Année universitaire 2024-2025

**RAPPORT DE STAGE :**

*CRÉATION D’INTERFACE DE PILOTAGE DE SOLVEUR D’EMPLOIS DU TEMPS UNIVERSITÉ*

***Encadré par :***

*M. David Lesaint, M. David Genest, M. Marc Legeay,*

*M.Vincent Barichard, M. Aurélien Simon, M. Corentin Behuet*

***Réalisé par :***

*Mme. BUI Thi Vi*

*M. ANANI Messan Joseph*

*M. NDAYIZEYE Guy Keny*



# REMERCIEMENT

Nous tenons à exprimer notre sincère gratitude à l'Université d'Angers et au Laboratoire d'Étude et de Recherche en Informatique d'Angers (LERIA) pour nous avoir offert l'opportunité de réaliser ce stage enrichissant.

Nos remerciements s'adressent particulièrement à notre équipe d'encadrants : M. David Lesaint, M. David Genest, M. Marc Legeay, M. Vincent Barichard, M. Aurélien Simon et M. Corentin Behuet. Leur expertise, leurs conseils pertinents et leur disponibilité ont été déterminants dans la réussite de ce projet. Leurs retours constructifs lors de nos réunions hebdomadaires nous ont permis d'améliorer continuellement notre travail et d'approfondir notre compréhension des problématiques liées à la génération d'emplois du temps universitaires.

Nous souhaitons également remercier l'ensemble du personnel du LERIA pour leur accueil chaleureux et le cadre de travail stimulant qu'ils nous ont offert tout au long de cette période.

Cette expérience a été particulièrement formatrice, tant sur le plan technique que sur celui de la gestion de projet en équipe, et constituera sans aucun doute un atout majeur dans notre parcours académique et professionnel.

# CONTACTS

Enseignant encadrant: *M. David Lesaint, M. David Genest, M. Marc Legeay,*

*M.Vincent Barichard, M. Aurélien Simon, M. Corentin Behuet.*

Etudiant réalisant:

* BUI Thi Vi

[thivibui@etud.univ-angers.fr](mailto:thivibui@etud.univ-angers.fr)

* *ANANI Messan Joseph*

[*messanjoseph.anani@etud.univ-angers.fr*](mailto:messanjoseph.anani@etud.univ-angers.fr)

* *NDAYIZEYE Guy Keny*

[*guykeny.ndayizeye@etud.univ-angers.fr*](mailto:guykeny.ndayizeye@etud.univ-angers.fr)

# LIENS DE NOTRE PROJET

# SOMMAIRE

**INTRODUCTION**

Ce rapport présente le travail réalisé par notre équipe de trois stagiaires au sein du Laboratoire d'Étude et de Recherche en Informatique d'Angers (LERIA) durant notre période de stage du 24 mars 2025 à 20 mai 2025. Notre mission consistait à développer une interface web permettant de piloter un solveur d'emplois du temps universitaire développé par le laboratoire.

La génération d'emplois du temps universitaires est un problème complexe qui nécessite la prise en compte de nombreuses contraintes : disponibilité des enseignants et des salles, répartition cohérente des cours dans la semaine, respect des maquettes pédagogiques, etc. Le LERIA a développé un solveur performant pour traiter ce problème, mais son utilisation requiert actuellement des compétences techniques avancées.

Notre objectif principal était donc de créer une interface intuitive permettant aux utilisateurs non-spécialistes, comme les responsables de formation ou le personnel administratif, d'utiliser efficacement ce solveur. Cette interface devait permettre de sélectionner les données en entrée, configurer les paramètres du solveur, lancer les calculs et visualiser les résultats.

Dans ce rapport, nous présenterons d'abord le contexte et les objectifs détaillés du projet, puis nous décrirons notre démarche de conception et de réalisation. Nous exposerons ensuite les fonctionnalités développées, les difficultés rencontrées et les solutions apportées. Enfin, nous conclurons par un bilan de cette expérience et les perspectives d'évolution de l'application.

# PRÉSENTATION DE NOTRE PROJET

1. **Contexte :**

Le Laboratoire d'Étude et de Recherche en Informatique d'Angers (LERIA) travaille depuis plusieurs années sur le développement de solutions algorithmiques pour résoudre le problème complexe de la génération d'emplois du temps universitaires. Cette problématique constitue un défi majeur pour les établissements d'enseignement supérieur en raison des nombreuses contraintes à respecter : disponibilité des salles, des enseignants, cohérence des parcours étudiants, règles pédagogiques, etc.

Dans ce cadre, le LERIA a développé un solveur performant capable de calculer des solutions optimisées pour la création d'emplois du temps universitaires. Ce solveur, basé sur des techniques avancées de recherche opérationnelle et de programmation par contraintes, peut traiter des instances complexes comportant de nombreuses variables et contraintes.

Jusqu'à présent, l'utilisation de ce solveur nécessitait des compétences techniques spécifiques et une compréhension approfondie de son fonctionnement interne, ce qui limitait son accessibilité aux utilisateurs non-spécialistes comme les responsables de formation ou le personnel administratif.

1. **Objectifs du projet :**

Notre mission consiste à développer une interface utilisateur intuitive et fonctionnelle permettant de piloter le solveur d'emplois du temps. Cette interface doit servir d'intermédiaire entre les utilisateurs finaux et le système algorithmique complexe sous-jacent. Plus précisément, notre application doit permettre de :

1. **Définir le périmètre des données** : Permettre aux utilisateurs de rechercher, sélectionner et combiner des formations, des périodes et d'autres paramètres pour délimiter précisément le périmètre de l'emploi du temps à générer.
2. **Filtrer les règles et contraintes** : Offrir une interface permettant de gérer les différentes règles et contraintes du solveur, tant fondamentales que métier, en adaptant le modèle aux besoins spécifiques de l'utilisateur.
3. **Gérer les ressources** : Faciliter la sélection et le filtrage des cours, enseignants et salles à inclure dans le processus de génération, optimisant ainsi le temps de calcul et la pertinence des résultats.
4. **Configurer le solveur** : Proposer une interface intuitive pour ajuster les paramètres techniques du solveur (stratégies des variables, temps d'exécution, etc.) sans nécessiter de connaissances avancées, tout en offrant des options pour les utilisateurs experts.
5. **Visualiser et exploiter les résultats** : Suivre l'avancement du processus de résolution, afficher les solutions générées sous forme de statistiques et d'aperçus structurés, et permettre leur visualisation détaillée, leur sauvegarde et leur export.
6. **Architecture technique :**

Notre application s'appuie sur une architecture client-serveur où le frontend interagit avec deux API REST distinctes :

**API de données** : Cette API permet de récupérer les informations nécessaires à la constitution des instances (formations, périodes, enseignants, salles, cours, etc.).

**API du solveur** : Cette seconde API permet de configurer le solveur, de lancer les calculs et de récupérer les solutions générées ainsi que des statistiques sur le processus de résolution.

Le développement de notre application se concentre exclusivement sur la partie frontend, les API étant déjà développées et mises à notre disposition par l'équipe du LERIA. Conformément aux spécifications du projet, nous avons choisi de développer l'interface en utilisant les technologies web standards :

**HTML5** pour la structure de l'application

**CSS3** pour la mise en forme, avec l'utilisation du framework Bootstrap pour améliorer l'ergonomie et l'esthétique de l'interface

**JavaScript** pour la logique applicative et l'interaction avec les API

Cette approche technique nous permet de créer une application web accessible depuis n'importe quel navigateur moderne, sans nécessiter d'installation spécifique, tout en offrant une expérience utilisateur fluide et responsive.

# RÉALISATION DE PROJET

1. **Interface utilisateur :**

Pour le développement de l'interface utilisateur, nous avons adopté une approche centrée sur l'expérience utilisateur, visant à rendre l'utilisation du solveur accessible même aux personnes sans connaissances techniques approfondies. Nous avons opté pour une structure claire et intuitive, organisée autour de cinq sections principales qui suivent le flux de travail naturel des utilisateurs.

### **1 . Technologies utilisées :**

Conformément aux spécifications du projet, nous avons développé l'interface en utilisant les technologies web standards :

* **HTML5** pour la structure de base de l'application
* **CSS3** pour la mise en forme et les animations
* **JavaScript** pour l'interactivité et la communication avec les API
* **Bootstrap 5** comme framework CSS pour assurer une interface responsive et esthétique

Nous avons choisi Bootstrap 5 comme framework CSS principal pour sa robustesse et sa flexibilité. Ce framework nous a permis d'implémenter rapidement une interface responsive et cohérente grâce à ses composants prédéfinis comme les cartes, les onglets, les formulaires et les alertes.

Nous avons exploité le système de grille de Bootstrap pour organiser le contenu, ses composants de navigation pour structurer l'application, et ses utilitaires pour gérer les espacements et les alignements. La personnalisation de l'apparence a été réalisée via plusieurs fichiers CSS dédiés (*style.css, tabs\_navigation.css, config\_tab.css*), où nous avons adapté les couleurs et les styles aux besoins spécifiques du projet tout en conservant la cohérence visuelle globale.

### **2. Structure générale de l'interface**

L'interface est organisée en cinq sections principales, accessibles via une barre d'onglets en haut de la page :

1. **Définition du périmètre** : Permet de rechercher, filtrer et sélectionner les instances sur lesquelles travailler.
2. **Filtrage des règles** : Permet de gérer les contraintes fondamentales et métier du solveur.
3. **Filtrage des ressources** : Permet de sélectionner les cours, enseignants et salles à utiliser.
4. **Configuration du solveur** : Offre différentes options pour paramétrer le fonctionnement du solveur.
5. **Solutions** : Affiche les résultats générés et les statistiques associées.

Cette structure séquentielle guide l'utilisateur à travers les différentes étapes du processus de génération d'emploi du temps, depuis la sélection des données jusqu'à la visualisation des résultats.

**2.1 Section de définition du périmètre**

La première section de notre interface permet aux utilisateurs de rechercher et sélectionner les formations et périodes sur lesquelles travailler. Cette section a été conçue pour être intuitive et offrir une expérience utilisateur fluide.

Nous avons implémenté une barre de recherche avec autocomplétion qui affiche des suggestions en temps réel à mesure que l'utilisateur tape. Cette fonctionnalité permet de trouver rapidement une formation spécifique sans parcourir toute la liste disponible.

En complément, un menu déroulant permet de filtrer les instances par type de formation. Une fois une formation sélectionnée, l'interface affiche automatiquement les périodes associées (semestres, années) que l'utilisateur peut sélectionner individuellement ou globalement grâce au bouton "Tout sélectionner".

Un résumé des choix effectués s'affiche ensuite dans une zone dédiée avec la possibilité de modifier ou supprimer des sélections existantes. Cette approche permet à l'utilisateur de visualiser clairement ses choix avant de lancer la recherche des données. Un bouton "Visualiser le XML" permet alors d'afficher le fichier XML correspondant à la sélection.

Le code JavaScript associé à cette section utilise des fonctions asynchrones pour communiquer avec l'API et récupérer les données nécessaires :

* fetchFormations() pour charger la liste des formations disponibles
* fetchPeriods(formationId) pour obtenir les périodes d'une formation spécifique
* updateSummaryDisplay() pour afficher le résumé des sélections
* fetchPerimetersData() pour récupérer les données complètes du périmètre
* generateCompleteTimetabling() pour générer le XML correspondant

***Figure 1 :***

**2.2 Section de configuration du solveur :**

La deuxième section permet aux utilisateurs de gérer les règles et contraintes du solveur. Compte tenu du nombre important d'options disponibles, nous avons organisé cette section en trois sous-onglets pour une navigation claire et intuitive :

1. **Contraintes fondamentales** : Cette partie permet d'activer ou désactiver les contraintes de base nécessaires au bon fonctionnement du solveur comme disjunctive\_teacher, disjunctive\_group, disjunctive\_room, teacher\_service, etc. Ces contraintes sont essentielles pour assurer la cohérence des emplois du temps générés.
2. **Contraintes métier** : Cet onglet offre la possibilité de gérer les contraintes spécifiques au domaine de l'emploi du temps universitaire, comme sameRooms, allowedPeriod, weekly, forbiddenPeriod, etc. Un champ de recherche dédié facilite la navigation parmi ces nombreuses contraintes, permettant aux utilisateurs de trouver rapidement des contraintes spécifiques.
3. **Désactivation de règles** : Ce dernier sous-onglet propose trois modes différents de désactivation de règles :
   * **Liste complète** : Affiche toutes les règles dans un tableau interactif avec possibilité de filtrage par texte et tri par colonne
   * **Filtrage avancé** : Permet de filtrer les règles selon divers critères (type de contrainte, nom, label, générateur, etc.)
   * **Saisie manuelle** : Permet de désactiver des règles en entrant directement leurs numéros, avec support des plages (ex: 1-5,8,10-12)

Pour la liste complète, nous avons implémenté un tableau interactif avec des fonctionnalités de tri pour chaque colonne (ID, label, générateur, filtre, contrainte, type, statut), ainsi qu'une barre de recherche pour filtrer rapidement les règles. Un compteur indique le nombre de règles actives par rapport au total.

Le code JavaScript associé à cette section inclut des fonctions comme :

* renderRulesDetails(data) pour afficher les règles dans le tableau
* sortRulesTable(field, direction) pour le tri des colonnes
* applyAdvancedFilters() pour le filtrage multicritère
* parseRuleInput(input) pour la saisie manuelle des règles à désactiver

***Figure 2***

#### **2.3 Section de filtrage des ressources**

La troisième section permet de gérer les ressources utilisées par le solveur, réparties en trois catégories accessibles par des onglets :

1. **Cours** : Affiche la liste des cours disponibles avec leur identifiant, label et nombre de parties. Chaque cours peut être activé ou désactivé individuellement.
2. **Enseignants** : Présente les enseignants avec leurs informations (nom, département, volume horaire) en permettant leur inclusion ou exclusion de la génération d'emploi du temps.
3. **Salles** : Liste les salles disponibles avec leur capacité, type et bâtiment, offrant la possibilité de filtrer les ressources spatiales à considérer.

Chaque catégorie dispose de fonctionnalités similaires :

* Barre de recherche pour filtrer les ressources par texte
* Boutons "Tout sélectionner" et "Tout désélectionner" pour une gestion en masse
* Tableau interactif avec tri pour chaque colonne (ID, nom, capacité, etc.)
* Cases à cocher pour activer/désactiver individuellement les ressources
* Indication visuelle (badge coloré et opacité) de l'état actif/inactif
* Compteur de ressources actives par rapport au total

Le code JavaScript de cette section (filter\_ressource.js) comprend des fonctions comme:

* *renderCoursesTable(data), renderTeachersTable(data)* et *renderRoomsTable(data)* pour l'affichage des tableaux
* *setupCoursesSearch(), setupTeachersSearch() et setupRoomsSearch()* pour la recherche textuelle
* *sortTable(tableId, field, direction*) pour le tri générique des tableaux
* *updateResourceCounter(type, activeCount, totalCount)* pour mettre à jour les compteurs
* *getFilteredResourcesData()* pour extraire les données filtrées à envoyer au solveur

#### **2.4 Section de configuration du solveur**

La quatrième section permet de paramétrer finement le solveur selon les besoins de l'utilisateur. Elle est divisée en trois sous-onglets :

1. **Paramètres généraux** : Permet de définir les configurations de base, notamment le temps maximum d'exécution du solveur en secondes. Cette limite temporelle est cruciale pour éviter que le solveur ne s'exécute indéfiniment sur des problèmes complexes.
2. **Stratégies** : Offre un contrôle avancé sur la manière dont le solveur traite les différentes variables du problème. Les utilisateurs peuvent ajouter différentes stratégies pour les salles, les enseignants et les créneaux horaires, en définissant pour chacune :
   * Type de variable (x\_room, x\_teachers, x\_slot, etc.)
   * Type de sélecteur (rank, label, id)
   * Valeur du sélecteur (avec validation du format)
   * Stratégie d'ordre (first\_fail, input\_order, etc.)
   * Stratégie de parcours (indomain\_min, indomain\_random, etc.)
3. **Mode avancé** : Permet aux utilisateurs experts de modifier directement la configuration JSON du solveur, offrant ainsi un accès complet à toutes les options disponibles. Un avertissement prévient l'utilisateur des risques potentiels de cette approche.

Des boutons "Charger configuration" et "Enregistrer configuration" permettent respectivement d'importer une configuration existante et d'exporter la configuration actuelle au format JSON, facilitant ainsi la réutilisation et le partage des paramètres.

Le bouton "Lancer le solveur" déclenche l'exécution avec les paramètres définis et redirige automatiquement vers la section des résultats. Le code JavaScript (app.js et config\_Tab.js) gère la génération du JSON de configuration et les interactions avec l'API du solveur.

#### **2.5 Section des solutions**

La cinquième section présente les résultats générés par le solveur de manière structurée et informative. Elle est initialement masquée et s'affiche après le lancement du solveur.

La partie supérieure affiche le statut actuel du solveur (inactif, en cours, terminé, échec) accompagné du temps écoulé et d'une barre de progression visuelle. Ces indicateurs permettent à l'utilisateur de suivre en temps réel l'avancement du processus de résolution.

Une fois le calcul terminé, un résumé statistique de la solution s'affiche, présentant des métriques clés comme le nombre de séances planifiées, de groupes d'étudiants, de classes, d'enseignants assignés et de salles utilisées. Ces statistiques offrent une vue d'ensemble rapide de la solution générée.

Un aperçu détaillé de l'emploi du temps est ensuite présenté, listant les différentes séances programmées avec leurs informations essentielles : classe, rang, jour, créneau, salle et enseignant. Si présent, un aperçu des groupes d'étudiants est également affiché.

Trois boutons d'action sont proposés :

* "Enregistrer la solution" pour une utilisation ultérieure
* "Exporter" pour générer un fichier JSON contenant la solution complète
* "Voir l'emploi du temps complet" qui ouvre une application de visualisation externe

Un bouton "Nouvelle résolution" permet de revenir à la première étape pour réaliser une nouvelle génération d'emploi du temps.

Le fichier resultat.js contient les fonctions associées à cette section :

* *attendreEtat(etatVise)* pour surveiller l'état du solveur
* *chargerResultatSolveur()* pour récupérer et afficher la solution
* *afficherStatsSolution(solution)* pour générer les statistiques
* *afficherEmploiDuTemps(solution)* pour créer l'aperçu visuel
* *exporterSolutionJSON(data)* pour l'export des données

Cette organisation modulaire de l'interface, avec une navigation fluide entre les différentes étapes du processus, permet aux utilisateurs de tous niveaux techniques de configurer efficacement le solveur d'emplois du temps et d'obtenir des résultats exploitables.

***Figure 3 :***

**II . Interaction avec les API :**

**III . Fonctionnalités implémentées :**

Nous avons développé plusieurs fonctionnalités clés pour offrir une expérience utilisateur complète:

1. **Recherche et sélection avancées** :
   * Autocomplétion dans la barre de recherche des formations
   * Sélection multiple de périodes avec "Tout sélectionner"
   * Résumé visuel des sélections avec possibilité d'édition et suppression
2. **Gestion des contraintes et règles** :
   * Activation/désactivation des contraintes fondamentales et métier
   * Interface de désactivation de règles avec trois modes d'accès (liste, filtrage, saisie manuelle)
   * Recherche et filtrage avancé des règles avec plusieurs critères
   * Compteur de règles actives/inactives
3. **Gestion des ressources** :
   * Filtrage et tri des cours, enseignants et salles
   * Activation/désactivation individuelle ou globale
   * Recherche textuelle dans chaque catégorie
   * Indication visuelle de l'état actif/inactif
4. **Configuration avancée du solveur** :
   * Interface intuitive pour les stratégies de variables
   * Validation des entrées utilisateur
   * Export/import de configurations
   * Mode avancé avec édition directe du JSON
5. **Suivi et visualisation des résultats** :
   * Barre de progression et temps écoulé
   * Statistiques sur la solution générée
   * Aperçu de l'emploi du temps
   * Export de la solution en JSON
6. **Navigation et expérience utilisateur** :
   * Interface à onglets avec navigation séquentielle (boutons Précédent/Suivant)
   * Sauvegarde de l'état des onglets dans l'URL
   * Animations et transitions fluides
   * Responsive design adapté aux différents formats d'écran
7. **Intégration avec les API** :
   * Génération dynamique du XML à partir des sélections
   * Communication asynchrone avec le serveur
   * Gestion des erreurs avec messages utilisateur
   * Polling pour suivre l'état du solveur
8. **Difficultés rencontrées et solutions apportées :**

Durant le développement de l'application, nous avons rencontré plusieurs défis techniques que nous avons surmontés :

1. **Gestion des dépendances entre sections** : La configuration du solveur dépendait fortement des données sélectionnées dans les sections précédentes. Nous avons implémenté un système d'événements personnalisés pour notifier les différentes parties de l'application des changements d'état.
2. **Traitement de grandes quantités de données** : Les règles et ressources pouvaient être très nombreuses, risquant de ralentir l'interface. Nous avons optimisé les performances par :
   * L'utilisation de tableaux paginés
   * Des mécanismes de recherche et filtrage efficaces
   * Le chargement à la demande des données détaillées
3. **Complexité de la configuration du solveur** : Pour faciliter la gestion des nombreux paramètres, nous avons :
   * Créé une interface graphique intuitive pour les stratégies de variables
   * Ajouté des validations en temps réel des entrées utilisateur
   * Intégré une aide contextuelle pour guider les utilisateurs
4. .

1. **Résultats obtenus :**
2. **Conclusion :**
3. **Annexes**