMN

- Modèles de prédictions (1H et 13C) https://github.com/KreeZeG123/PredictionRMN-Models

14.2. Dossier de conception complet

Une archive a été créée pour rassembler l'ensemble des documents produits au cours des phases de conception du projet. On y retrouve :

- Les documents de choix des technologies (+ benchmark)
- Les cahiers de conception (architecture, modélisation, fonctionnalités)
- Les cahiers de conception UI / UX (maquette, chemin utilisateur, charte graphique)
- Les diagrammes UML et C4
- Les versions successives des documents

Archive disponible ici :
https://drive.google.com/file/d/1Cq8jtVs2qFykD_C70186Mb3qZudWxfeM/view?usp=sharing

14.3. Metric des différentes version du modèles de prédiction 1H :

Version 1H	Métriques clés	Features
v1	MAE ppm: 0.169 RMSE ppm: 0.294 MAE nb_atoms: 0.120 RMSE nb_atoms: 0.417 Accuracy multiplicity: 0.774 MAE couplings: 0.698 RMSE couplings: 1.385	- heavy_atomic_num - num_H - heavy_degree - heavy_hybridization - heavy_in_ring - heavy_is_aromatic - heavy_formal_charge - heavy_partial_charge - num_heavy_neighbors - neighbor_atomic_nums - num_H_neighbors
v2	MAE ppm: 0.147 RMSE ppm: 0.366 MAE nb_atoms: 0.140 RMSE nb_atoms: 0.490 Accuracy multiplicity: 0.835 MAE couplings: 0.469 RMSE couplings: 1.140	**Même features que la v1**
v3	MAE ppm: 0.131 RMSE ppm: 0.317 MAE nb_atoms: 0.420 RMSE nb_atoms: 0.870 Accuracy multiplicity: 0.854 MAE couplings: 0.486 RMSE couplings: 1.068	**Pareil que v1 et avec ces features en plus :** - neighbor_H_counts _per_atom - num_couplable_H _neighbors - distance_to_nearest _non_equivalent_H - in_symmetric_env - H_coupling_type - is_in_CH3 - is_in_CH2 - is_terminal_CH

14.4. Metric des différentes version du modèles de prédiction 13C :

Version 13C	Métriques clés	Features
v1	MAE ppm: 2.963 RMSE ppm: 5.967 MAE nb_atoms: 0.211 RMSE nb_atoms: 0.522 Accuracy multiplicity: 1.000 MAE couplings: 0.000 RMSE couplings: 0.000	- heavy_atom_idx - degree - hybridization - in_ring - is_aromatic - formal_charge - partial_charge - num_H_neighbors - num_heavy_neighbors - neighbor_atomic_nums - neighbor_C_count - neighbor_O_count - neighbor_N_count - neighbor_S_count - neighbor_Halogen _count - dist_to_O - dist_to_N - dist_to_aromatic - in_symmetric_env
v2	MAE ppm: 2.345 RMSE ppm: 5.557 MAE nb_atoms: 0.198 RMSE nb_atoms: 0.494 Accuracy multiplicity: 1.000 MAE couplings: 0.000 RMSE couplings: 0.000	**Même features que la v1**

15. Tables des figures

16. Bibliographie

1. Modélisation C4 :
https://www.alexandrevandekerkhove.fr/2020/02/10/c4model-pour-les-diags-darchi.html

2. Format SMILES :
https://fr.wikipedia.org/wiki/Simplified_Molecular_Input_Line_Entry_System

3. Résonance Magnétique Nucléaire (RMN ou NMR) :
https://fr.wikipedia.org/wiki/Spectroscopie_RMN
https://fr.wikipedia.org/wiki/R%C3%A9sonance_magn%C3%A9tique_nucl%C3%A9aire

4. Mestrelab. Mnova NMR Predict :
https://mestrelab.com/software/mnova/nmr-predict/

5. NMRium (Zakodium) :
https://www.nmrium.org

6. Ketcher :
https://lifescience.opensource.epam.com/ketcher/

7. Plotly :
https://plotly.com/

8. RDKit :
https://www.rdkit.org/

9. Flask :
https://flask.palletsprojects.com/en/stable/

10. SciPy Documentation :
https://docs.scipy.org/doc/scipy/

11. Pyinstaller :
https://pyinstaller.org/en/stable/

12. Fonction utilisée pour l'extraction de pics :
scipy.signal.find_peaks
https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html

13. nmrdb.org (Online NMR Predictor) ;
https://www.nmrdb.org.

14. PubChem (Base de donnée moléculaire) :
https://pubchem.ncbi.nlm.nih.gov/

15. Selenium (Navigateur Web Automatisable) :
https://www.selenium.dev/

16. RandomForest :
RandomForestRegressor
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
RandomForestClassifier
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

17. Scikit-Learn :
https://scikit-learn.org/stable/

18. Mean Absolute Error (MAE) :
https://en.wikipedia.org/wiki/Mean_absolute_error

19. Root Mean Square Deviation (RMSE) :
https://en.wikipedia.org/wiki/Root_mean_square_deviation

20. Accuracy :
https://en.wikipedia.org/wiki/Accuracy_and_precision