INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

Département STPI

TIP 2025 - Rapport

Plateforme tutorat

Gestion du tutorat en STPI

Étudiant

Anonymisé

18 septembre 2025

Table des matières

Se référer au cahier des charges également fourni avec ce rapport afin de consulter les attendus de ce projet, les natures et types de tâches, les compétences développées et le lien des tâches et compétences avec la formation. Ce rapport ne se concentre que sur le projet rendu.

1 Fonctionnement global

Nous parlerons de campagne de tutorat afin de distinguer les différentes années et les différents semestres de tutorat. Cette approche a été choisie pour ajouter de la flexibilité en cas de modification du fonctionnement du tutorat en STPI. Dans le fonctionnement actuel, il y aura donc normalement 2 campagnes de tutorat par année scolaire.

1. Les utilisateurs sont importés par insertion SQL dans la base de données, en spécifiant les groupes et statuts (tuteur, tutoré, tuteur et tutoré). Pas d'outil d'import intuitif n'a été développé, le format des données à importer étant inconnu/variable
2. Une campagne de tutorat est créée pour l'année en cours et le semestre souhaité depuis l'interface d'administration : les utilisateurs correspondants sont alors associés à cette campagne et s'affichent sur leurs tableaux de bord
3. Selon le statut des inscriptions, les utilisateurs peuvent s'inscrire à cette campagne : les tuteurs renseignent les matières préférées avec un nombre de tuteur souhaité, les tutorés choisissent les matières dans lesquelles ils ont besoin d'aide
4. Les utilisateurs remplissent leurs disponibilités en dehors des cours qu'ils suivent durant le semestre. Ces disponibilités seront utilisées pour l'affectation automatique
5. Au moment voulu, les affectations peuvent être générées automatiquement depuis l'interface. Il est toujours possible de les modifier manuellement, avant et après affectation automatique. Les inscriptions après affectation sont toujours possibles selon le statut des inscriptions de la campagne. L'ajout / la suppression manuelle de tuteurs / tutorés est également possible.
6. Les tuteurs s'affichent sur les espaces tutorés et les tutorés s'affichent sur les espaces tuteurs, où les tuteurs peuvent désormais saisir les contenus des séances de tutorat sur les pages dédiées au tutorat. Les tutorés peuvent en parallèle saisir les heures effectuées : la saisie des séances est informative et ne conditionne pas la saisie des heures. En cas de suppression/déplacement d'un tutoré, les heures sont conservées.
7. Le responsable du tutorat peut consulter à tout moment le volume horaire total réalisé par chaque tuteur et tutoré depuis l'espace administrateur. Il peut également saisir des heures manuellement.
8. La campagne de tutorat est terminée et le responsable du tutorat dispose de toutes les informations : contenus des séances, heures effectuées par tuteur.

2 Structuration du projet

2.1 Côté client (front-end)

Le client de notre application est une interface web permettant aux utilisateurs d'interagir avec un serveur via un navigateur. Dans notre cas, j'ai utilisé le framework Nuxt, basé sur Vue.js, qui facilite la création d'interfaces utilisateur dynamiques et réactives.

Lors de la phase de « build », Nuxt génère automatiquement des fichiers HTML, CSS et JavaScript optimisés, à partir du code source. Ces fichiers sont ensuite utilisés pour afficher l'interface utilisateur, avec tous les comportements interactifs que nous avons développés. L'intérêt d'une telle structuration est la décorrélation totale de la logique interface de la logique serveur. On obtient alors une navigation beaucoup plus fluide et intuitive, tous les calculs et les rendus étant effectués directement par le client ! Nuxt offre aussi la possibilité d'implémenter le rendu côté serveur (SSR) ou le rendu statique (SSG). La réalisation de ce client a mobilisé plusieurs compétences clés autour de l'écosystème Vue : la réactivité du DOM, les composants, `props`, événements, `watch`, stores (Vuex / Pinia), DOM virtuel, cycle de vie des composants (`onMounted`, `onUnmounted` ...). Le client réalise ce qu'on appelle des « appels à l'API », l'API étant notre serveur. L'accès aux données de l'API est structurées par des routes et des méthodes d'appels différentes.

2.2 Côté serveur (back-end)

En parlant d'appels à l'API, j'ai convenu dans mon cahier des charges l'utilisation d'une API dite « REST » (REpresentational State Transfer). Seules les routes et les méthodes effectivement utiles au client ont été implémentées, se référer à l'annexe (table 3). `GET, POST, PATCH, DELETE` sont des méthodes HTTP ayant été utilisées sur cette API. Les `:campaignId` sont des paramètres dynamiques. Toutes ces routes permettent donc d'exploiter les données et d'intéragir avec la base de données. Grâce à un ORM (Mapping Objet-Relationnel), les requêtes SQL sont directement écrites par gorm, utilisé dans ce projet. L'intérêt d'un ORM est d'utiliser une représentation concrète des données, en utilisant des `struct` (structures de données). Les migrations sont gérées par l'ORM.

2.2.1 Sessions et sécurité

Ce projet repose sur l'authentification par sessions par cookies, un point cybersécurité non négligeable. Ces cookies `httponly` et `secure` ne peuvent être récupérés par aucun script JS malveillant étant donné qu'ils ne sont envoyés que par le navigateur. Les routes sont protégées par un `middleware` vérifiant la session de l'utilisateur. Un `middleware` est un code exécuté avant et éventuellement après la requête : ici, lorsqu'un utilisateur envoie une requête sur une ressource protégée, le middleware stoppe la requête et la propagation dans le routeur s'il n'est pas authentifié. Un autre middleware, `error`, récupère les éventuelles erreurs durant l'exécution du code afin d'envoyer à l'utilisateur une réponse personnalisée présentant correctement l'erreur rencontrée.

2.2.2 Authentification

Si ce projet est voué à être mis en place, une authentification via le CAS de l'INSA Rouen serait idéal afin de récupérer les groupes des étudiants. En l'absence de cette possibilité, une authentification via e-mail et jeton temporaire a été mis en place. En cas de refus de la DSI, il sera alors toujours possible de mettre en place ce projet en important les données des groupes directement dans l'outil.

3 Disponibilité des étudiants

3.1 Récupération des agendas

Dans l'optique de pouvoir optimiser les disponibilités entre les tuteurs et les tutorés pour faciliter l'organisation des séances de tutorat, il a fallu trouver un moyen de les récupérer. Plusieurs sources étaient possibles, mais la seule source ouverte, fiable, mise à jour était malheureusement... https://agendas.insa-rouen.fr, une vieille plateforme proposant un seul format pour récupérer un flux agenda : le RSS, signifiant Really Simple Syndication exploitant le format XML (Extensible Markup Language). Ne tentons pas de comprendre son interface mais plutôt le contenu du flux RSS que l'on exploitera :

```
<item>
<guid
↪  isPermaLink="false">https://agendas.insa-rouen.fr:443/day.php</guid>
<title>Mar Mai 20 2025: STPI22-M7-TD-3</title>
<ev:startdate>2025-05-20T15:00:00</ev:startdate>
<ev:enddate>2025-05-20T16:30:00</ev:enddate>
<link>https://agendas.insa-rouen.fr:443/day.php?\texttt{GET}date=20250520</link>
<description>Mar Mai 20 2025 15:00:
↪  <br/><br/>STPI22-M7-TD-03<br/>ROUXELIN Nathan<br/>STPI2<br/>(Exporté
↪  le:20/05/2025 21:30)<br/></description>
<ev:location>DU-B-RJ-01</ev:location>
</item>
```

On obtient alors plusieurs `items` dont on extrait les informations telles que : le nom du cours (`STPI22-M7-TD-3`), la date et le créneau, la salle (`DU-B-RJ-01`) et les groupes (`STPI22-M7-TD-03`).

Avec l'aide d'un RSS parser permettant de faciliter la lecture de chaque `item`, beaucoup de travail a été réalisé afin d'en extraire les données et de les transformer afin de les faire correspondre aux structures de données utilisées dans le projet.

Une mise à jour de cet ancien outil proposé par l'INSA vers une alternative moderne aurait enfin pu permettre de proposer d'autres flux plus modernes tels que le JSON où l'accès aux données aurait été immédiat et aurait évité quelques heures de travail à ce sujet.

3.2 Saisie des disponibilités

Les étudiants (tuteurs et tutorés) sont invités à saisir leurs disponibilités par le biais de l'interface utilisateur. Cette interface affiche les créneaux déjà occupés par des cours sur le semestre et laisse le choix à l'utilisateur de marquer sa disponibilité sur les autres créneaux. Au final, les données stockées en base de données sont représentées au format suivant :

```
{"1": [3,13,0,0,13,10,0], "2": [11,15,1,0,14,9,0], "3": [8,9,14,0,0,12,0]}
```

Chaque couple clé/valeur représente l'ordinal du jour/disponibilités. L'ordinal de `lundi` est bien égal à `1`, Golang ayant choisi de démarrer ses semaines au dimanche de façon américaine. `0` correspond à l'ordinal de `dimanche`. Le tableau associé à chaque clé représente chaque statut de chaque créneau. Pour un créneau donné, on attribue une valeur

c, selon si l'étudiant est indisponible ($c = -1$), disponible ($c = 0$) ou s'il a cours sur ce créneau, connaissant son nombre n de cours sur le semestre ($c = n > 0$). Cette valeur c est donc comprise dans l'intervalle $[-1, +\infty[$.

3.3 Optimisation des disponibilités

Pour mettre en oeuvre l'algorithme d'optimisation des disponibilités, il a fallu déterminer un moyen de quantifier et de représenter la compatibilité entre deux agendas (celui du tuteur et du tutoré). C'est le rôle de la fonction `AvailabilityScore` retournant un `float64`, un nombre réel avec une grande précision. L'idée est que plus cette valeur est grande, plus les deux agendas sont compatibles. Cette fonction va alors parcourir chaque créneau un à un de chaque agenda.

3.3.1 Première approche

Voici mon premier raisonnement : plus cette valeur c est grande, moins l'étudiant est disponible, ce qui est l'inverse de ce que doit renvoyer la fonction. Alors on inverse, en ajoutant 1 au dénominateur pour rester dans le domaine des réels positifs : $V(c) = \frac{1}{1+c} \in]0, 1]$ si $c \neq -1$ sinon $V(c) = 0$. Comme $V(c) \in [0, 1]$, on fait le produit entre le score du créneau du premier agenda avec le score du créneau du second agenda, qui est également compris entre $[0, 1]$. Maintenant que nous avons un score pour chaque créneau global, il nous reste à faire la somme de ces scores.

Val. créneau A	Val. créneau B	Score créneau
$c = -1$, $V(c) = 0$	$c = 1$, $V(c) = \frac{1}{2}$	0
$c = 11$, $V(c) = \frac{1}{12}$	$c = 23$, $V(c) = \frac{1}{24}$	$\frac{1}{288}$
$c = 3$, $V(c) = \frac{1}{4}$	$c = 7$, $V(c) = \frac{1}{12}$	$\frac{1}{48}$
$c = 2$, $V(c) = \frac{1}{3}$	$c = 1$, $V(c) = \frac{1}{2}$	$\frac{1}{6}$
$c = 0$, $V(c) = 1$	$c = 0$, $V(c) = 1$	1
0		

TABLE 1 – Première approche de $V(c)$

Néanmoins, on constate que cette approche n'est pas correcte. Si le premier étudiant a 2 cours sur l'entièreté du semestre et que l'autre étudiant n'en a qu'un ponctuellement, le score est de $1/6 = 0.167$, ce qui est trop peu.

3.3.2 Meilleure approche

Je suis donc parti à la recherche d'une fonction qui vaudrait 1 lorsque $c = 0$, pouvant décroître lorsque $c \to +\infty$: c'est la fonction exponentielle. Après multiples tentatives sur GeoGebra, j'ai finalement opté pour cette fonction : $V(c) = e^{-\frac{c}{8}}$ si $c \geq 0$ sinon $V(c) = 0$, le dénominateur 8 ayant été choisi arbitrairement pour avoir des résultats cohérents.

Les résultats sont meilleurs dans le sens où lorsqu'un étudiant n'a que 2 cours et l'autre n'a que 4 cours sur le semestre, on obtient un score proche de 0.5. De plus, la fonction décroît plus vite lorsque le nombre de cours augmente.

Val. créneau A	Val. créneau B	Score créneau
$c = -1,\ V(c) = 0$	$c = 1,\ V(c) = e^{\frac{1}{3}}$	0
$c = 8,\ V(c) = e^{-1} = 0.37$	$c = 16,\ V(c) = e^{-2} = 0.16$	0.06
$c = 2,\ V(c) = e^{-\frac{1}{4}} = 0.78$	$c = 4,\ V(c) = e^{-\frac{1}{2}} = 0.61$	0.48
$c = 0,\ V(c) = 1$	$c = 0,\ V(c) = 1$	1
0		

TABLE 2 – Meilleure approche de $V(c)$

4 Moteur d'affectation

Grâce à la saisie des disponibilités des étudiants depuis leur espace personnel, on possède désormais toutes les données afin d'essayer de maximiser les créneaux en commun entre les tuteurs et les tutorés.

4.1 Algorithme glouton

Ma première approche à ce problème était instinctivement un algorithme glouton (greedy) : faire le choix localement optimal à chaque étape. Dans notre contexte : pour chaque tuteur, choisir le tutoré le plus compatible au sens de l'emploi du temps avec le tuteur. C'est la méthode qui me semblait idéale, mais en faisant le lien avec mes cours de I4, je me suis questionné sur la pertinence de cet algorithme ici. Prenons un exemple avec deux tuteurs A, B et deux tutorés C et D. Chaque couple de tuteur/tutoré dispose donc d'un score calculé sur la base de leurs agendas.

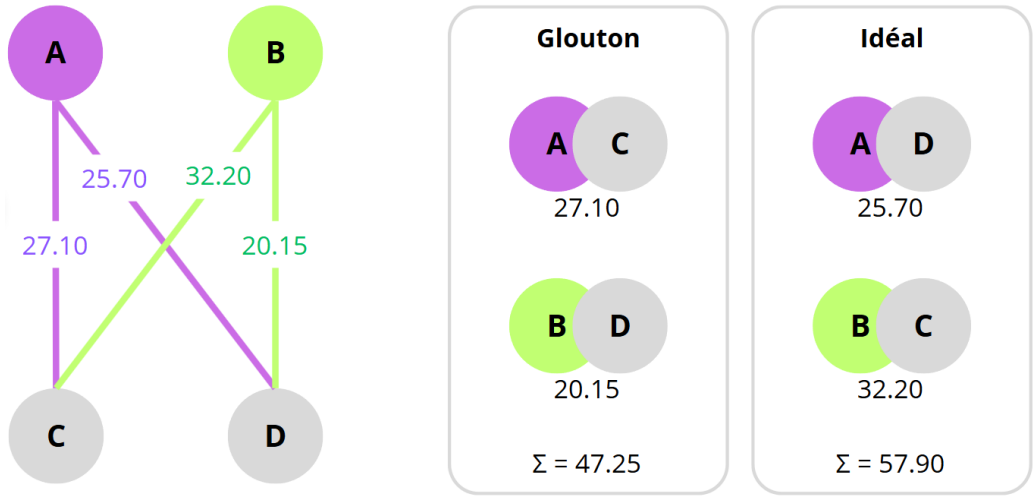


FIGURE 1 – Représentation du problème algorithmique de l'affectation

L'algorithme glouton consiste à choisir le meilleur tutoré localement pour le tuteur A. On a un tutoré C avec un score de 27.10 et un tutoré D à 25.70. On affecte le tutoré C au tuteur A. Passons au tuteur B : il ne reste plus que le tutoré D, avec un score de 20.15. Pourtant, le tutoré C possède un score de 32.20 avec le tuteur B ! Quelque chose ne va pas. Idéalement, on cherche à maximiser la somme de tous les scores de disponibilité comme présenté dans l'encart idéal. Il faudrait trouver un algorithme qui permette de faire un compromis en affectant au tuteur A un tutoré avec un score un peu plus faible pour permettre au tuteur B d'avoir un bien meilleur score avec le tutoré C.

4.2 Problème des appariements stables

Heureusement, il s'agit d'un problème connu possédant une solution. Dans la littérature française, il est souvent évoqué comme le « problème des mariages stables », mais nous parlerons plutôt d'appariements stables. Ce problème consiste à rechercher un appariement stable entre deux ensembles d'éléments de taille égale, compte tenu d'un ordre de préférences pour chaque élément. Un appariement est stable lorsqu'il n'existe aucune paire (A, B) qui se préfère mutuellement à leur partenaire actuel dans la paire.

4.3 Algorithme de Gale et Shapley

L'algorithme résolvant ce problème s'appelle l'« algorithme de Gale et Shapley », également utilisé par Parcoursup et le SCEI (Service de Concours Ecoles d'Ingénieurs). L'algorithme de Gale et Shapley utilisé dans ce projet provient d'un dépôt open-source en Golang : https://github.com/yutohub/matching. L'idée est donc d'exécuter cet algorithme pour chaque matière. En revanche, il existe encore un problème : cet algorithme cherche à associer de façon bijective les éléments de A avec ceux de B. Soit $n_{tuteurs}$ le nombre de places proposés par les tuteurs (et non le nombre de tuteurs en eux-mêmes) et $n_{tutores}$ le nombre de demandes pour du tutorat en cette matière.

Si $n_{tutores} > n_{tuteurs}$, alors il est **impossible** de réaliser des affectations en garantissant à tous les tutorés qu'ils seront affectés à un tuteur. On invite le responsable du tutorat à augmenter les quotas de certains tuteurs pour correspondre à la demande.

Si $n_{tuteurs} = n_{tutores}$, **aucun problème**. Tout le monde sera affecté correctement.

Si $n_{tuteurs} > n_{tutores}$, alors on doit réfléchir à comment fonctionner. Ma première approche était de sélectionner autant de créneaux que de $n_{tutores}$ de manière équitable chez chacun des tuteurs en fonction de leur quota. Mais en reprenant l'exemple de la figure 1, si nous avions qu'un tutoré C, alors j'aurai conservé un seul créneau du tuteur A (étant le premier tuteur). Problème identique au précédent, il aurait en réalité fallu sélectionner le tuteur B, possédant un meilleur score. La réponse à ce problème a donc été de créer des **« tutorés factices »**, qui ne correspondent à aucun tutoré réel et possédant simplement un score de `-1`. De ce fait, on s'assure que toutes les possibilités seront explorées et que le cas précédent ne sera pas choisi : un tutoré factice sera assigné au tuteur A.

D'un point de vue programmation, cette approche a été réalisée avec l'aide de pointeurs. Pour un tuteur factice, la valeur du tuteur associée pointe vers la valeur `NIL`, nulle. Les tuteurs réels pointent vers leur valeur réelle. Grâce à la gestion de ces différents cas, on arrive à créer une solution idéale décrite à la figure 1. Attention, ce n'est pas la meilleure solution, seulement une solution stable.

5 Réalisation du projet

5.1 Volume horaire

Au début du projet, j'ai installé une extension permettant de calculer le temps passé sur le projet ainsi que différentes statistiques sur les différents fichiers sur lesquels on a passé le plus de temps. Ce calcul ne se base que sur le temps d'édition actif. Il ne prend pas en compte les recherches sur certains algorithmes, la gestion de la base de données et toutes les recherches sur internet afin de résoudre certains problèmes rencontrés. Au total, le temps passé à éditer du code s'élève à **49h33**, ce qui correspond effectivement

au volume horaire prévisionnel de 49h comme présenté dans le cahier des charges. Je pense évaluer à quelques heures supplémentaires le temps de recherche, de réflexion sur les différents algorithmes et la rédaction de ce rapport.

Toutes les fonctionnalités obligatoires ont bien été implémentées. Ayant atteint la barrière horaire des 50 heures et pour rester dans un temps de projet raisonnable dans le cadre d'un TIP comme conseillé, les fonctionnalités optionnelles n'ont pas été implémentées mais pourront l'être si ce projet venait à être concrétisé.

Voici un extrait du tableau rendant compte du temps passé par fonctionnalité du projet. On remarque que certains fichiers ont été renommés durant le développement, le temps de travail d'un élément pouvant donc être réparti entre plusieurs fichiers. On remarque également que dans cet extrait de données, on retrouve majoritairement des fichiers liés au développement du client : il s'agissait de l'élément le plus chronophage.

15 premiers fichiers triés par durée de travail	**Durée**
/routes/admin/campaign/generate_assignments.go	3h 47m 20s
/main.go	3h 8m 32s
/client/pages/campaign/[campaignId]/admin/assignments.vue	2h 47m 41s
/routes/admin/generate_assignments.go	2h 20m 26s
/client/pages/tutee/register.vue	1h 59m 13s
/client/components/Navbar.vue	1h 54m 29s
/client/pages/tutee/register/[campaignId].vue	1h 40m 47s
/client/pages/tutoring/[tutorSubjectId]/index.vue	1h 37m 12s
/client/pages/tutee/index.vue	1h 32m 4s
/client/pages/tutee.vue	1h 19m 17s
/client/pages/campaign/[campaignId]/admin/overview.vue	1h 6m 11s
/routes/campaign/registration/`POST`_register.go	0h 59m 16s
/core/insaAgenda.go	0h 54m 55s
/client/pages/campaign/[campaignId]/availabilities.vue	0h 48m 8s
/client/pages/tutor/index.vue	0h 41m 50s

5.2 Utilisation d'IA générative

Souhaitant être transparent concernant l'utilisation de l'IA, il me semble important de préciser dans quelle mesure j'ai eu recours à l'intelligence artificielle générative dans le cadre de ce projet. Ma vision personnelle de cet outil est qu'il s'agit d'un accélérateur de productivité dès lorsqu'il est bien utilisé. Lorsque l'on maîtrise déjà la tâche à accomplir, l'IA peut réduire le temps passé sur des aspects répétitifs (ou parfois sans intérêt pédagogique dans notre cas).

Une assistance via l'IA a été utilisée pour certains éléments du front-end, notamment pour la génération de composants ou de structures de pages répétitives qui n'auraient pas enrichi significativement mon apprentissage à mon sens, maîtrisant déjà ce stack (Nuxt, Vue, TailwindCSS) dans de nombreux projets personnels. Exemple typique : après avoir développé moi-même un composant de saisie des heures de tutorat (avec le modal), demander à transformer ce code pour renommer les variables et ajouter un champ « contenu » pour les contenus des séances.

Également, je considère que l'IA ne doit pas remplacer notre propre réflexion : lorsque l'on ne sait ni où aller ni comment structurer la solution, l'IA peut proposer des solutions inadaptées à notre environnement de code, ne disposant pas de tout le contexte ni du

jugement pour considérer qu'elle n'est pas adaptée.

Aucune IA n'a été utilisée pour le développement back-end en Golang, qui constitue le cœur algorithmique du projet. Cette partie a été intégralement conçue et développée moi-même avec l'aide d'algorithmes et de recherches présentées précédemment afin de que ce projet puisse apporter un apprentissage des concepts de programmation et de résolution algorithmique. L'IA a été utilisé en soutien pour des tâches basiques et non pas pour raisonner à ma place.

5.3 Publication open-source

Comme précisé dans le cahier des charges, ce projet fait l'objet d'une publication en open source. Ce projet est déjà disponible sur un dépôt personnel sur GitHub : https://github.com/Romitou/INSATutorat. Tous les codes sont donc ouverts à tous et sont consultables par le public.

6 Ressenti personnel

Comme expliqué dans le cahier des charges, ce projet a été initialement motivé dès l'été 2024 sans avoir eu connaissance du TIP. Dès lorsque j'en ai pris connaissance, j'ai directement pensé à ce projet qui aurait pu trouver sa place en répondant aux attendus, notamment en terme de service aux personnels et aux étudiants.

Sa réalisation n'a pas été simple, elle a malgré tout occupé beaucoup de temps personnel. Réussir à trouver du temps pour ce projet a été compliqué, comme expliqué dans les difficultés prévisionnelles du cahier des charges. Beaucoup de projets associatifs, tous aussi liés à l'INSA, ont occupé une partie majeure de mon temps.

Malgré tout, je suis personnellement satisfait du résultat et pense que ce projet peut avoir une utilité réelle. Dans une optique de mise en place, je suis prêt à ajouter les fonctionnalités manquantes et en assurer sa maintenance si ce projet est jugé utile et puisse répondre à de réelles problématiques. Dans le cas contraire, la réalisation de ce projet restera tout de même une bonne expérience.

7 Installation

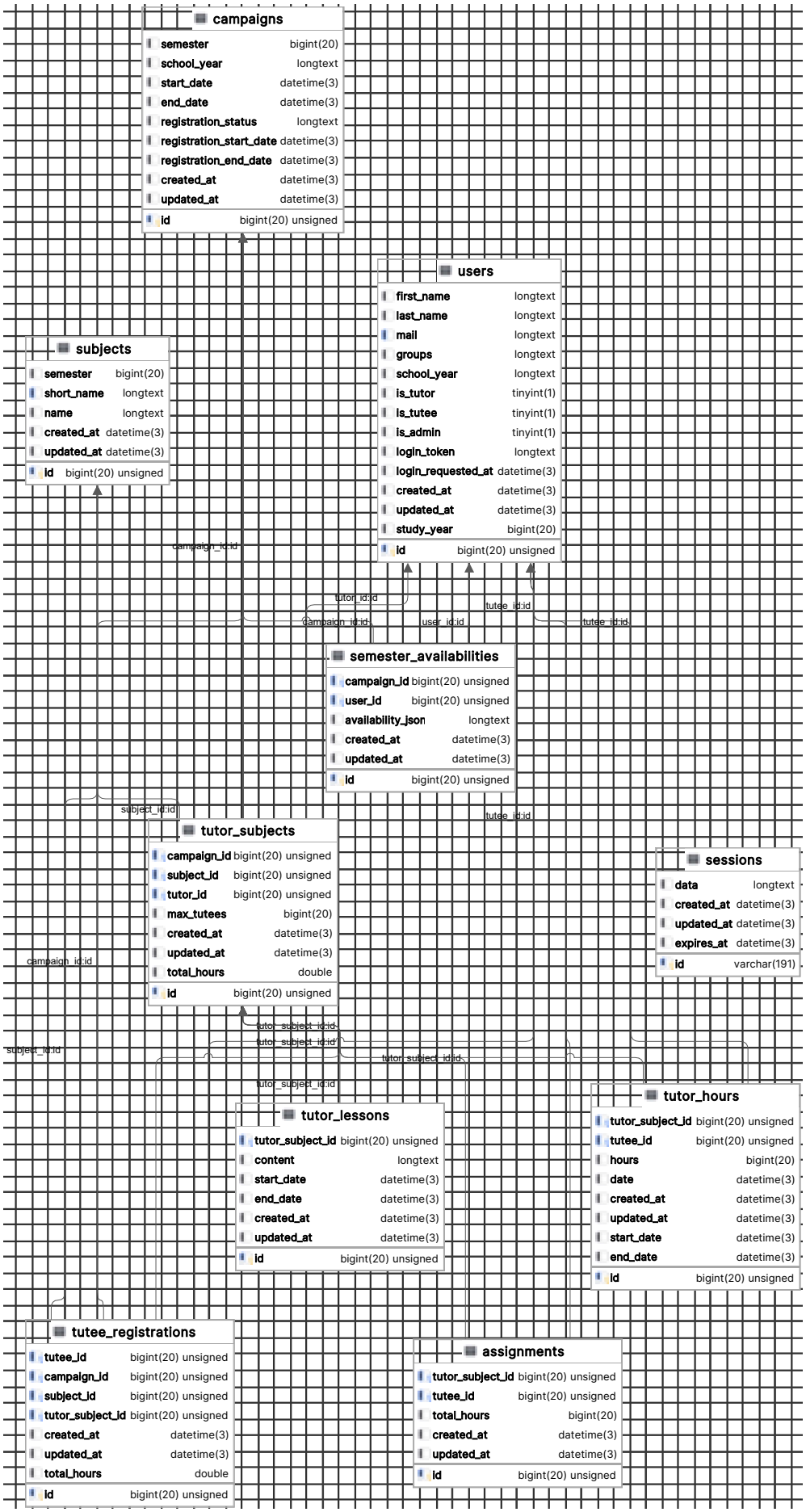
Consultez le `README.md` afin de procéder à l'installation des dépendances du projet. En cas de besoin, n'hésitez pas à me contacter si vous souhaitez que je fournisse des builds cross-plateformes du binaire du serveur.

Les instructions suivantes permettent de tester la plateforme localement :

— Remplir le `.env` en prenant exemple sur le `.env.example`
— Définissez `DEV_MODE=true`, `SESSIONS_KEY=f6z4erv1c52` (random string)
— Créer un utilisateur et une base de données pour l'outil, remplir le `MYSQL_DSN`
— La partie mail peut être ignorée (utilisation d'un service tiers)
— En local, définissez `DOMAIN=localhost`
— Importez le fichier `demo_data.sql` dans votre base de données
— Chaque utilisateur comporte un token simple à utiliser. Pour vous connecter en tant que l'utilisateur `1`, accédez à http://localhost:3000/login?token=1

POST	/auth/login
POST	/auth/send-link
GET	/auth/self
GET	/auth/logout
GET	/assignments/tutee
GET	/assignments/tutor
GET	/campaign/:campaignId/agenda
GET	/campaign/:campaignId/availabilities
POST	/campaign/:campaignId/availabilities
GET	/campaign/:campaignId/subjects
GET	/campaign/:campaignId/tutee/registrations
POST	/campaign/:campaignId/tutee/registrations
GET	/campaign/:campaignId/tutor/registrations
POST	/campaign/:campaignId/tutor/registrations
GET	/admin/subjects
GET	/admin/users
GET	/admin/campaigns
POST	/admin/campaigns
PATCH	/admin/campaign/:campaignId
GET	/admin/campaign/:campaignId/overview
GET	/admin/campaign/:campaignId/users
GET	/admin/campaign/:campaignId/assignments
POST	/admin/campaign/:campaignId/assignments
DELETE	/admin/campaign/:campaignId/assignments/tutor
DELETE	/admin/campaign/:campaignId/assignments/tutee
GET	/admin/campaign/:campaignId/generate-assignments
GET	/tutoring/:tutorSubjectId/summary
POST	/tutoring/:tutorSubjectId/lessons
PATCH	/tutoring/:tutorSubjectId/lesson/:lessonId
DELETE	/tutoring/:tutorSubjectId/lesson/:lessonId
POST	/tutoring/:tutorSubjectId/hours
PATCH	/tutoring/:tutorSubjectId/hour/:hourId
DELETE	/tutoring/:tutorSubjectId/hour/:hourId

TABLE 3 – Liste des routes et méthodes disponibles sur l'API REST


campaigns
semester bigint(20)
school_year longtext
start_date datetime(3)
end_date datetime(3)
registration_status longtext
registration_start_date datetime(3)
registration_end_date datetime(3)
created_at datetime(3)
updated_at datetime(3)
id bigint(20) unsigned
users
first_name longtext
last_name longtext
mail longtext
groups longtext
school_year longtext
is_tutor tinyint(1)
is_tutee tinyint(1)
is_admin tinyint(1)
login_token longtext
login_requested_at datetime(3)
created_at datetime(3)
updated_at datetime(3)
study_year bigint(20)
id bigint(20) unsigned
subjects
semester bigint(20)
short_name longtext
name longtext
created_at datetime(3)
updated_at datetime(3)
id bigint(20) unsigned
campaign_id:id
tutor_id:id
campaign_id:id
user_id:id
tutee_id:id
tutee_id:id
semester_availabilities
campaign_id bigint(20) unsigned
user_id bigint(20) unsigned
availability_json longtext
created_at datetime(3)
updated_at datetime(3)
id bigint(20) unsigned
subject_id:id
tutee_id:id
tutor_subjects
campaign_id bigint(20) unsigned
subject_id bigint(20) unsigned
tutor_id bigint(20) unsigned
max_tutees bigint(20)
created_at datetime(3)
updated_at datetime(3)
total_hours double
id bigint(20) unsigned
sessions
data longtext
created_at datetime(3)
updated_at datetime(3)
expires_at datetime(3)
id varchar(191)
campaign_id:id
subject_id:id
tutor_subject_id:id
tutor_subject_id:id
tutor_subject_id:id
tutor_subject_id:id
tutor_lessons
tutor_subject_id bigint(20) unsigned
content longtext
start_date datetime(3)
end_date datetime(3)
created_at datetime(3)
updated_at datetime(3)
id bigint(20) unsigned
tutor_hours
tutor_subject_id bigint(20) unsigned
tutee_id bigint(20) unsigned
hours bigint(20)
date datetime(3)
created_at datetime(3)
updated_at datetime(3)
start_date datetime(3)
end_date datetime(3)
id bigint(20) unsigned
tutee_registrations
tutee_id bigint(20) unsigned
campaign_id bigint(20) unsigned
subject_id bigint(20) unsigned
tutor_subject_id bigint(20) unsigned
created_at datetime(3)
updated_at datetime(3)
total_hours double
id bigint(20) unsigned
assignments
tutor_subject_id bigint(20) unsigned
tutee_id bigint(20) unsigned
total_hours bigint(20)
created_at datetime(3)
updated_at datetime(3)
id bigint(20) unsigned