



TELECOM NANCY

RAPPORT DE PROJET PII

Avril 2021

Application de gestion de données du Concours Mines-Télécom

GROUPE 9

Étudiants :

Maël SAILLOT
Céline ZHANG
Ahmed ZIANI

Numéro Étudiant :

31831300
32024925
31824316

Encadrants du projet :

Sébastien DA SILVA
Gérald OSTER



Remerciements

Nous tenons à remercier chaque membre de l'équipe pour leurs contributions à la réalisation de ce projet ainsi que toute l'équipe pédagogique de Telecom Nancy, en particulier à M. Gérald OSTER, M. Sébastien DA SILVA et M. Christophe BOUTHIER, pour l'aide qu'ils nous ont apporté afin de mener à bien notre scolarité durant ces temps difficiles, et tout au long du projet.

Résumé

Le projet concerne une application de gestion de la base de données du concours Mines-Télécom. Cette application doit être facile à utiliser, elle doit permettre à l'utilisateur d'accéder aux informations des candidats simplement, et d'afficher des résultats (notes, rangs, écoles, coordonnées, bac, etc.), ils doivent pouvoir être filtré si nécessaire. Afin de réaliser une tel application, le groupe a commencé la création d'un modèle adapté aux données fournies, puis a rempli la base de données issue de ce modèle. Ensuite, elle a fait l'implémentation de l'application et des fonctionnalités (affichage, filtrage, recherche, etc). Par ailleurs, la cohérence des données doit être vérifiée, en conséquence, un script de vérification a été réalisé.

Abstract

This project is about an application that will help manage the Mines-Télécom competitive exam data. The application needs to be easy to use, the user should be able to access easily to candidates information, and it needs to display the results (scores, ranks, schools, contact information, graduation, etc.) that have to be filtered when necessary. In order to create this app, the team started by visualizing it through a model that was in compliance with the given datas, before filling the database with the model. Then, the team implemented the app and its features (display, filtering, search, etc). Besides, the data's consistency needs to be checked, that is the reason why a test script was written.

Table des matières

Introduction	5
Cahier de charges	6
Mentions légales	6
1 État de l'art	7
1.1 Formes normales	7
1.1.1 Première forme normale	7
1.1.2 Deuxième forme normale	7
1.1.3 Troisième forme normale	8
1.2 Conception de la structure	8
1.3 Contraintes d'intégrité	11
2 Implémentation et application des algorithmes	14
2.1 Script de remplissage de la base de données	14
2.1.1 Algorithmes de remplissage	14
2.1.2 Regroupement des scripts et optimisation	14
2.1.3 Algorithme du script de test de cohérence	15
2.1.4 Interface en ligne de commande	15
2.2 Application web	16
2.2.1 Outils utilisés	16
2.2.2 Fonctionnalités	16
3 Tests de cohérence et de performance	18
3.1 Cohérence	18
3.2 Performance	18
4 Gestion de projet	19
4.1 Équipe de projet	19
4.2 Analyse du projet	19
4.2.1 Définition des objectifs	19
4.2.2 Analyse des risques : Matrice SWOT	20
4.2.3 Répartition des tâches : Matrice RACI	21
4.3 Organisation du projet	21
4.3.1 Durée	21
4.3.2 Le cadre méthodologique SCRUM	21
4.4 Outils de travail	23
4.4.1 Communication	23
4.4.2 Programmation	23
4.4.3 Partage du travail	23
4.4.4 Rédaction du rapport	23
4.5 Les réunions de projet	24
Conclusion	25
Bilan global du projet d'équipe	26

Annexes	28
Les déclarations sur l'honneur de non-plagiat	28
Trello	31
Comptes rendus de réunions	34
Réunion d'équipe du 3 avril 2021	34
Réunion d'équipe du 6 avril 2021	36
Réunion d'équipe du 11 avril 2021	37
Réunion d'équipe du 25 avril 2021	39
Réunion d'équipe du 29 avril 2021	40
Réunion d'équipe du 3 mai 2021	41
Réunion d'équipe du 14 mai 2021	42
Réunion d'équipe du 24 mai 2021	43
Réunion d'équipe du 4 juin 2021	44
Bibliographie	46

Introduction

Dans le cadre de notre module PPII, il est demandé de faire un projet mettant œuvre à la fois nos compétences en base de données relationnelles, nos compétences en programmation de script, et nos connaissances en web. Le projet a pour finalité le développement d'une application qui permet la gestion d'une base de données du concours Mines-Télécom.

L'objectif de ce projet est de réaliser une structure de base de données relationnelles respectant les critères de la troisième forme normale à partir des informations fournies par l'utilisateur. Les informations concernent les candidats inscrits aux concours d'admission des écoles de la banque Mines-Télécom, elles doivent être mises dans la base de données relationnelles produite. L'application demandée doit pouvoir accéder aux informations de la base de données et les afficher de plusieurs manières. Un script de test de cohérence des données est nécessaire pour omettre toute erreur dans nos données. La performance est demandée, le remplissage des tables de la base de données par le script ne doit pas dépasser une certaine durée précisée dans le cahier des charges.

Tout d'abord, nous devons créer notre base de données, c'est-à-dire réaliser notre modèle de données relationnelles respectant la troisième forme normale et les contraintes de certains champs. Ensuite, il faut remplir les tables de la base de données par les données fournies, de ce fait des fonctions d'importation de données, regroupées dans un script de remplissage, ont été implémentées. Puis, une vérification des tables a été faite pour identifier les données manquantes, les erreurs de remplissage, et les données inutilisables (comme les candidats anonymes). Enfin, pour afficher ces données par un utilisateur, nous avons mis en place une application web qui permet de montrer les données de plusieurs manières selon les demandes. Les principales technologies utilisées sont **SQLite3** qui permet de créer la base de données, **SQLiteStudio** qui permet d'afficher la base de données, les bibliothèques **Flask**, **Pandas**, **Openpyxl** de Python pour écrire les scripts et l'application web.

Les résultats du projet sont présentés dans ce rapport, il y a une partie consacrée à l'état de l'art, c'est-à-dire la présentation des notions en base de données relationnelles, et l'explication théorique des procédés utilisés ; une partie consacrée à l'implémentation et l'application de l'algorithme, elle explique le raisonnement entre les lignes de codes, et présente les résultats lors de l'exécution des codes. Dans ce rapport se trouve également une partie pour les tests et les performances des fonctions réalisées ; une partie présentant la gestion de projet de l'équipe et les moyens utilisés. À la fin de ce rapport, se trouvent les annexes, dans lesquelles se rangent les déclarations de non-plagiat, les tableaux d'organisation du travail, les comptes rendus de réunions, et d'autres divers documents.

Cahier des charges

Travaux demandés
Base de données pour les informations du concours des Mines-Télécom
Script de remplissage de la base de données
Temps de remplissage < 3 min
Application permettant d'accéder aux données (candidats, notes, classements, diverses)
Affichage des résultats de recherches dans l'application avec des graphes ou sur une carte
Amélioration de l'apparence de l'application
Faciliter l'utilisation de l'application (rendre interactif)
Tests de cohérence des données fournies et du remplissage

TABLE 1 – Le cahier des charges

Mentions légales

Ce projet n'est pas destiné à un usage commercial, ainsi, les images présentées, notamment les images de tests, d'applications ou de gestion de projet, ne sont pas destinées à la publication. Cependant, le caractère strictement scolaire de ce projet nous autorise à les inclure en accord avec :

- Code civil : articles 7 à 15, article 9 : respect de la vie privée
- Code pénal : articles 226-1 à 226-7 : atteinte à la vie privée
- Code de procédure civil : articles 484 à 492-1 : procédure de réfééré
- Loi n°78-17 du 6 janvier 1978 : Informatique et libertés, Article 38

Les déclarations sur l'honneur de non-plagiat des membres de l'équipe du projet sont présentés dans les quatre premières pages de l'annexe.

Cette version¹ du rapport a été finalisée le 11 Juin 2021.

1. version 1

1 État de l'art

Actuellement, nous sommes dans un monde qui exploite fortement des données, que ce soit pour l'administration, l'économie ou le divertissement. Un problème de gestion de ces données se pose, notamment les limites de stockage et le temps d'accès aux données. Il existe de nos jours plusieurs manières gérer ces données de manière optimale, ce sont les structures de base de données relationnelles. Pour qualifié la robustesse d'un modèle relationnel, nous avons plusieurs critères à vérifier. Ces critères ainsi que notre modèle sont présentés dans les paragraphes suivants.

1.1 Formes normales

Les formes normales sont les résultats de la normalisation des modèles de données permettant, selon le niveau, d'éviter les redondances, les problèmes de mise à jour, de garder la cohérence et d'optimiser le stockage des données. Dans le modèle de type OLTP², il existe huit formes normales. Les trois premières sont les plus connus, c'est ce que nous allons rappeler [2]. Le non-respect de ces règles engendre des redondances inutiles. Dans les exemples qui suivent, les attributs soulignés forment les clés.

1.1.1 Première forme normale

Une relation (ayant par définition une clé) dont les attributs possèdent tous une valeur sémantiquement atomique³. Les attributs constitués par un ensemble de valeurs énumérées dont les différentes valeurs sont sémantiquement indépendantes et les attributs n'ayant aucune valeur violent l'atomicité.

<u>id</u>	nom	prénoms
1	MERLIN	[Pierre, Henry]
2	BUTTON	[Georges, Léo, Ane]
3	DUPONT	[Hélène, Inès]
4	RENARD	[Chléo, Marie]
5	CHEMIN	[Karim, Claire]
6	TURPIN	[Raphaël]

FIGURE 1 – Exemple de table non-atomique, l'attribut **prénoms** viole l'atomicité

1.1.2 Deuxième forme normale

Les attributs d'une relation sont séparés en deux groupes, le premier est composé de la clé, qui peut être un ou plusieurs attributs, et le deuxième est composé des autres attributs, éventuellement vide. Pour respecter la 2FN⁴, il faut d'abord respecter la 1FN, et il faut que tout attribut du deuxième groupe ne dépende pas d'un sous-ensemble (strict) d'attribut(s) du premier groupe. C'est-à-dire, un attribut non clé ne dépend pas d'une partie de la clé mais de toute la clé⁵.

2. online transaction processing, en français, le traitement transactionnel en ligne est un type d'application informatique qui sert à effectuer des modifications d'informations en temps réel. [1]

3. ce qu'on ne peut pas diviser.
4. deuxième forme normale
5. extrait de Wikipédia 2FN

date	heure	jour
2021-06-01	12:00:00	mardi
2021-06-02	14:00:00	mercredi
2021-06-03	15:30:20	jeudi
2021-06-04	16:59:59	vendredi
2021-06-05	18:30:01	samedi
2021-06-06	22:00:00	dimanche

FIGURE 2 – Exemple de table non-2FN, l’attribut **jour** ne dépend que de l’attribut **date**

1.1.3 Troisième forme normale

Pour respecter la 3FN, il faut d’abord respecter la 2FN et il faut que tout attribut du deuxième groupe ne dépende pas d’un sous-ensemble (strict et excluant l’attribut considéré) d’autres attribut(s) du second groupe. C’est-à-dire, un attribut non clé ne dépend pas d’un ou plusieurs attributs ne participant pas à la clé. Autrement dit : « Tous les attributs non clé doivent dépendre directement de la clé, au sens où il n’y a aucun attribut non clé dépendant de la clé par dépendances transitives par l’intermédiaire d’autres attributs non clé »⁶.

id_livre	auteur	nationalité
123456	Flore GUINE	belge
234567	Bob MARIN	espagnole
345678	Julien PUIT	française
456789	Philippe DOT	norvégienne
567890	Marion HAN	française
678901	John FOUR	indienne

FIGURE 3 – Exemple de table non-3FN, l’attribut **nationalité** dépend de l’attribut **auteur**

1.2 Conception de la structure

Nous avons en tout 18 tables contenant des champs dans lesquels on range les informations. Les tables sont représentées dans le modèle de données qui se trouve ci-après, Figure ???. L’outil dbdiagram.io a été utilisé pour faire la représentation graphique du modèle de données. Voici une présentation des tables conçues :

- **candidat** : affiche les informations du candidat inscrit sur le site **scei**, la clé primaire est le numéro unique du candidat ;
- **classe** : stock les différentes classes associées à une clé primaire , utile pour remplir le champ **classe** de la table **candidat** représentant la classe d’origine, en relation 1N (*foreign key*) ;
- **civilite** : sauvegarde les appellations, Monsieur ou Madame (il est possible d’en ajouter), chacun identifié à une clé primaire, cela permet de gagner du stockage sur l’attribut **civ** de la table **candidat**, en relation 1N ;
- **ep_option** : regroupe les différentes combinaisons pour les épreuves d’options (par exemple LV1 Espagnol), elles sont associées par clé primaire, cette table permet de remplir les champs **option1**, **option2**, **option3**, **option4** de la table **candidat**, en relation 1N ;

6. extrait de Wikipédia 3FN

- **etat_dossier** : on y trouve le code avancement du dossier du candidat, permet de remplir le champ **dossier** de la table **candidat**, en relation 1N ;
- **serie_bac** : contient les différents BAC (S, ES, STI2D, etc.), permet de remplir le champ **bac** de la table **candidat**, en relation 1N ;
- **csp_parent** : on y trouve les différents milieux socio-professionnel des parents, associé à une clé primaire , utile pour remplir les champs **csp_pere**, **csp_mere** de la table **candidat**, en relation 1N ;
- **pays** : garde les différents pays et la nationalité correspondante sont associés à une clé primaire qui sert de *foreign key* pour la relation 1N des champs **code_adr_pays**, **code_pays_naissance**, **code_pays_nationalite** de la table **candidat** ;
- **autres_prenoms** : 2e prénoms de certains candidats, lié à une *foreign key* (**candidat.code-1N-etudiant**) ;
- **epreuve** : contient les différentes épreuves, ont un nom (**lib**) unique et un type, la clé primaire **epreuve.id** est une clé étrangère de ma table **notes** ;
- **notes** : stock les résultats des candidats selon les épreuves identifiées par leur **id** de la table **epreuve** ;
- **type_classement** : admissible, admis, etc., il est nécessaire de différencier les types de classements pour savoir à quoi correspondant le classement demandé, la clé primaire de cette table s'utilise en clé étrangère dans la table **classement** ;
- **classement** : on a besoin d'identifier la nature du classement, d'où la FK⁷ en champ **type**, la clé primaire indique le classement unique, et on demande aussi à que **etudiant** (en FK avec **candidat.code**, relation 1N) et **type** soient un couple unique ;
- **ecole** : rassemble les écoles, le **lib** est associé à une clé primaire qui sert de clé étrangère à **voeux.ecole** ;
- **voeux** : regroupe les voeux émis des candidats, la table est composée d'une clé primaire **code**, d'une clé étrangère qui vient de **ecole.code**, et d'un ordre de classement ;
- **concours** : contient les différentes filières du concours, avec **lib**⁸ et **voie**⁹, le champ **concours.code** est la clé primaire qui sert de clé étrangère à **candidat.voie_concours** en relation 1N ;
- **établissement** : regroupe tous les établissements d'origine des candidats, **rne** est la clé primaire qui sert de FK pour **candidat.établissement**, suivi des attributs donnant des informations sur l'établissement ;
- **etat_reponse** : on y trouve les différents états de réponse (lors de la phase admission certainement), c'est la seule table qui est reliée à rien.

7. foreign key

8. nom du concours

9. la voie du concours

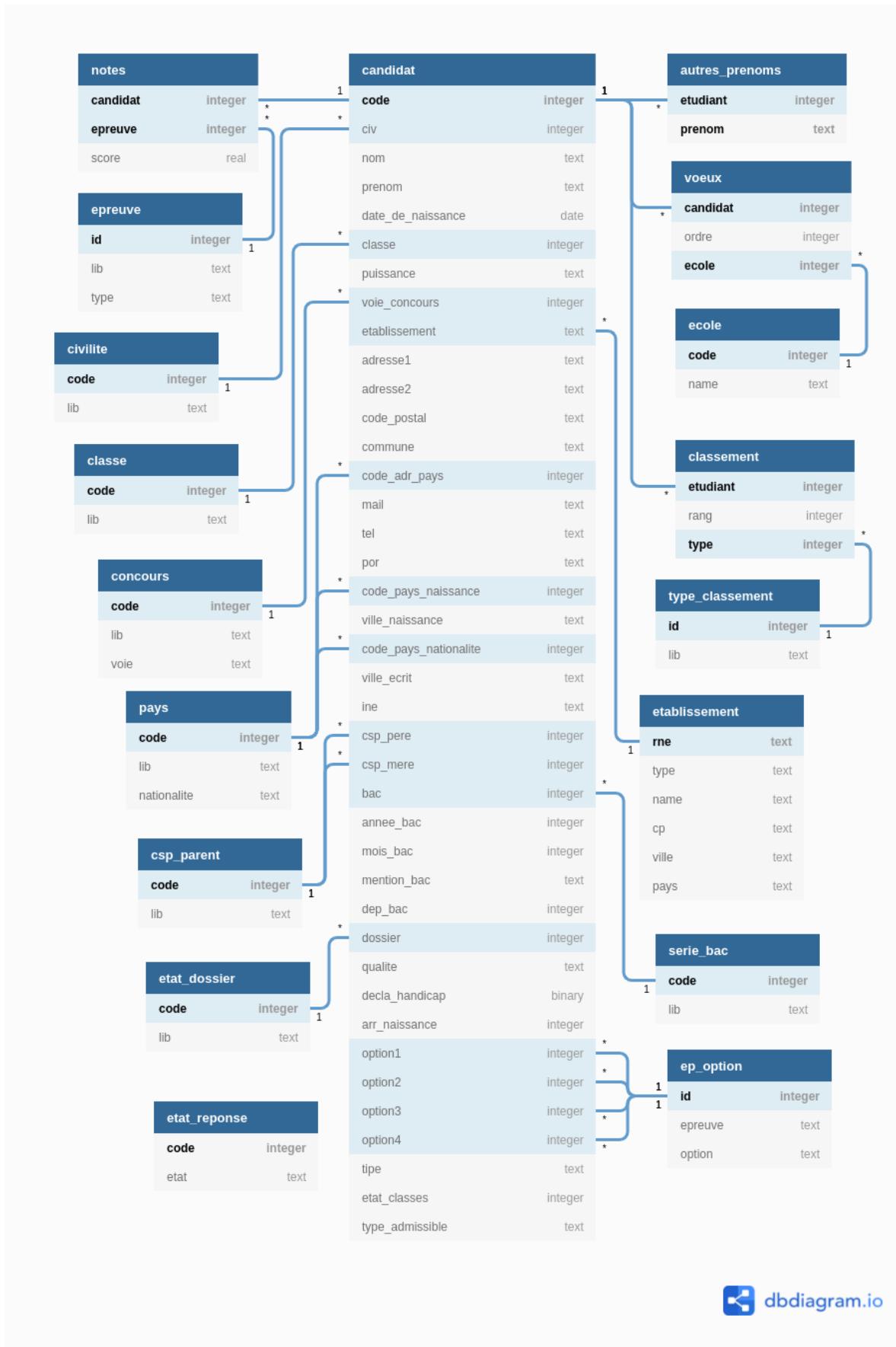


FIGURE 4 – Schéma du modèle de données

Chaque attribut de nos tables possèdent une valeur sémantiquement atomique, qui ne peut pas être divisé. Les attributs ne possèdent pas de valeurs multiples (listes, tableaux, etc.) ni de valeurs NULL, elles sont soit des `integer`, soit des `text`, et rarement des `binary` qui peuvent être ramenés à des `integer` (dans l'implémentation en SQLite, le `binary` n'existe pas, on utilisera un `integer`). De plus, les attributs non-clé dépendent de ou des attributs clés. Donc, le modèle est 1FN.

En séparant les attributs clés et les attributs non-clé, dans chaque table les attributs non-clé dépendent de l'ensemble des attributs clés de la table. Les attributs en bleus sont des clés. Dans la table `classement`, le couple (`etudiant`, `type`) est une clé primaire, et `rang` désignant le rang dépend du candidat et du type de classement (admissible, admis, etc.); dans la table `voeux`, le couple (`candidat`, `ecole`) est une clé et `ordre` dépend de ce couple, car l'ordre des voeux est associé au candidat et à l'école concernée ; de même pour le couple (`candidat`, `epreuve`) dans la table `notes`, `score` (la note) n'a de signification que lorsqu'on précise le candidat et l'épreuve en question. Pour les autres tables, il n'y a qu'une seule clé, les attributs non-clé dépendent que de cette clé. Ainsi, dans toutes les tables, les attributs non-clé dépendent de la clé entière, le modèle est 2FN.

Dans la table `candidat`, les attributs non-clé sont mutuellement indépendants, en particulier, pour les attributs `code_postal` et `commune`, deux communes différentes peuvent avoir le même code postal, notamment les communes voisines. Par exemple, Carcès, Correns, Cotignac, Entrecasteaux et Montfort-sur-Argens ont le même code postal qui est 83570 [3]. De même pour la table `établissement`, les attributs non-clé sont mutuellement indépendants. Les banques de concours ont pour la plupart des filières en commun, donc les attributs `concours.lib` et `concours.vie` sont indépendants. Tout comme le nom de l'épreuve et le type de l'épreuve, il peut y avoir une épreuve de mathématiques à l'écrit et à l'oral, donc les attributs `epreuve.lib` et `epreuve.type` sont indépendants. De même pour, les options des épreuves, le même type d'épreuve peut proposer des options différentes, par exemple, l'épreuve de Info./SI propose deux options, Sciences Industrielles et Informatique, donc les attributs `ep_option.epreuve` et `ep_option.option` n'ont pas de relation. Les autres tables n'ont qu'un seul attribut non-clé, dont nécessairement, ils dépendent de la clé de la table. De ce fait, le modèle de données est bien 3FN.

1.3 Contraintes d'intégrité

Il a fallu contraindre notre base de données pour qu'elle ne prenne que des valeurs autorisées. Ces contraintes sont nécessaires pour que la base ne contiennent pas des données incohérentes ou incomplètes. Dans ce but, des contraintes ont été ajouté à certains champs des tables de la base de données. Listes du contenu des champs de la Figure 4 :

- `etat_reponse.code` : un entier, clé primaire ;
- `etat_reponse.etat` : du texte non vide ;
- `ecole.code` : un entier, clé primaire ;
- `ecole.name` : du texte, non vide ;
- `établissement.rne` : la rne de l'établissement et sert de clé primaire ;
- `établissement.type` : du texte ;
- `établissement.name` : du texte, non vide ;
- `établissement.cp` : du texte (certains codes postaux étrangers contiennent des lettres) ;
- `établissement.ville` : du texte, non vide ;
- `établissement.pays` : du texte ;
- `concours.code` : un entier, clé primaire ;
- `concours.lib` : du texte, non vide ;

- `concours.voie` : du texte, non vide, couple unique avec `concours.lib` ;
- `voeux.candidat` : un entier, non vide, clé étrangère provenant de `candidat.code` ;
- `voeux.ordre` : un entier strictement supérieur à 0 et, non vide, forme un couple unique avec `voeux.candidat` ;
- `voeux.ecole` : un entier, non vide, clé étrangère provenant de `ecole.code`, forme une clé primaire avec `voeux.candidat` ;
- `classement.etudiant` : un entier, non vide, clé étrangère provenant de `candidat.code` ;
- `classement.rang` : un entier strictement supérieur à 0 et non vide ;
- `classement.type` : un entier, non vide, clé étrangère provenant de `type_classement.id`, forme une clé primaire avec `classement.etudiant` ;
- `type_classement.id` : un entier, clé primaire ;
- `type_classement.lib` : du texte unique, non vide, on ne veut pas plus de une fois le même type de classement ;
- `epreuve.id` : un entier, clé primaire ;
- `epreuve.lib` : du texte, non vide ;
- `epreuve.type` : du texte qui se trouve dans ECRIT, ORAL, SPECIFIQUE, CLASSEMENT ;
- `notes.candidat` : un entier, non vide, clé étrangère provenant de `candidat.code` ;
- `notes.epreuve` : un entier, non vide, clé étrangère provenant de `epreuve.id`, forme une clé primaire avec `notes.epreuve` ;
- `notes.score` : un réel, non vide ;
- `autres_prenoms.etudiant` : un entier, non vide, clé étrangère provenant de `candidat.code` ;
- `autres_prenoms.prenom` : du text, non vide, forme une clé primaire avec `autres_prenoms.etudiant`¹⁰ ;
- `pays.code` : un entier, clé primaire ;
- `pays.lib` : du text unique ;
- `pays.nationalité` : du text unique ;
- `csp_parent.code` : un entier, clé primaire ;
- `csp_parent.lib` : du texte unique, non vide ;
- `serie_bac.code` : un entier, clé primaire ;
- `serie_bac.lib` : du texte unique, non vide ;
- `etat_dossier.code` : un entier, clé primaire ;
- `etat_dossier.lib` : du texte unique, non vide ;
- `ep_option.id` : un entier, clé primaire ;
- `ep_option.epreuve` : du texte, non vide ;
- `ep_option.option` : du texte, non vide, forme un couple unique avec `ep_option.epreuve` ;
- `civilite.code` : un entier, clé primaire ;
- `civilite.lib` : du texte unique, non vide, en particulier M. et Mme ;
- `classe.code` : un entier, clé primaire ;
- `classe.lib` : du texte unique, non vide ;

10. on suppose que tout le monde n'a qu'un deuxième prénom, dans les données les candidats ont au plus un deuxième prénom

- `candidat.code` : un entier, clé primaire, représente le numéro scel du candidat ;
- `candidat.civ` : une entier, non vide, clé étrangère provenant de `civilite.code` ;
- `candidat.nom` : du texte, non vide ;
- `candidat.prenom` : du texte, non vide ;
- `candidat.date_de_naissance` : du type date ;
- `candidat.classe` : un entier, clé étrangère provenant de `classe.code` ;
- `candidat.puissance` : du texte à prendre dans 3/2, 5/2, 7/2 ;
- `candidat.voie_concours` : un entier, clé étrangère provenant de `concours.code` ;
- `candidat.etablissement` : du texte, clé étrangère provenant de `etablissement.rne` ;
- `candidat.adresse1` : du texte, non vide ;
- `candidat.adresse2` : du texte ;
- `candidat.code_postal` : du texte (car pourrait contenir des lettres, non vide) ;
- `candidat.commune` : du texte, non vide ;
- `candidat.code_adr_pays` : un entier, non vide, clé étrangère provenant de `pays.code` ;
- `candidat.mail` : du texte, non vide ;
- `candidat.tel` : du texte (car il peut y avoir +33(0)1) ;
- `candidat.por` : du texte (car il peut y avoir +33(0)6) ;
- `candidat.code_pays_naissance` : un entier, non vide, clé étrangère provenant de `pays.code` ;
- `candidat.ville_naissance` : du texte ;
- `candidat.code_pays_nationalite` : un entier, non vide, clé étrangère provenant de `pays.code` ;
- `candidat.ville_ecrit` : du texte ;
- `candidat.ine` : du texte unique ;
- `candidat.csp_pere` : un entier, clé étrangère provenant de `csp_parent.code` ;
- `candidat.csp_mere` : un entier, clé étrangère provenant de `csp_parent.code` ;
- `candidat.bac` : un entier, clé étrangère provenant de `serie_bac.code` ;
- `candidat.annee_bac` : un entier ;
- `candidat.mois_bac` : un entier ;
- `candidat.mention_bac` : du texte, TB, B, AB, S ;
- `candidat.dep_bac` : un entier, département d'obtention du BAC ;
- `candidat.dossier` : un entier, clé étrangère provenant de `etat_dossier.code` ;
- `candidat.qualite` : du texte ;
- `candidat.decla_handicap` : un entier, 0 ou 1, par défaut 0 ;
- `candidat.arr_naissance` : un entier, par défaut 0 ;
- `candidat.option1` : un entier, clé étrangère provenant de `ep_option.id` ;
- `candidat.option2` : un entier, clé étrangère provenant de `ep_option.id` ;
- `candidat.option3` : un entier, clé étrangère provenant de `ep_option.id` ;
- `candidat.option4` : un entier, clé étrangère provenant de `ep_option.id` ;
- `candidat.tipe` : du texte ;
- `candidat.etat_classes` : un entier ;
- `candidat.type_admissible` : du texte¹¹.

11. lettre entre A et B

2 Implémentation et application des algorithmes

Tout d'abord, nous avons écrit des algorithmes permettant de remplir la base de données à partir des fichiers excel fournis [4], puis ces codes ont été regroupés dans un seul script et optimisé. Ensuite, un script de test de cohérence a été fait pour vérifier la cohérence des fichiers. Et enfin, nous avons réalisé l'application de gestion avec les fonctionnalités décrites qui font appel à du SQL pour récupérer les informations dans la base de données.

2.1 Script de remplissage de la base de données

Pour gérer notre base de donnée, nous utilisons le système de gestion de base de données SQLite 3 et le langage SQL. L'initialisation de la base de données est faite grâce au fichier `createdb.sql` qui crée toutes les tables de notre modèle et remplit les tables `civilite` et `type_classement`. Ce fichier peut être exécuté avec SQLite mais notre script qui importe le reste des données permet aussi de l'exécuter. Les scripts de cette partie ont été écrits dans le langage Python 3 avec les modules suivant :

- `pandas` : permet de lire les fichiers `.csv` et `.xlsx` et permet de nommer les colonnes des tableurs [5] ;
- `openpyxl` : lit uniquement les fichiers `.xlsx` et ne permet pas de nommer les colonnes mais charge les tableurs plus rapidement [6] ;
- `sqlite3` : permet la connexion avec la base de donnée [7] ;
- `click` : permet l'intégration du script dans une interface en ligne de commande ;
- `pathlib` : permet de gérer facilement les chemins systèmes.

2.1.1 Algorithmes de remplissage

Nous chargeons les tableurs l'un après l'autre et remplissons les tables en fonction des données contenues dans ces fichiers. Tous les fichiers sont traités de façon similaire mais le code a du être adapté aux caractéristiques de chaque fichier.

La fonction suivante est un exemple type de la façon dont nous avons rempli la base de donnée. Elle remplit la table `ecole` depuis les données du fichier `listeEcoles.xlsx`. Le paramètre `database` est la connexion vers la base de donnée et `path` est le chemin du dossier contenant les tableurs. La méthode `c.execute` permet de d'exécuter une commande SQL. Ici elle insère un couple de donnée dans la table `ecole`.

```
def import_ecole(database: sql.Connection, path: Path):
    data = pd.read_excel(path/'listeEcoles.xlsx', header=0)
    c = database.cursor()
    for i in range(len(data)):
        nom = data['Nom_ecole'][i]
        code = int(data['Ecole'][i])
        c.execute("INSERT INTO ecole VALUES (?,?)", (code, nom))
```

2.1.2 Regroupement des scripts et optimisation

Nous nous sommes réparti l'écriture des fonctions donc nous avons dû regrouper les différents morceaux de code. Cette phase c'est accompagné d'une optimisation du code. Ainsi nous ne chargeons qu'un fichier à la fois et chaque fichier n'est chargé qu'un seule fois. De cette façon le temps de chargement des fichiers et la charge mémoire sont réduits au maximum.

L'ordre de traitement des données est défini de façon à s'aider de tables précédemment remplies. Par exemple pour les candidats provenant de la filière ATS nous n'avons que le nom du pays de leur adresse. Or la clé étrangère à mettre dans la table candidat est le code du pays. Nous devons donc rechercher dans la table pays le code correspondant. Si il n'existe pas déjà nous l'ajoutons à la table.

Enfin une première review du code a été faite pour éliminer les parties dupliquées ou trop ressemblantes. Elles ont été remplacées par des appels à des fonctions.

2.1.3 Algorithme du script de test de cohérence

Comme pour tester la cohérence des données nous n'effectuons que des requêtes SQL, tous les identifiants des données à tester ont été mis dans un fichier texte. Il est construit selon l'exemple suivant :

```
ADMIS_ATS.xlsx;ADMIS_MP-SPE.xlsx
Can _cod;pk
Civ _lib;civilite;ci;ci.lib;J;candidat;ca;ca.civ=ci.code;W;ca.code
```

Une première ligne sans tabulation est la liste des fichiers pour lesquels les lignes suivantes vont se référer. Les lignes suivantes commencent avec une tabulation et le premier champs correspond à un nom de colonne des tableurs. Les champs suivants caractérisent cette colonne. Si il y a pk, alors cette colonne va servir de clé primaire pour identifier l'enregistrement à vérifier. Sinon l'ensemble des champs suivants décrivent la requête SQL à effectué. Les 3 premiers champs indiquent la table et le nom du champs de la table à récupérer. Le champs J (qui est optionnel) et les 3 suivants indique une jointure à effectué et le champs W et le suivant caractérise la partie WHERE de la requête.

Pour résumer, chaque ligne suit la grammaire suivante :

$$\{(L, F, P, A, S, J), (\text{;}, \text{pk}, \text{J}, \text{W}, \text{\t}, \text{statements}), \rightarrow, L\}$$

$$\left\{ \begin{array}{l} L \rightarrow F|P \\ F \rightarrow \text{filename}|\text{filename};F \\ P \rightarrow \text{column_header};A \\ A \rightarrow \text{pk}|S \end{array} \right. \quad \left\{ \begin{array}{l} S \rightarrow \text{table};\text{table_alias};\text{column_name};JW \\ J \rightarrow \epsilon|\text{J};\text{table};\text{table_alias};\text{on_statement}; \\ W \rightarrow \text{W};\text{where_statement} \end{array} \right.$$

2.1.4 Interface en ligne de commande

Grâce au module Click, l'ensemble des scripts pour la création, le remplissage et la vérification (voir section 3.1) ont été regroupés dans une unique commande. La commande prend en paramètre les chemins vers les différents fichiers nécessaire et comme option les différentes étapes à exécuter. Exemple d'utilisation :

```
importxl concours.db files/ --init createdb.sql --import --verif
```

Ici les fichiers tableurs sont situés dans le dossier `files/` et la base de donnée `concoures.db` est initialisée avec le fichier `createdb.sql`. La commande effectue aussi l'importation et la vérification des données.

2.2 Application web

2.2.1 Outils utilisés

Pour la conception de notre application web, nous avons décidé de rester sur Python car c'est un langage intuitif, très flexible et qui possède plusieurs librairies qui facilitent le développement web et la manipulation de bases de données (et c'est aussi le langage que nous avons utilisé en WebBd).

La partie Back-end de notre application repose essentiellement sur **Flask** et **SQLite3** qui sont deux framework disponibles sur Python.

- **Flask** est un micro-framework simple et léger qui permet, entre autres, de créer un serveur de développement et de débugage ainsi qu'un moteur de template pour le rendu **HTML**.
- **SQLite3** est une bibliothèque qui propose un moteur de base de données relationnelle accessible par le langage **SQL** et qui nous permet donc d'interroger facilement notre base de données.

Pour la partie Front-end nous avons utilisé **HTML** et **CSS** ainsi que le framework **Bootstrap** pour la création du design (barre de navigation, formulaires, tableaux...).

2.2.2 Fonctionnalités

Le fichier `app.py` contient toutes les fonctions créées avec **Flask** et **SQLite3**. Pour lancer l'application, il suffit d'entrer l'adresse `http://127.0.0.1:5000/X` dans votre navigateur web - après l'installation et la configuration de **Flask** bien entendu (*voir le fichier `README.md` du dépôt Git*) où X est l'une des 'routes' suivantes :

- `ecole` : Affiche l'ensemble des écoles du Concours Mines-Télécom, voir l'exemple ci-dessous.
- `établissement` : Affiche les établissements des candidat, avec le RNE, le type, le nom, le code postal, la ville et le pays de chaque établissement.
- `epreuve` : Affiche la liste de toutes les épreuves du concours avec leurs numéros et leurs types.
- `option` : Affiche la liste des épreuves en option.
- `csp` : Affiche l'ensemble des catégories socioprofessionnelles des parents des candidats.
- `candidatByCode/code` : Affiche l'ensemble des informations du candidat numéro `code`, telles que son code, son nom et prénom, son adresse, sa classe, son établissement, etc.
- `voeux/code` : Affiche la liste des vœux présentés par le candidat numéro `code`.
- `notes/code` : Renvoie la liste des notes obtenues par le candidat numéro `code`.
- `classement/code` : Donne les différents rangs du candidat numéro `code` avec le type correspondant (rang de l'épreuve écrite, de l'épreuve orale, d'admissibilité ou le rang de classe).
- `rech` : Permet de trouver un candidat à partir de son nom. A savoir qu'il n'est pas nécessaire de fournir le nom complet du candidat : Par exemple si on tape `ben` nous obtenons la liste de tous les candidats pour lesquels le nom contient la chaîne de caractères `ben`, voir l'exemple ci-dessous.
- `moyenne/codepreuve/rne` : Permet d'obtenir la moyenne sur l'épreuve numéro `codepreuve` pour les candidats venant de l'établissement numéro `rne`.

Ecole	Etablissements	Epreuves	Options	Csp	Rechercher
-------	--------------------------------	--------------------------	-------------------------	---------------------	----------------------------

RNE	Type	Nom	Code postal	Ville	Pays
81731980	Ecole	Institut Gabriel Hamon	35171	Etienne-sur-Mer	Maroc
L1501	Université	Lycée Susan Rousseau	32844	Saint Patrick	Maroc
80584565	IUT	Lycée Alphonse Maillet	40258	Meunier	Maroc
889973	IUT	Institut Isabelle-Christelle Collet	99450	Monnier	Maroc
5024138H	Université	Lycée Alfred Lebrun	9555	GilletBourg	Maroc
L9854	Ecole	Lycée Tristan Navarro-Hoarau	70180	Weber	France
5789908M	Ecole	Institut Dorothée Pineau du Gomez	74386	Guillaumenec	France
73178729	IUT	Lycée Gilbert Pichon	98235	Wagner	Maroc
1142307G	Université	Institut Guy Hebert-Moreno	82062	Saint Xavier-les-Bains	Maroc
69111701	Lycée	Ecole Supérieure Philippine Girard	58306	ReyBourg	France
L0136	Université	Institut Léon Cousin	23152	Dumontnec	Maroc
037074	Lycée	Ecole Supérieure Hélène Marques	94622	Toussaint	Tunisie
L3350	Ecole	Lycée Antoinette Lefort	30979	Meunier	France

FIGURE 5 – Résultat de <http://127.0.0.1:5000/rech>

Ecole	Etablissements	Epreuves	Options	Csp	Rechercher
<ul style="list-style-type: none"> • BENOIT Marcel • BENARD Zacharie • BENOIT Éric • BENARD Emmanuelle • BENOIT Roland • BENARD Étienne • BENARD Arthur • BENARD Stéphane • BENOIT Yves • BENOIT Luce • BENOIT Xavier • BENARD Martine • BENARD Alexandre • BENOIT Étienne 					

FIGURE 6 – Résultat de la recherche des noms contenant 'ben'

3 Tests de cohérence et de performance

3.1 Cohérence

Après l'importation des données dans la base, nous avons effectué une vérification de la cohérence de celles-ci. Cette vérification est constituée de trois étapes.

Premièrement les contraintes d'intégrité lèvent des erreurs si les données mises dans les tables ne respectent pas certaines caractéristiques.

Deuxièmement nous vérifions l'existence de toutes les clé étrangères. Pour cela nous utilisons la commande `PRAGMA foreign_key_check` présente dans `SQLite` qui renvoie la liste de toutes les clés étrangères problématiques. Lors de l'exécution de cette commande, nous nous sommes aperçus que certains candidats venaient d'établissements non présents dans la table du même nom et que certaines notes et rangs référaient à des candidats inconnus. Nous avons décidé de complété la table établissement pour le premier cas et de supprimer les enregistrements problématiques pour le second. Si d'autres enregistrements réfèrent à des candidats inconnus, ils seront supprimés sinon ils seront affiché sur la sortie standard.

Troisièmement nous vérifions la cohérence des informations entre celles de la base de donnée et celles des fichiers non utilisé pour la remplir. Car, comme certaines données sont redondantes à travers plusieurs fichiers, nous ne les récupérons que dans un seul. Il faut alors vérifier si certaines informations ne sont pas contradictoires entre ces fichiers. Le script qui s'occupe de ça ne fait qu'afficher les éventuelles erreurs.

Le script a relevé une erreur, un candidat se trouvait être 3/2 est peut-être 5/2.

3.2 Performance

Le script qui initialise et remplit la base de donnée met environ 40 secondes à s'exécuter. Cependant, le script qui vérifie la cohérence des données met environ 2 minutes.

4 Gestion de projet

Cette partie est consacrée à la présentation de notre gestion de projet. L'équipe s'est inspirée du cadre méthodologique SCRUM, pour l'organisation du groupe et du projet, voir la table 2, la méthode SMART a été utilisé pour le découpage des tâches, voir la table 3 ; avant le début du projet une analyse des risques en utilisant la matrice SWOT a été effectuée, voir la figure 4.

4.1 Équipe de projet

Ce projet a été réalisé par le groupe 9 dont les membres sont :

- Maël SAILLOT, le développeur responsable du dépôt git, il s'assure de l'organisation du dépôt, il vérifie les lignes de codes (*code review*) et les optimise, il s'occupe de régler les soucis informatique au sein du projet ;
- Céline ZHANG, le développeur chef de projet, elle s'occupe de l'organisation des réunions, du déroulement des réunions, du découpage et de la répartition des tâches, de l'avancée du rapport, et elle se charge de rédiger les comptes rendus de réunions ;
- Ahmed ZIANI, le développeur responsable du contrôle, il s'occupe l'exactitude des notions abordées, vérifie les sources de celles-ci, la bonne application des règles de constructions de la base de données.

Chaque membre est développeur, donc participe à la conception et à l'écriture des codes.

Membre de l'équipe 9	Rôles ou charges
Céline ZHANG	leader
Ahmed ZIANI	reviewer
Maël SAILLOT	responsable du git

TABLE 2 – Tableau des rôles

4.2 Analyse du projet

4.2.1 Définition des objectifs

Les objectifs ont été définis en se basant sur la méthode SMART comme indiqué sur la table 3 :

	Critère	Indicateur
S	Spécifique	Objectif clair, précis et sans ambiguïté
M	Mesurable	Objectif quantifié permettant de mesurer l'état d'avancement
A	Ambitieux et Atteignable	Objectif représentant un défi atteignable et non démotivant
R	Réaliste	Objectif envisageable et suffisamment motivant
T	Temporellement défini	Objectif défini et délimité dans le temps

TABLE 3 – La méthode SMART

4.2.2 Analyse des risques : Matrice SWOT

Nous avons évalué les risques du projet en utilisant la matrice SWOT sur la figure.

Forces	Faiblesses
Expérience en projet d'équipe	Première coopération avec les membres
Compétences en Python et en L ^A T _E X	Consignes ambiguës et incomplets
Quelques connaissances en développement Web	Pas assez de connaissances avancées en Web
Connaissances en SQL	Première réalisation d'une base de données
Opportunités	Menaces
Possibilité de demander de l'aide aux 2A et 3A	Temps réduit avec les partiels
Obtenir l'aide des professeurs	Situation sanitaire

TABLE 4 – Matrice SWOT du projet

4.2.3 Répartition des tâches : Matrice RACI

Nous avons transformé les objectifs en tâches énumérées dans notre matrice RACI.

	Ahmed	Céline	Maël
Phase 1 : Conception de la base de données			
Schémas sur dbdiagram.io	CI	R	RA
Contraintes d'intégrité	R	CI	RA
Formes normales	R	R	R
Fichier createdb.sql	R	RA	R
Phase 2 : Importation des données et tests			
Scripts d'importation de données	RA	R	R
Regroupements des codes	I	I	RA
Tests de cohérence	CI	CI	RA
Phase 3 : Application Web			
Conception Back-end	RA	R	CI
Conception Front-end	R	RA	R

TABLE 5 – Matrice RACI

Légende

R : réalisateur.

A : avoir l'autorité.

C : conseiller.

I : informé.

4.3 Organisation du projet

4.3.1 Durée

Le projet a commencé 3 Avril 2021 et s'est terminé le 11 Juin 2021, pour une durée de 68 jours. Cette version du rapport a été terminée le 11 Juin 2021.

4.3.2 Le cadre méthodologique SCRUM

Pour une meilleure organisation, nous avons choisi d'adapter une gestion de projet agile avec SCRUM. On a estimé que l'adaptation collective et rapide sera plus productive et plus constructive pour tous les membres de l'équipe.

Par conséquent, nous nous sommes basés sur les 3 piliers fondamentaux de SCRUM :

- Transparence (visibilité concrète sur la situation)
- Inspection (détection des écarts par rapport aux objectifs)
- Adaptation (réctification de ces écarts par rapport aux objectifs)

Conformément à ce cadre, un ordonnancement du product backlog a été effectué, au début. Nous avons divisé les tâches sur des *sprints* fixés afin de pouvoir concevoir, réaliser et tester les nouvelles fonctionnalités au fur et à mesure. En effet, pour chaque *sprint* on fixe un objectif à court

terme et on se lance dans la réalisation. Une fois cet objectif atteint, nous discutons et nous nous adaptons à la situation, à travers les réunions hebdomadaires.

Toutes ces organisations ont été mis en oeuvre dans le tableau Trello¹² comme illustré dans la figure 7.

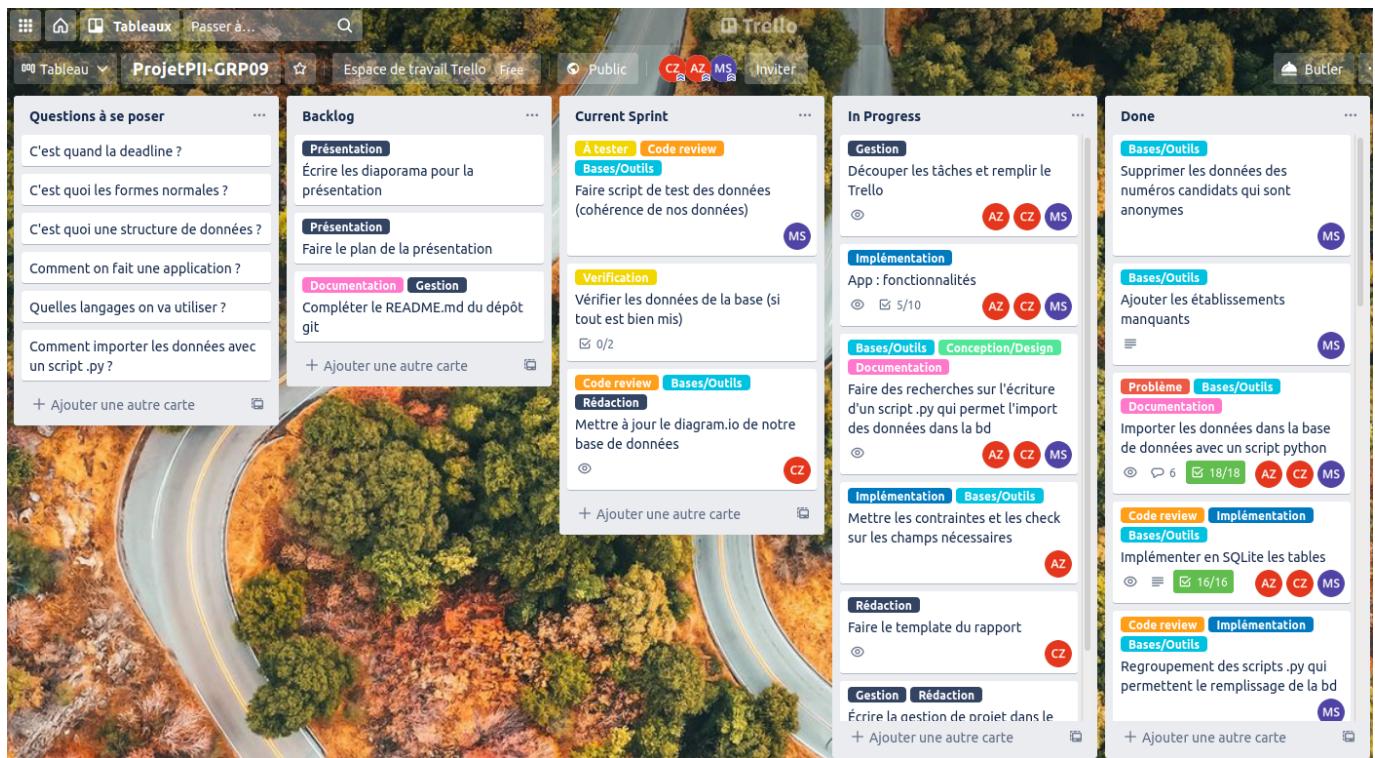


FIGURE 7 – L'organisation du projet sur Trello

L'organisation du tableau Trello était comme suit :

- **Question à se poser :** les interrogations nécessaires ;
- **Backlog :** la liste de tâches priorisées ;
- **Current Sprint :** les tâches à réaliser au cours du *sprint* actuel ;
- **In Progress :** les tâches en cours de réalisation ;
- **Done :** les tâches finies.

Pour chaque tâche, il y a un ensemble d'étiquettes qui permettent de caractériser une tâche facilitant le traitement de celle-ci. On distingue principalement des étiquettes qui précisent la nature ou l'avancement des tâches comme :

- **Documentation :** la phase de documentation ;
- **Conception/Design :** la phase de la conception de l'algorithme.
- **Implémentation :** la phase de l'implémentation de l'algorithme ;
- **Code Review :** une étiquette qui est attribuée à la tâche après l'implémentation pour que les autres développeurs passent vérifier le code ;
- **À tester :** la phase des tests ;
- **Bases/Outils :** production d'élément nécessaire à l'implémentation de l'application ;
- **Vérification :** une tâche de vérification ;

12. <https://trello.com/b/A40HOVP4/projetppi-grp09>

- **Problème** : la tâche rencontre un problème ;
- **Gestion** : la tâche est en lien avec la gestion du projet ;
- **URGENT** : la tâche doit être traité très rapidement ;
- **Rédaction** : la tâche est en relation avec la rédaction du rapport ;
- **Présentation** : la tâche est en relation avec la réalisation de la présentation.

4.4 Outils de travail

4.4.1 Communication

Les principaux outils de communication sont : Discord, Messenger.

Discord a été utilisé pour communiquer en messages, pour organiser des réunions, montrer les codes, diffuser des explications, et éventuellement envoyer des documents.

Messenger est principalement utilisé pour la communication des réunions, l'échange des disponibilités et la discussion des particularités des tâches lors des *sprint* en dehors des réunions, pour régler un problème ou poser une question rapidement par exemple.

4.4.2 Programmation

Selon les préférences de chaque membre, plusieurs outils pour la programmation ont été utilisés.

- Les langages et les modules : Python, Pandas, Openpyxl, Flask.
- Les IDE : PyCharm, Visual Studio Code.
- Les consoles/logiciels : shell, PowerShell, SQLite3, SQLiteStudio.
- Représentation de la base de données : dbdiagram.io¹³

4.4.3 Partage du travail

Tout au long du projet, nous avons utilisé essentiellement le dépôt GitLab fournit par l'école. Nous avons travaillé sur plusieurs branches. Notre dépôt a plusieurs répertoires pour organiser nos fichiers :

- **src** pour les fichiers de code ;
- **tests** pour les fichiers test ;
- **app** pour les applications des algorithmes ;
- **rapport** pour enregistrer les résultats des applications ;
- **fichiers** où on ajoute les fichiers .xlsx et .csv pour le remplissage de la base de données et les applications.

4.4.4 Rédaction du rapport

Le rapport a été rédigé sur Overleaf permettant aux membres de modifier leurs parties simultanément, cela facilite la compilation du rapport (éitant les soucis de packages ou de versions). Un répertoire **rapport** a été mis en place aussi sur le dépôt Gitlab dans lequel est rassemblé l'ensemble des résultats des applications.

13. <https://dbdiagram.io/home>

4.5 Les réunions de projet

Les réunions ont été essentielles pour planifier les *sprints*, régler les problèmes rencontrés. Nous avons eu 9 réunions d'équipe sur Discord (représantant une durée totale de 12 heures) qui sont listées dans le tableau 6. Des comptes rendus de réunion ont été écrites, ils se trouvent dans les annexes. Par ailleurs quelques *stand-up-meeting* ont été fait en début de projet et pendant les périodes de partiels à l'école **Télécom Nancy**.

Date	Durée	Lieu
3 Avril 2021	1h45	Discord
6 Avril 2021	1h30	Discord
11 Avril 2021	1h00	Discord
25 Avril 2021	1h40	Discord
29 Avril 2021	1h10	Discord
3 Mai 2021	1h00	Discord
14 Mai 2021	0h40	Discord
24 Mai 2021	1h30	Discord
4 Juin 2021	1h30	Discord

TABLE 6 – Les réunions de groupe

Conclusion

Pour conclure le projet, la structure de données créé permet de pouvoir stocker et accéder aux données facilement par un utilisateur. Le choix de faire une base de données de troisième forme normale, au minimum, permettait réduire les redondances. Il est important pour une base de données de garder l'unicité des données, pour cela on stock une donnée qu'une fois. De plus, l'ajout des contraintes sont nécessaires pour vérifier, la cohérence des données à chaque ajout. Après avoir conçu le modèle de données, un script de remplissage automatique a été écrit pour remplir cette base de données rapidement par les fichiers classeurs fournis. Tout de même, plusieurs erreurs ont été retrouvé lors du remplissage, comme des candidats anonymes (sans nom, ni prénom), des candidats non-inscrits, des établissements d'enseignement manquants, etc. Ces données ont été supprimées pour ne garder que les données de candidats identifiés. Par ailleurs, un test de cohérence a été réalisé pour vérifier la cohérence des données fournies au départ. Une erreur a été trouvé, un candidat était à la fois 3/2 et 5/2, sinon l'ensemble des données sont cohérentes.

Une fois notre base de données remplie, il était important d'avoir une application Web qui permettait de faire l'intermédiaire entre la base de données et un utilisateur grand public. L'application Web réalisé permettait de faire des recherches sur un candidat, d'afficher les résultats et surtout de les sélectionner ou de les filtrer, par exemple, l'affichage statistique. Ainsi, elle permet à l'utilisateur de ne voir que les informations qui l'intéressent simplement.

Sur l'exécution des scripts de remplissage et de test, il est important que le code s'exécute et se termine rapidement. Le but est d'avoir un moyen de remplir rapidement et efficacement la base de données. Le remplissage se fait en 40 secondes, et la vérification des données se fait en 2 minutes. Il est normal que la vérification soit plus longue que le test de cohérence, car celle-ci de accéder à toutes les données et les comparer à leur 'doublons'.

La base de données est créé, remplie, l'application est implantée et gère cette base, donc dans l'ensemble les objectifs du projet ont été atteints, bien qu'il y a eu des zones floues dans le sujet. L'équipe adopté la méthode de gestion agile pour ce projet, ceci a permis de se fixer des objectifs à atteindre chaque semaine. Toutes les une à deux semaines, une réunion est organisée pour suivre l'avancement du projet et supprimer tous les points bloqueurs. La progression était régulière, le travail était séiruex, l'équipe se félicite pour ces résultats.

Bilan global du projet d'équipe

Le point sur le projet

Travaux demandés	Travaux réalisés
Base de données du concours Mines-Télécom	Schéma relationnel de la base de données
Respect de la 3e forme normale	Script de remplissage de la base de données
Application de gestion de la base de données	Respect de la 3e forme normale Fonctionnalités app web : Accès info candidat (coordonnées, résultats, etc.) ; Liste des épreuves, options, écoles, voeux, etc. ; Statistiques (moyennes, rang) Ajout de style pour enrichir l'apparence de l'app web
Cohérence des données	Script de test de cohérence
Performance remplissage < 3 min	Temps de remplissage de la BD < 1 min

TABLE 7 – Le bilan du projet

Le point sur l'équipe

Points positifs	Expérience en travail d'équipe Application des notions vues en cours dans des situations concrètes Amélioration des compétences de programmation en Python Application des connaissances en gestion de projet et en WEB Progression des compétences de tests Bases en Python Harmonie de l'équipe Participation active de chacun des membres Réunions régulières Accessibilité des ressources en ligne
Points négatifs	Sujet pas complet, certains aspects ne sont pas précisé Incohérences dans les données fournis Données longues à traiter Coupure du projet dû à la période d'examens Couver-feu (travail essentiellement à domicile en autonomie)
Difficultés	Autonomie L'utilisation du <code>sqlite3</code> , <code>flask</code> , <code>openpyxl</code> et <code>pandas</code> Rédaction du rapport
Améliorations possibles	Enrichir l'application web d'interactions Optimiser le stockage mémoire en rajoutant des tables Défier la FNBC (Boyce-Codd) et d'autres FN plus avancées Augmenter la vitesse d'exécution en écrivant en C Réduire la taille du programme en utilisant que <code>openpyxl</code> Rédaction du rapport

TABLE 8 – Le bilan d'équipe

Thèmes	Maël SAILLOT	Céline ZHANG	Ahmed ZIANI
Documentation	6h	5h	9h
Conception	10h	7h	8h
Implémentation	20h	11h	12h
Code Review	8h	4h	8h
Tests et performances	12h	2h	2h
Rédaction des documents	8h	29h	20h
TOTAL	64h	58h	59h

TABLE 9 – Le temps moyen consacré au projet de chaque membre

Annexes

Les déclarations sur l'honneur de non-plagiat

Déclaration sur l'honneur de non-plagiat

Je, soussigné,

Nom, prénom : SAILLOT, Maël

Étudiant ingénieur inscrit en 1ère année à TELECOM Nancy

Numéro de carte étudiante : 31831300

Année universitaire : 2020 - 2021

Auteur, en collaboration avec ZHANG Céline, ZIANI Ahmed, du rapport :

Application de gestion de données de concours Mines-Télécom

Je déclare sur l'honneur que ce rapport est le fruit d'un travail personnel et que je n'ai ni contrefait, ni falsifié, ni copié tout ou partie de l'œuvre d'autrui, en particulier, texte ou code informatique, afin de la faire passer pour mienne.

Je certifie donc que le travail rendu est un travail original et que les sources utilisées, notamment pour les formulations, les idées, les documentations, les raisonnements, les analyses, les schémas ou autres créations ont été mentionnées conformément aux usages en vigueur.

Je suis conscient que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l'Université et qu'il peut être sévèrement sanctionné.

Fait à Nancy, le 02/06/2021
Signature de l'étudiant :



Déclaration sur l'honneur de non-plagiat

Je, soussignée,

Nom, prénom : ZHANG, Céline

Étudiant ingénieur inscrit en 1ère année à TELECOM Nancy

Numéro de carte étudiante : 32024925

Année universitaire : 2020 - 2021

Auteur, en collaboration avec SAILLOT Maël, ZIANI Ahmed, du rapport :

Application de gestion de données de concours Mines-Télécom

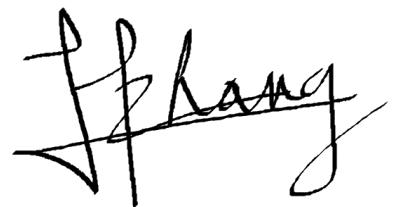
Je déclare sur l'honneur que ce rapport est le fruit d'un travail personnel et que je n'ai ni contre-fait, ni falsifié, ni copié tout ou partie de l'œuvre d'autrui, en particulier, texte ou code informatique, afin de la faire passer pour mienne.

Je certifie donc que le travail rendu est un travail original et que les sources utilisées, notamment pour les formulations, les idées, les documentations, les raisonnements, les analyses, les schémas ou autres créations ont été mentionnées conformément aux usages en vigueur.

Je suis consciente que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l'Université et qu'il peut être sévèrement sanctionné.

Fait à Nancy, le 02/06/2021

Signature de l'étudiant :



Déclaration sur l'honneur de non-plagiat

Je, soussigné,

Nom, prénom : ZIANI, Ahmed

Étudiant ingénieur inscrit en 1ère année à TELECOM Nancy

Numéro de carte étudiante : 31824316

Année universitaire : 2020 - 2021

Auteur, en collaboration avec SAILLOT Maël, ZHANG Céline, du rapport :

Application de gestion de données de concours Mines-Télécom

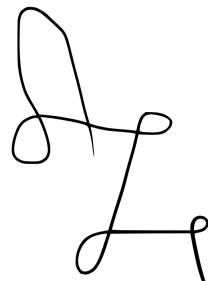
Je déclare sur l'honneur que ce rapport est le fruit d'un travail personnel et que je n'ai ni contre-fait, ni falsifié, ni copié tout ou partie de l'œuvre d'autrui, en particulier, texte ou code informatique, afin de la faire passer pour mienne.

Je certifie donc que le travail rendu est un travail original et que les sources utilisées, notamment pour les formulations, les idées, les documentations, les raisonnements, les analyses, les schémas ou autres créations ont été mentionnées conformément aux usages en vigueur.

Je suis conscient que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l'Université et qu'il peut être sévèrement sanctionné.

Fait à Nancy, le 02/06/2021

Signature de l'étudiant :



Trello

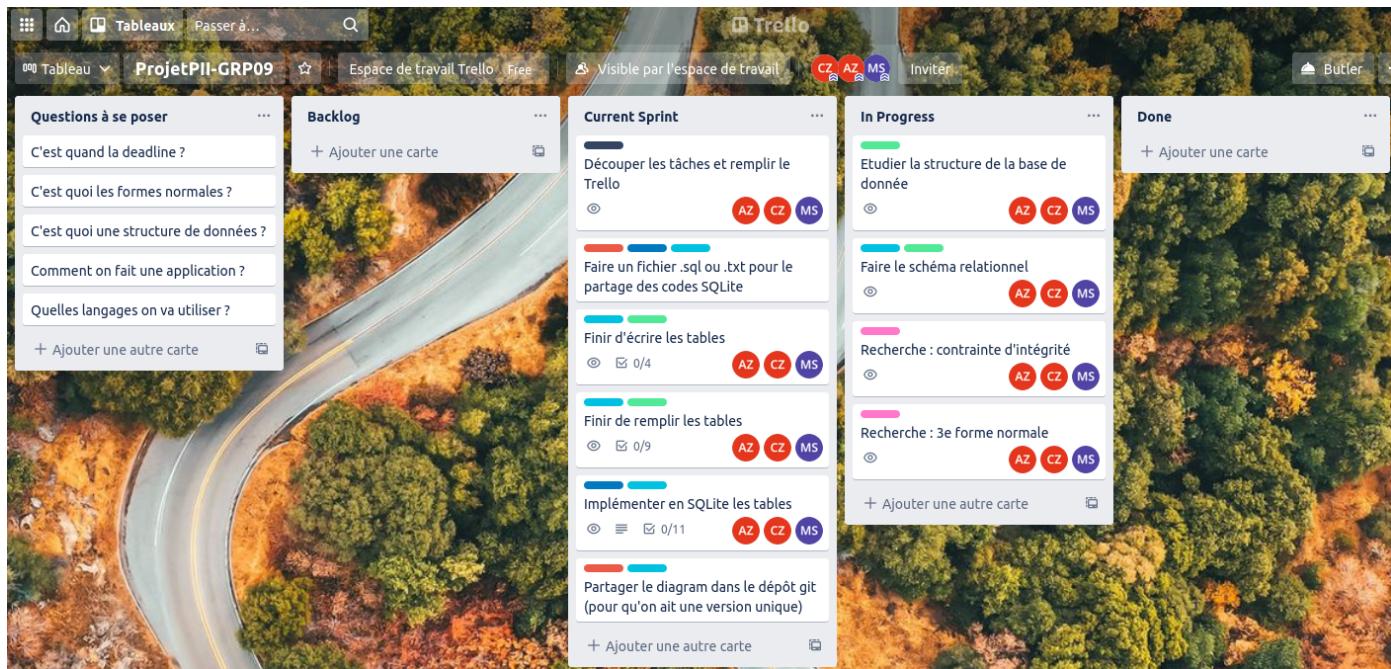


FIGURE 8 – L'organisation du Trello le 11 avril 2021

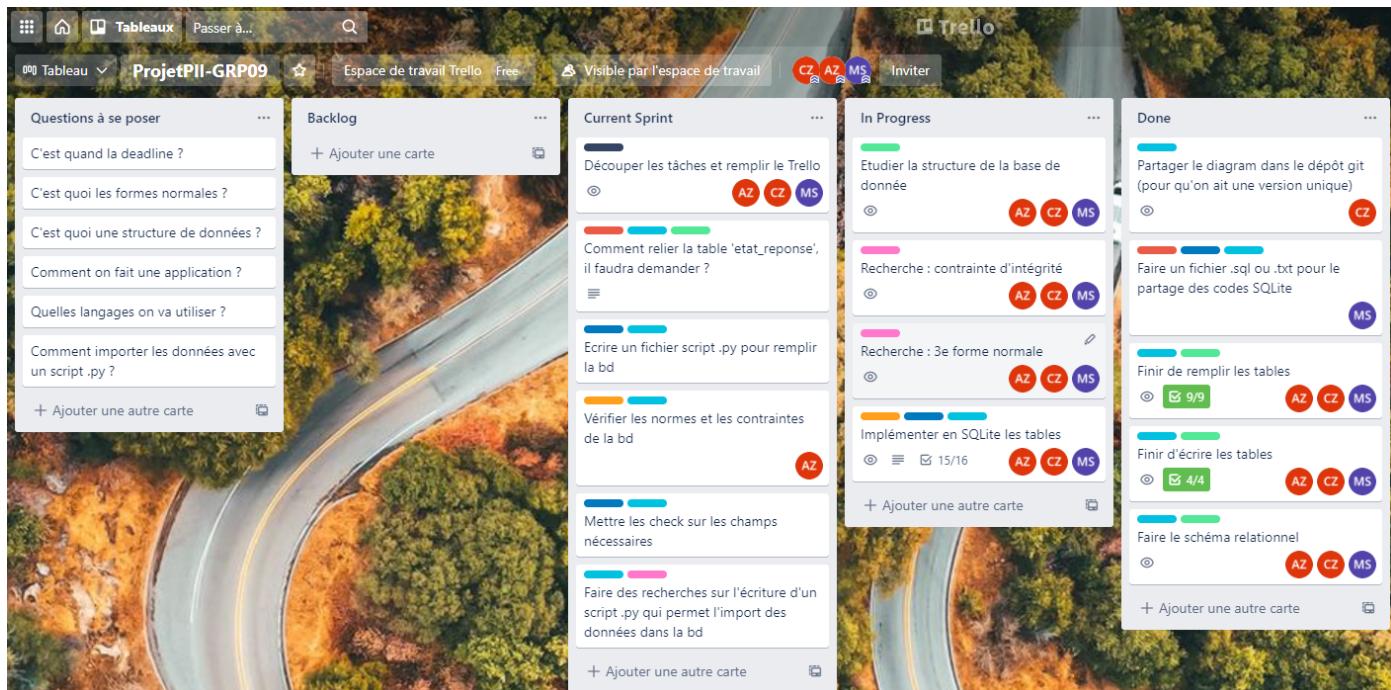


FIGURE 9 – L'organisation du Trello le 25 avril 2021

13. <https://trello.com/b/A4OH0VP4/projetpii-grp09>

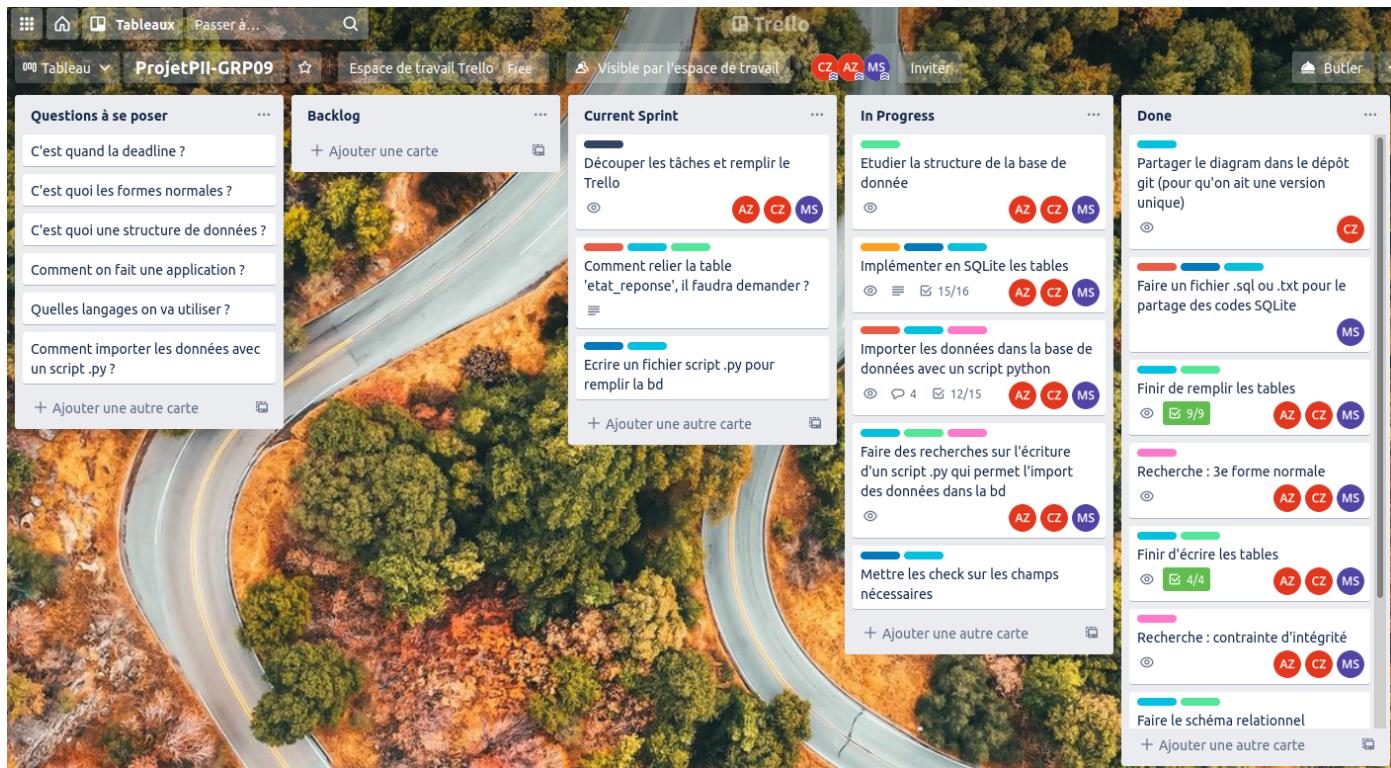


FIGURE 10 – L'organisation du Trello le 12 mai 2021

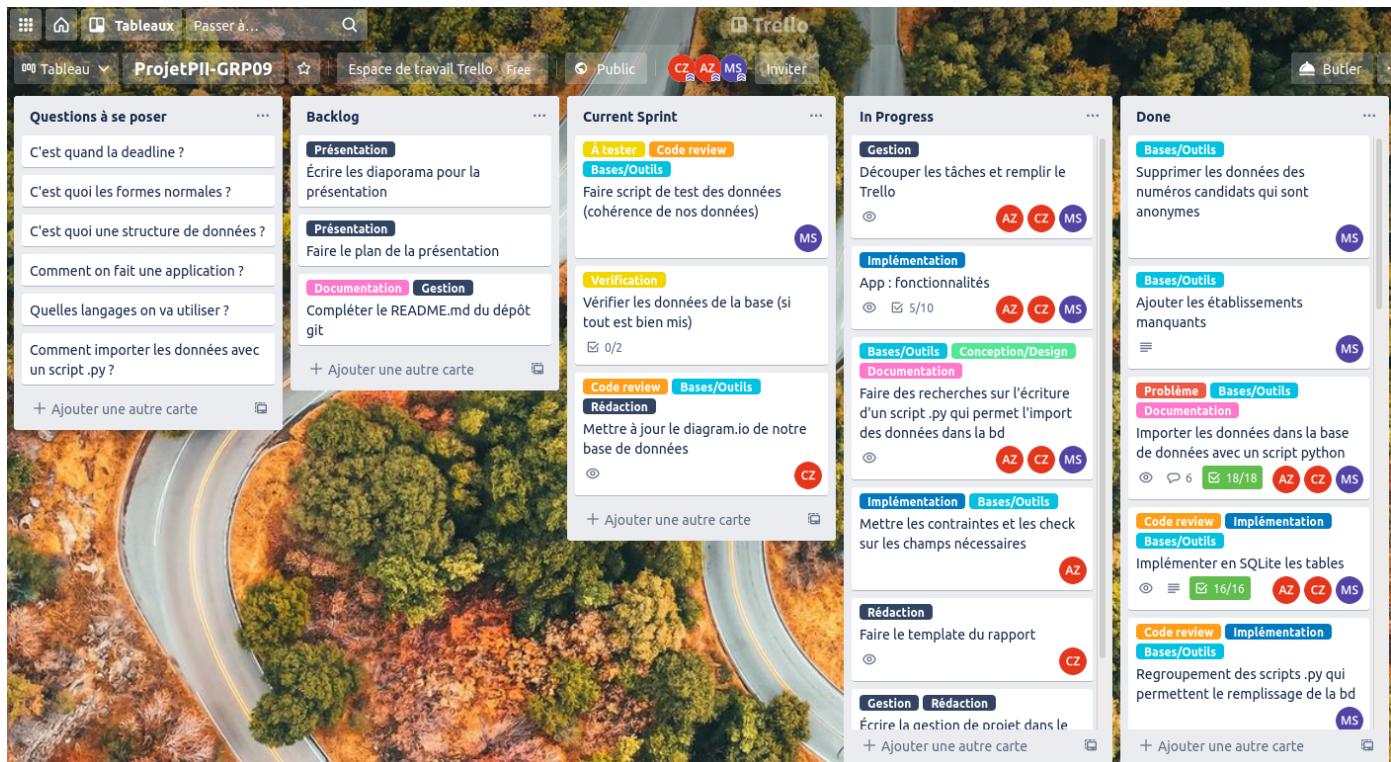


FIGURE 11 – L'organisation du Trello le 2 juin 2021

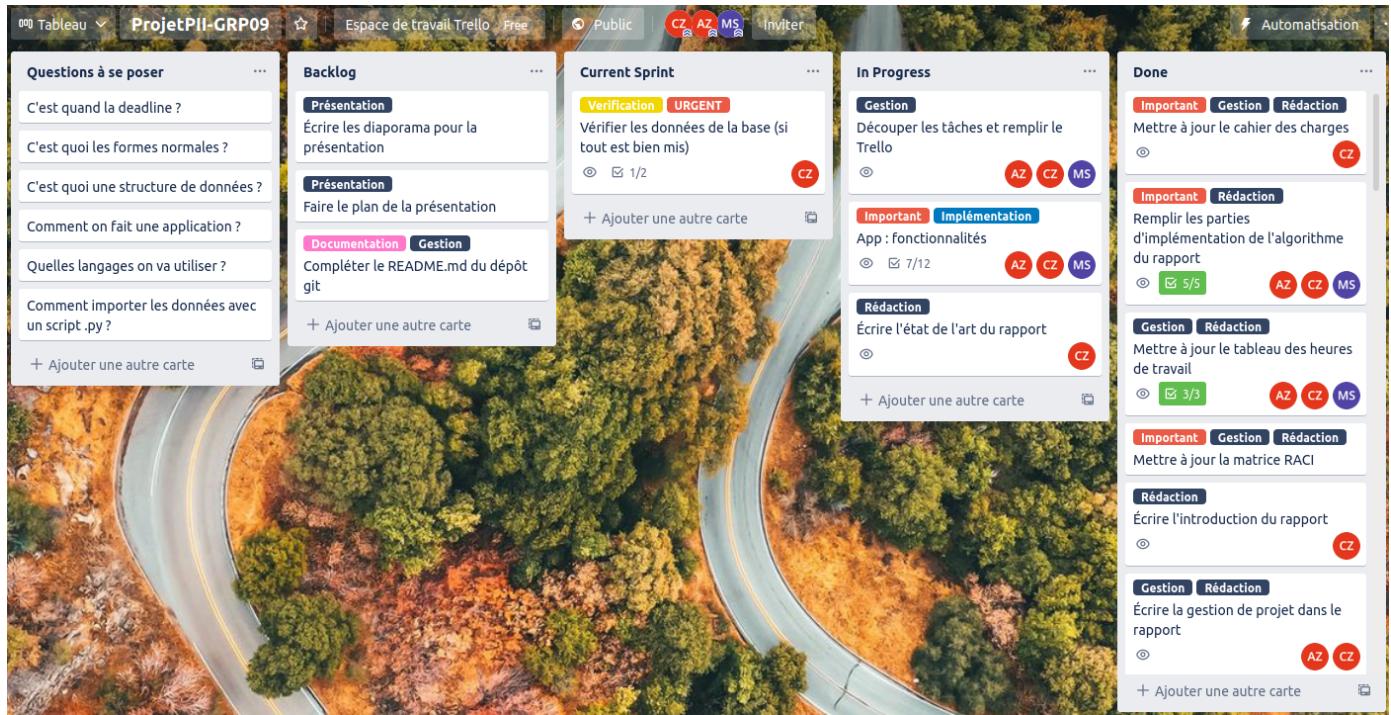


FIGURE 12 – L'organisation du Trello le 11 juin 2021

Comptes rendus de réunions

Réunion d'équipe du 3 avril 2021

Membres présents	Membres absents	Durée	Lieu
Maël SAILLOT Céline ZHANG Ahmed ZIANI		1h45	Discord

Ordre du jour

1. Mise à niveau générale sur les connaissances du sujet
2. Organisation du projet
3. Sélection des outils nécessaires au projet
4. Mise en place de la gestion de projet
5. Planification des prochaines réunions

Mise à niveau générale sur les connaissances du sujet

Nous avons mis en commun nos connaissances concernant le sujet, partagé nos documentations, et expliqué les notions ambiguës. Ceci s'est suivi d'une présentation de nos préférences, de nos points forts et points faibles respectives.

Organisation du projet

Nous avons parlé du sens de déroulement général du projet (quand faire les documentations, dans les premières lignes, quand aborder les parties, quand commencer la rédaction du rapport, etc.). Nous avons discuté d'une gestion de projet (les stratégies, les rôles, voir le tableau 2, les facilités, les difficultés par la matrice de SWOT, voir figure 4). Nous adoptons la gestion SCRUM, qui utilise la méthode du *sprint*¹⁴ pour réaliser les travaux.

Membre de l'équipe 9	Rôles ou charges
Céline ZHANG	leader
Ahmed ZIANI	reviewer
Maël SAILLOT	responsable du git

Céline ZHANG se chargera de vérifier les tâches, les objectifs, de planifier les réunions et d'écrire les compte-rendus de réunions d'équipe.

Ahmed ZIANI se chargera de revoir nos lignes de code (code review), de vérifier la cohérence et les tests.

Maël SAILLOT s'occupera de gérer le dépôt git de l'équipe, de gérer les erreurs dû à des conflits.

¹⁴. Méthode d'organisation de travail d'équipe qui consiste à se fixer des tâches pour un cycle (1 à 2 semaines) et de les finir avant la fin de celle-ci.

Sélection des outils nécessaires au projet

La présentation des outils utilisables et les outils nécessaires est primordiale pour commencer à travailler. Nous avons donc présenté les outils à disposition ([gitlab](#), [Discord](#), [SQLite](#), [Overleaf](#), [dbdiagram.io](#), *C*, *Java*, *Python*, *html*).

Mise en place de la gestion de projet

Nous avons choisi d'utiliser le tableau de bord Trello¹⁵ qui va nous permettre de mettre le cahier de charge en tâches nécessaires à la réalisation du livrable final. Ceci nous permet aussi de surveiller l'avancement de chacun sur ses tâches. Nous avons commencé par créer le Trello et y mettre des tâches.

Planification des prochaines réunions

Nous décidons de faire en général une réunion par semaine (flexible suivant les disponibilités), qui sera tous les dimanches à 17h00 sur Discord, bien sûr il peut y avoir des stand-up-meeting pour régler les soucis ou prendre des nouvelles (le tableau Trello permet aussi de laisser des messages pour indiquer les problèmes et demander de l'aide).

TO-DO LIST

- Se documenter sur première partie du sujet concernant les notions de 3e forme normale, les contraintes d'intégrités
- Réfléchir à la structure de la base de données
- Faire un schéma liant les tables

Prochaine réunion : 06/04/2021

15. Outil qui nous permet de planifier en ligne des activités.⁷

Réunion d'équipe du 6 avril 2021

Membres présents	Membres absents	Durée	Lieu
Maël SAILLOT Céline ZHANG Ahmed ZIANI		1h30	Telecom Nancy

Ordre du jour

1. Avancement des tâches
2. Mise en commun des propositions
3. Optimisation des solutions proposées
4. Choix des prochaines tables

Avancement des tâches

Les membres sont venus avec leur schéma d'une structures des données. Nous avons pris connaissance de chacun de ces schémas qui étaient globalement similaires.

Mise en commun des propositions

Nous avons produit un schéma pour l'implémentation du diagram sur une interface graphique dbdiagram.io.

Optimisation des solutions proposées

En prenant en compte les contraintes d'intégrités et en respectant les règles de la 3e forme normale, nous avons choisi de mettre les principales informations de l'inscrit dans une table **candidat** (comme civilité, nom, prénom, date et lieu de naissance, coordonnées, rang, classe, filière concours, etc.) et certainement dans une deuxième table, les options du candidat. De plus des tables moins grandes ont été faites, comme les tables **établissement**, **voeux**, **classement**, **notes**, **épreuve**, etc. Le numéro du candidat sera une clé primaire, et servira de foreign key pour certaines des tables suivantes.

Choix des prochaines tables

Nous allons sûrement faire une table **autres_info**, **pays**, **concours**, etc. le nécessaire pour compléter la base de données.

Planification des prochaines tâches

L'équipe devra écrire ces tables sur dbdiagram.io, compléter les champs manquants, et ajouter éventuellement les tables manquantes ; mettre sur Trello les tâches.

TO-DO LIST

- Écrire les tables sur dbdiagram.io
- Compléter la structure de bases de données (les tables et les champs manquants, etc.)
- Mettre les tâches sur le Trello

Prochaine réunion : 11/04/2021

Réunion d'équipe du 11 avril 2021

Membres présents	Membres absents	Durée	Lieu
Maël SAILLOT Céline ZHANG Ahmed ZIANI		1h	Discord

Ordre du jour

1. Avancement des tâches
2. Discussion générale des avis sur la structure actuelle
3. Optimisation des solutions proposées
4. Prochaines tâches

Avancement des tâches

Maël SAILLOT a écrit dans le [dbdiagram.io](#) la première version de la structure de la base de données et l'a partagé aux autres membres.

Céline ZHANG a complété une partie des champs manquants dans les tables et ajouté une partie des tables manquantes.

Ahmed ZIANI a proposé une solution en faisant des tables `admissible_XX`, `admis_XX` pour chaque filière.

Discussion générale des avis sur la structure actuelle

La proposition des tables `admissible_XX`, `admis_XX` pour chaque filière a finalement été rejeté, puisqu'il y aurait redondance des données, par ailleurs, il se trouvait déjà dans le champ `type` (qui n'avait pas été explicité, d'où le malentendu avec les tables `admissible_XX`, `admis_XX`).

Optimisation des solutions proposées

Nous avons vérifié les relations entre les tables, et corrigé quelques imprecision des champs. Nous avons rajouté des notes et restrictions pour certains champs. Après discussion, nous avons choisi de mettre une partie des info candidat dans candidat, et une autre partie certainement dans une autre table, elles seront liées par une relation 1 1.

Prochaines tâches

L'équipe devra compléter les tables, et écrire les tables restantes sur [dbdiagram.io](#). Ensuite, chaque membre fera 3 à 4 tables en SQLite, et mettra leur code sur un fichier `.sql` (ou `.txt`) partagé sur le [git](#). Chacun sera libre de choisir les tables qu'il fera, mais il devra indiquer sur le Trello son choix et *push* son travail pour tenir au courant les autres membres. Une description détaillée de ce qui est à faire se trouve sur le Discord de l'équipe, dans le canal `#todo`.

TO-DO LIST

- Finir de mettre les tables manquantes (voir le canal `#todo` sur le Discord de l'équipe)
- Finir de remplir les champs des tables (voir le canal `#todo` sur Discord)
- Implémenter ces tables en SQLite et partager les codes sur le dépôt `git`
- Mettre à jour le Trello

Prochaine réunion : 25/04/2021

Réunion d'équipe du 25 avril 2021

Membres présents	Membres absents	Durée	Lieu
Maël SAILLOT Céline ZHANG Ahmed ZIANI		1h40	Discord

Ordre du jour

1. Avancement des tâches
2. Discussion générale et amélioration de la structure actuelle
3. Prochaines tâches

Avancement des tâches

Maël SAILLOT a ajouté les tables, les champs manquants, a créé les fichiers `createdb.sql` et `schemadb.txt` pour le partage.

Céline ZHANG a revu les codes du schéma de `dbdiagram.io`, a commencé à remplir le fichier `createdb.sql`.

Ahmed ZIANI a relu les travaux de l'équipe.

Discussion générale et amélioration de la structure actuelle

Nous avons relu chaque ligne de schéma, nous avons fixé les contraintes nécessaires sur certains champs, et corrigé les erreurs. Les codes postaux doivent être en `TEXT` car les adresses étrangères ont des lettres dans leur code postal, idem pour les numéros particulier comme +33(0).

Prochaines tâches

L'équipe devra compléter le fichier `createdb.sql`, ajouter les contraintes, vérifier le respect des formes normalisés et les contraintes (`review`), et se renseigner sur l'écriture d'un script en `Python` qui permettra de remplir la base de données de manière automatique.

TO-DO LIST

- Remplir le fichier `createdb.sql` et tester l'exécution de celui-ci
- Ajouter les `CHECK` nécessaires (voir le canal `#todo` sur Discord)
- Partager fichier `.py` sur le dépôt `git`
- Se documenter sur l'écriture du script en `.py`
- Écrire un exemple de fonction dans le script pour l'import des données

Prochaine réunion : 29/04/2021

Réunion d'équipe du 29 avril 2021

Membres présents	Membres absents	Durée	Lieu
Maël SAILLOT Céline ZHANG Ahmed ZIANI		1h10	Discord

Ordre du jour

1. Avancement des tâches
2. Discussion générale et amélioration du travail actuel
3. Prochaines tâches

Avancement des tâches

Globalement chaque membre a fait ses recherches pour trouver un moyen de remplir la base de données à partir des fichiers .xlsx donnés.

Maël SAILLOT a fait un exemple de script pour le remplissage de la table des `etat_reponse` en utilisant la bibliothèque `openpyxl` et a fait deux versions, l'une utilisant `click`, l'autre sans.

Céline ZHANG a fait un exemple de script pour le remplissage à partir du fichier `Inscription.xlsx` en utilisant la bibliothèque `pandas`.

Ahmed ZIANI a fait un exemple de script pour le remplissage de la table des `etat_reponse` en utilisant la bibliothèque `openpyxl` et a essayé avec `pandas` aussi.

Discussion générale et amélioration du travail actuel

Nous avons relu le fichier `createdb.sql`, nous constatons qu'il fallait ajouter des `CHECK`. Nous avons choisi d'utiliser principalement la bibliothèque `pandas` bien qu'elle est plus coûteuse en mémoire par rapport à `openpyxl` (elle permet la lecture des titres de colonne), nous utiliserons également `openpyxl` pour d'autres cas.

Prochaines tâches

L'équipe devra écrire les algorithmes de remplissage des tables de la base de données en faisant en priorité les tables qui ont le moins de dépendances par rapport aux autres tables. Il faudra vérifier les contraintes des champs et réfléchir sur les ATS.

TO-DO LIST

- Écrire les algorithmes de remplissage selon les priorités et la *checklist* sur Trello
- Vérifier les contraintes et les tables
- Réfléchir au cas des ATS

Prochaine réunion : 03/05/2021

Réunion d'équipe du 3 mai 2021

Membres présents	Membres absents	Durée	Lieu
Maël SAILLOT Céline ZHANG Ahmed ZIANI		1h	Discord

Ordre du jour

1. Avancement des tâches
2. Discussion générale et amélioration du travail actuel
3. Prochaines tâches

Avancement des tâches

Maël SAILLOT a rempli la table `etat_reponse`.

Céline ZHANG a rempli les tables `etat_dossier`, `concours`, `autres_prenoms`, `serie_bac`, `ep_option`.

Ahmed ZIANI a rempli les tables `ecole`, `établissement`, `pays`, `csp_parent`.

Discussion générale et amélioration du travail actuel

Pour le remplissage de `epreuve`, il a fallu un dictionnaire. Nous avons choisi de réutiliser les codes proposées par les fichiers comme clé primaire et nous avons ajouté certains code pour les champs qui n'en possédaient pas. Les codes ajoutés commence à partir de 9000, par exemple 9898 pour la `bonification_ecrit`. Nous avons remarqué que les champs `etat_classes` et `type_admissible` n'étaient pas dans nos tables de la base de données, il a fallu les ajouter. Certains champs manquaient des `NOT NULL`, il faut les ajouter. Concernant les ATS, certains candidats n'avaient pas de données pour certains champs, nous avons choisi d'ignorer les candidats anonymes (qui n'ont ni nom, ni prénom).

Prochaines tâches

L'équipe devra finir d'écrire les algorithmes de remplissage des tables de la base de données (en faisant attention aux nouveaux champs ajoutés), elle devra ajouter les `NOT NULL` nécessaire, et commencer le remplissage des candidats d'ATS.

TO-DO LIST

- Finir le remplissage des tables de la `checklists` sur Trello
- Ajouter les `NOT NULL` et vérifier les contraintes
- Commencer le remplissage pour les ATS

Prochaine réunion : 14/05/2021

Réunion d'équipe du 14 mai 2021

Membres présents	Membres absents	Durée	Lieu
Maël SAILLOT Céline ZHANG Ahmed ZIANI		40min	Discord

Ordre du jour

1. Avancement des tâches
2. Discussion générale et amélioration du travail actuel
3. Prochaines tâches

Avancement des tâches

Maël SAILLOT a vérifié les contraintes et le dépôt git.

Céline ZHANG a rempli les tables `epreuvre`, `notes`, `classement`, a écrit les dictionnaires des épreuves et des classements nécessaire à l'algorithme de remplissage.

Ahmed ZIANI a rempli la table `voeux`.

Discussion générale et amélioration du travail actuel

La plupart des tables ont été remplies, cependant les ATS n'ont pas été ajoutés, il manque également la remplissage de la table `candidat`. Pour l'instant chacun a son `script`, les tests ont été fait manuellement lors de l'import des données dans les tables de la base de données. Il faut maintenant les regrouper dans un `script` et optimiser le code de chacun. De plus, pour vérifier la cohérence des données, il est nécessaire de faire un `script` de test.

Prochaines tâches

L'équipe devra remplir la table `candidat` avec vérification manuelle, puis faire un `script` pour vérifier la cohérence des données importer avec la bd et les fichiers. Il faudra écrire les algorithmes de remplissage pour les ATS, et regrouper tous les codes dans un `script` qui rempli en une fois toute la base de données.

TO-DO LIST

- Remplir la table `candidat`
- Ajouter les données des candidats ATS
- Faire un `script` qui regroupe tous les codes
- Commencer un `script` de test

Prochaine réunion : 24/05/2021

Réunion d'équipe du 24 mai 2021

Membres présents	Membres absents	Durée	Lieu
Maël SAILLOT Céline ZHANG Ahmed ZIANI		1h30	Discord

Ordre du jour

1. Avancement des tâches
2. Discussion générale et amélioration du travail actuel
3. Prochaines tâches

Avancement des tâches

Maël SAILLOT a rempli la table `candidat`, a ajouté les ATS.

Céline ZHANG a corrigé les contraintes qui se trouvaient dans `createdb.sql`.

Ahmed ZIANI a vérifié et a ajouté les contraintes nécessaires.

Discussion générale et amélioration du travail actuel

En remplaçant les ATS et la table `candidat`, nous remarquons que certains établissements ne se trouvent pas dans le fichier `listeEtablissements.xlsx` alors qu'ils se trouvent dans les informations des candidats. De plus, certains candidats avaient des notes ou un classement, mais ne se trouvaient pas dans la table `candidat`, ils ne sont pas dans le fichier `Inscription.xlsx`. Les TSI dans le fichier `Ecrit_TSI.xlsx` n'ont pas de rang. L'équipe a choisi de partir sur une application web écrit en Python en utilisant le module `Flask`.

Prochaines tâches

L'équipe devra supprimer toutes les données engendrant des incohérences (notamment ceux des candidats anonymes). Elle devra également ajouter les établissements manquants. Une vérification de l'ensemble des données sera nécessaire pour être sûr que rien n'a été négligé.

TO-DO LIST

- Supprimer les candidats anonymes
- Ajouter les établissements manquants
- Regrouper les scripts en un seul script optimisé
- Faire le script de test pour vérifier la cohérence des données
- Commencer l'écriture de l'application web

Prochaine réunion : 04/06/2021

Réunion d'équipe du 4 juin 2021

Membres présents	Membres absents	Durée	Lieu
Maël SAILLOT Céline ZHANG Ahmed ZIANI		1h30	Discord

Ordre du jour

1. Avancement des tâches
2. Discussion générale et amélioration du travail actuel
3. Prochaines tâches

Avancement des tâches

Maël SAILLOT a regroupé les scripts de remplissage pour en faire un seul script optimisé, a commencé le script de test de cohérence entre les fichiers et a mis à jour le schéma de la base de données.

Céline ZHANG a fait le template du rapport, a fini l'écriture de l'introduction et de la gestion de projet, a ajouté les comptes rendus et les mentions légales, ainsi que d'autres annexes.

Ahmed ZIANI a fait la matrice SWOT, RACI et a fait les premières fonctionnalités de l'application de web tels que l'affichage des données d'un candidat, la liste des épreuves, des options, des professions, des classements, des notes, etc.

Discussion générale et amélioration du travail actuel

Il faudra vérifier les fichiers, et penser à faire un script de test de remplissage. Il manque pour l'application les fonctionnalités de mises en stats, d'affichage de graphes ou de cartes et les interactions permettant de rendre l'application plus facile d'utilisation.

Prochaines tâches

L'équipe devra écrire l'état de l'art, finir le script de test de cohérence, ajouter les fonctionnalités stats, cartographique, les styles et l'interface.

TO-DO LIST

- Écrire l'état de l'art
- Finir le script de test de cohérence
- Ajouter les fonctionnalités stats
- Ajouter du style et une interface
- Ajouter la cartographie et éventuellement des graphes

Table des figures

1	Exemple de table non-atomique, l'attribut <code>prénoms</code> viole l'atomicité	7
2	Exemple de table non-2FN, l'attribut <code>jour</code> ne dépend que de l'attribut <code>date</code>	8
3	Exemple de table non-3FN, l'attribut <code>nationalité</code> dépend de l'attribut <code>auteur</code>	8
4	Schéma du modèle de données	10
5	Résultat de <code>http://127.0.0.1:5000/rech</code>	17
6	Résultat de la recherche des noms contenant 'ben'	17
7	L'organisation du projet sur Trello	22
8	L'organisation du Trello le 11 avril 2021	31
9	L'organisation du Trello le 25 avril 2021	31
10	L'organisation du Trello le 12 mai 2021	32
11	L'organisation du Trello le 2 juin 2021	32
12	L'organisation du Trello le 11 juin 2021	33

Liste des tableaux

1	Le cahier des charges	6
2	Tableau des rôles	19
3	La méthode SMART	19
4	Matrice SWOT du projet	20
5	Matrice RACI	21
6	Les réunions de groupe	24
7	Le bilan du projet	26
8	Le bilan d'équipe	26
9	Le temps moyen consacré au projet de chaque membre	27

Références

- [1] Wikipédia. Traitement transactionnel en ligne, Avril 2021. https://fr.wikipedia.org/wiki/Traitement_transactionnel_en_ligne.
- [2] Wikipédia. Forme normale, Avril 2021. [https://fr.wikipedia.org/wiki/Forme_normale_\(bases_de_donn%C3%A9es_relationnelles\)](https://fr.wikipedia.org/wiki/Forme_normale_(bases_de_donn%C3%A9es_relationnelles)).
- [3] Bernard Henri Nicot. Identifiants des communes : Codes insee et codes postaux, Avril 2021. http://www.sirius-upvm.net/doc/usuels/codes_postaux.html.
- [4] Clark Consulting Research. Working with excel files in python, Avril 2021. <http://www.python-excel.org/>.
- [5] The pandas development team. Pandas documentation, Avril 2021. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_sql.html.
- [6] Eric Gazoni Charlie Clark. Openpyxl - a python library to read/write excel 2010 xlsx/xlsm files, Avril 2021. <https://openpyxl.readthedocs.io/en/stable/>.
- [7] The SQLite Consortium. Sqlite, Avril 2021. <https://www.sqlite.org/index.html>.