

Université Paris Est Créteil Val de Marne

Faculté des Sciences

Département d’Informatique

Master Informatique

|  |
| --- |
| **Projet Plateforme Base de données**  Encadrant : M. Bès Alexis |

Par :

Elazbaoui Maryem Abidi Asma

Mlis Maroine Zeggagh Amirouche

Souidi Abdenour Chiadmi Saad

Table des matières

Liste des figures

[Figure 1 : Interface principale de Generatedata.com 3](#_Toc534311066)

[Figure 2 : Génération de du schéma SQL avec les requêtés d’insertions 3](#_Toc534311067)

[Figure 3 : Interface principale de Datanamic Data Generator MultiDB 5](#_Toc534311068)

[Figure 4 : Fichier du script résultat de la génération 6](#_Toc534311069)

[Figure 5 : Interface principale de Datanamic Data Generator MultiDB 7](#_Toc534311070)

[Figure 6 : Diagramme de classes 11](#_Toc534311071)

[Figure 7 : Tableau des tâches à effectuer durant la réalisation 24](#_Toc534311072)

Liste des tableaux

[Tableau 1 : Les possiblités et les limites de Generatedata.com 2](#_Toc534310949)

[Tableau 2 : Les possiblités et les limites de Datanamic Data Generator MultiDB 4](#_Toc534310950)

[Tableau 3 : Les possibilités les limites de Dbforge Data Generator for MySql 6](#_Toc534310951)

[Tableau 4 : Backlog de produit de notre application 22](#_Toc534310952)

[Tableau 5 : Tâches à effectuer dans le premier sprint 23](#_Toc534310953)

# Présentation du contexte et du projet

Ce rapport se compose de trois chapitres. Le premier chapitre présente l’objectif de l’application et les différents éléments la composant, décrit et explique la solution suivie. Le deuxième chapitre s’intéresse à la mise en place d’une solution basée essentiellement sur le développement de l’algorithme de générations de données. Et le troisième chapitre

## Introduction et présentation du projet :

Les bases de données sont omni présentes dans la vie professionnelle que ce soit dans les entreprises, les usines, etc…. Afin de tester ces dernières, les ingénieures se réfèrent à l’écriture de plusieurs requêtes dont la génération de données et effectuer des analyses sur les résultats de ces dernières pour qu’ils puissent vérifier la structure des tables et la cohérence du schéma relationnel de la base de données. La réalisation de ces tâches prend beaucoup de temps et cela à un énorme impact sur le financement.

Les méthodes pour obtenir des données aléatoires existent depuis très longtemps et sont utilisées dans différents domaines. On peut citer :

* La simulation
* L’analyse
* L'échantillonnage
* La prise de décision
* La sécurité informatique (cryptologie) [1].

C’est là que notre application intervient en apportant une solution concrète au problème.

## Objectif :

Le but du projet est de concevoir un outil permettant d'effectuer des tests à partir d'un schéma SQL de bases de données. Il consiste à développer une application qui fournit aux utilisateurs la possibilité de générer différents formats de données lors de la phase de la génération et d’exporter les résultats selon différents aspects. Il se base sur les fonctionnalités suivantes :

* Lecture d’un schéma SQL (Structured Query Language).
* Génération aléatoire des données en fonction d’une certaine configuration que nous définissons dans notre application.
* Effectuer des tests à partir du schéma SQL de base de données.
* Définir des scénarios d’accès à la base de données avec différents types de requêtes, de simuler ces scénarios et d’évaluer l’impact d’une normalisation du schéma.

<===========

* ~~Lecture d’un schéma SQL (Structured Query Language) existant selon trois types de connexion (fichier SQL, fichier binaire (.db) et un processus d’un serveur SQL).~~
* ~~Génération aléatoire des données en fonction d’une certaine configuration que nous définissons dans notre application. L’outil permet à l’utilisateur de paramétrer la génération de données suivant ses besoins.~~
* ~~Effectuer des tests à partir d’un schéma SQL de base de données. L’outil permet de définir des scénarios d’accès à la base de données avec différents types de requêtes, de simuler ces scénarios et d’évaluer l’impact d’une normalisation du schéma.~~

===========>

## Etude de l’existant :

**Avant de commencer la réalisation de notre application, nous avons fait une étude sur les outils déjà existé qui proposent les mêmes fonctionnalités que notre application.**

**En première temps nous avons analysé les outils de génération de données, et en deuxième temps nous avons passé vers les outils de tests.**

### Outils de Générations

#### Generatedata.com

Est un outil web open source écrit en PHP, JavaScript et MySQL. Il vous permet de générer de gros volumes de données personnalisées (jusqu'à 5 000 enregistrements à la fois) sous divers formats, à utiliser dans les logiciels de test, les bases de données.

|  |  |
| --- | --- |
| Possibilités | Limites |
| * Génération de données aléatoires (Adresse, Villes…) selon quelques pays tel que la France, la Belgique, l’Allemagne…Etc. * Exportation des données générées sous différents formats : CSV, Excel, JSON, SQL et XML. * Possibilité de définir quelques propriétés pour la génération de données tel que : les intervalles et le format pour les données de type DATE, noms et prénoms de femmes, hommes ou les deux pour les données (Name)…Etc. * Expression régulière tel que Nom, Prénom, Email, Villes…Etc. * La génération de données (requêtes d’insertion) selon différents SGBD tel que : MySQL, PostgreSQL, SQLite, Oracle et SQL Server. * Génération de données uniques. * Utilisation de l’outil sans installation. | * Génération de données un peu lente (constat pour 100 tuples). * Génération de données pour une seule table (absence de relation entre les tables du schéma SQL). * Absence de TRIGGER. * Absence des VIEW. * Absence d’INDEX. |

Tableau 1 : Les possiblités et les limites de Generatedata.com

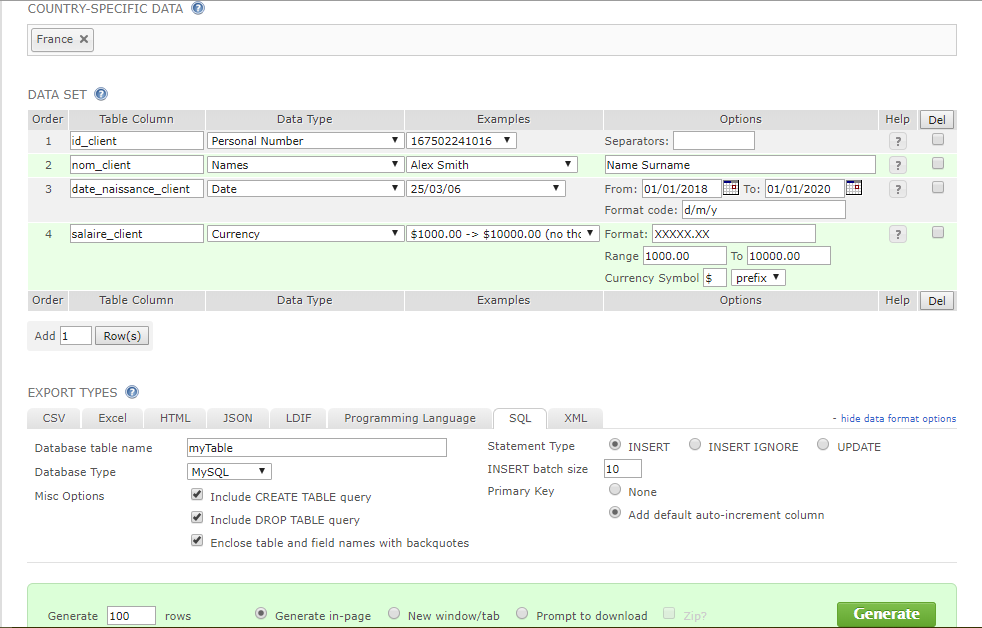


Figure 1 : Interface principale de Generatedata.com

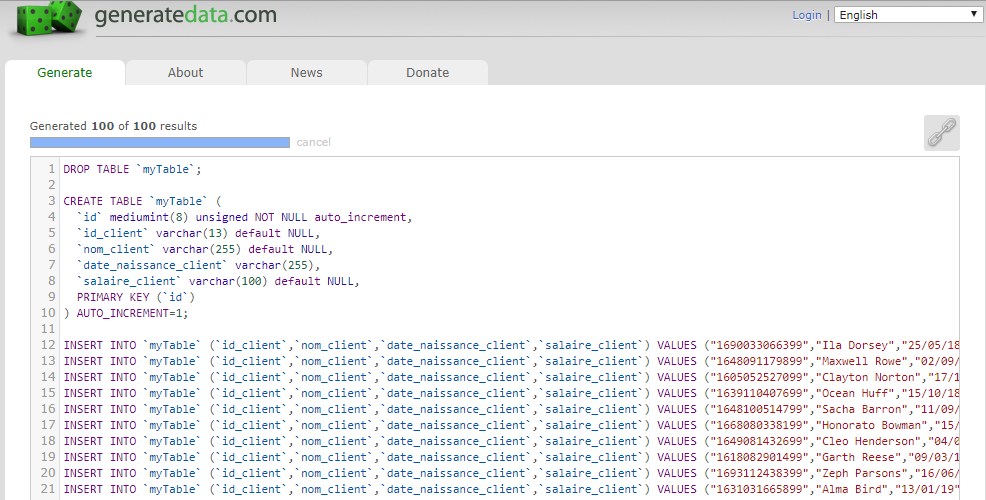


Figure 2 : Génération de du schéma SQL avec les requêtés d’insertions

#### Datanamic Data Generator MultiDB :

Est un outil qui permet de renseigner facilement des bases de données avec des milliers de tuples syntaxiquement correctes à des fins de. Cet outil lit votre base de données et affiche les tables et les colonnes avec leurs paramètres de génération de données. Seules quelques entrées simples sont nécessaires pour créer des données de test complètes. Vous pouvez l'utiliser pour produire des données de test à partir de zéro ou à partir de données existantes.

L'outil contient une vaste collection de générateurs intégrés spécifiques au pays ou à la langue. Il fournit un accès direct au serveur Microsoft SQL, Azure, MS Access, Oracle et MySQL.

|  |  |
| --- | --- |
| Possibilités | Limites |
| * Génération de données pour plusieurs plateformes tel que : Oracle, MySQL, SQL Server, MS Access, PostgreSQL et Azure SQL Database. * Option AUTO-INCREMENT pour clés primaires présente + choix de pas (incrémente de 1, 2, …). * Quelques propriétés pour les champs de la table tel que : le pourcentage de valeurs NULL, génération de valeurs UNIQUES, la répétition de valeurs. * Expression régulières tel que : le nom des rues, nom de produits, nom des banques…Etc. * Intervalle de génération pour les données INTEGER. * Génération de données pour plusieurs tables à la fois. * Génération de fichiers SQL contenant les requêtes d’insertion. * Générer des données de test à partir de zéro ou à partir de données existantes. * Prise en charge de l'intégrité référentielle. | * Outil non accessible au grand public (payant). * Génération de données lente (constat pour 2 tables, 5000 tuples chacune). * Absence de TRIGGER. * Absence des VIEW. * Absence d’INDEX. * Exportation de données qu’en fichier SQL, absence d’autres formats tel que : CSV, JSON…Etc. |

Tableau 2 : Les possiblités et les limites de Datanamic Data Generator MultiDB

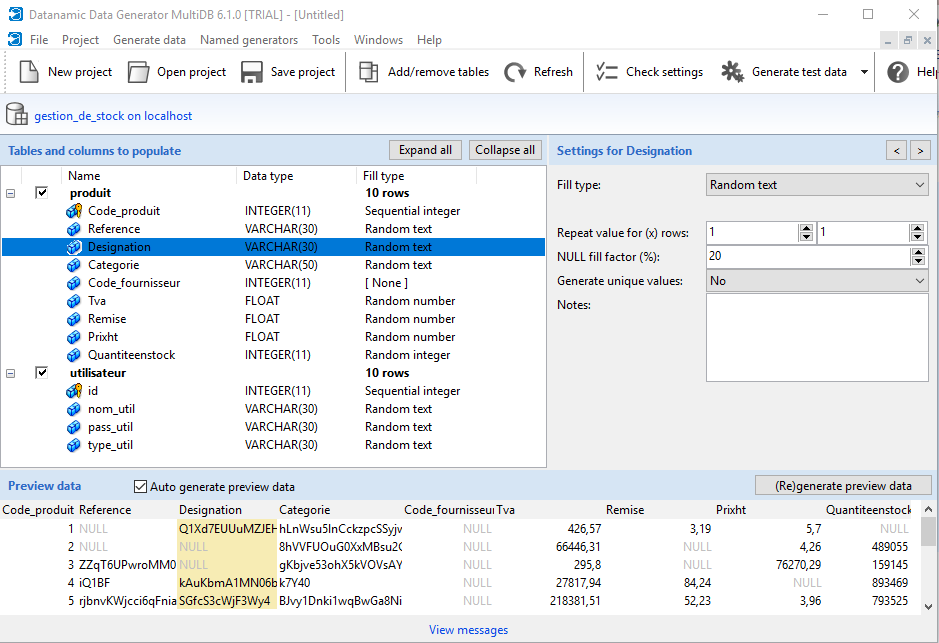


Figure 3 : Interface principale de Datanamic Data Generator MultiDB

#### Dbforge Data Generator for MySql :

Est un outil graphique permettant de générer rapidement des données de test significatives. Il contient plus de 200 générateurs intégrés avec une configuration appropriée qui permet d'émuler des données de test du monde réel.

Le programme prend en charge les clés étrangères pour la génération de données cohérentes dans plusieurs tables et la contrainte de vérification pour la génération de données cohérentes dans une ou plusieurs colonnes d'une table. Cela permet également de désactiver les déclencheurs et les contraintes afin d'éviter toute interférence avec la logique de la base de données.

|  |  |
| --- | --- |
| Possibilités | Limites |
| * Propriétés pour les colonnes : pourcentage de valeurs nulles, valeurs uniques, intervalles des données. * Génération de données selon le pays choisis (USA, Australie, Canada, Allemagne et Angleterre). * Possibilités de génération de données en mode séquentiel (Auto-incrément) + le choix du pas. * Génération du script SQL contenant les requêtes d’insertion. * Génération de données pour plusieurs tables à la fois. * La prise en charge des relations entre les tables (clés étrangères). | * Non accessible au grand public (payant). * La version gratuite permet de générer que 50 tuples par table. * Chaque version du produit opère sur un seul SGBD (une seule version ne regroupe pas tous les SGBD). * Absence d’autres formats tel que CSV, JSON, néanmoins leur présence dans la version payante. |

Tableau 3 : Les possibilités les limites de Dbforge Data Generator for MySql



Figure 4 : Fichier du script résultat de la génération de Dbforge Data Generator for MySql

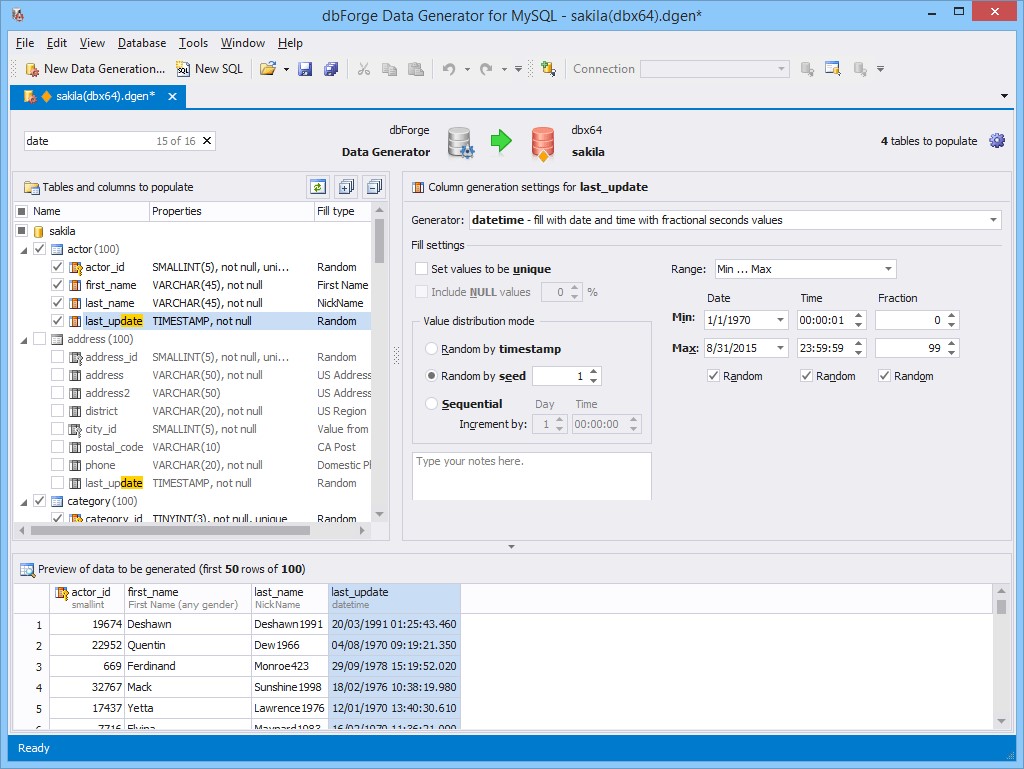


Figure 5 : Interface principale de Datanamic Data Generator MultiDB

#### Faker :

### **Est une bibliothèque open source crée par PHP, contribuer dans leur développement 367 personnes. Il 'est possible de l’intégrer dans un projet PHP afin de généré des donné riel pour la base de données avec moins de dix liens de code. Elle pemit au developeur de simulé des différant scénarios pour la base de données, [6]**

**La bibliothèque permit facilement de générer des clés primaires, des clés étranges et les différant format des données avec des valeur réale et avec des valeurs aléatoires.**

|  |  |
| --- | --- |
| **Possibilités et avantages** | **Limites** |
| * **Gratuit** * **Fournir un grand nombre des valeurs réale** * **Facile a utilisé et installé** * **Possibilité d’utilisation des expressions régulières** * **…. to be continue** | * **Utilisable seulement par les développeurs PHP** * **Pas d’interface graphique** * **Il n’est pas possible d’extraira les données générées par des défirent format comme XML ou JSON** * **C'est l’utilisateur qui donne différant type des donné pour chaque attribut de la base de données** * **C'est aux développeurs de spécifier l’ordre d’exécution des tables** |

**Outils de Teste des bases des données :**

**Apache Jmeter Tester :**

**Est une application open source développer en Java a pour objectif de faire des tests et mesuré la performance des applications, il permet de simuler des différentes actions vers un serveur et vérifier les résulta obtenir.**

**Elle est capable de tester plusieurs types de service comme les applications web, les protocole HTTP, HTTPS ou FTP et les bases des données via le SJBD.**

**Pour tester les bases de données Jmeter nous permet de simuler des utilisateurs et spécifier pour chaqu’un leurs requêtes et le nombre de fois a les exécuté, pour visualiser et analyser les résulta il est possible d’ajouter différant type des ‘Listeners’ , et afin qu’on puisse affirmé et vérifie qui les résultats obtenir sont bien celle attendue on ajoute à la configuration des ‘Assertions’ qui contrôle chaque réponse. [1]**

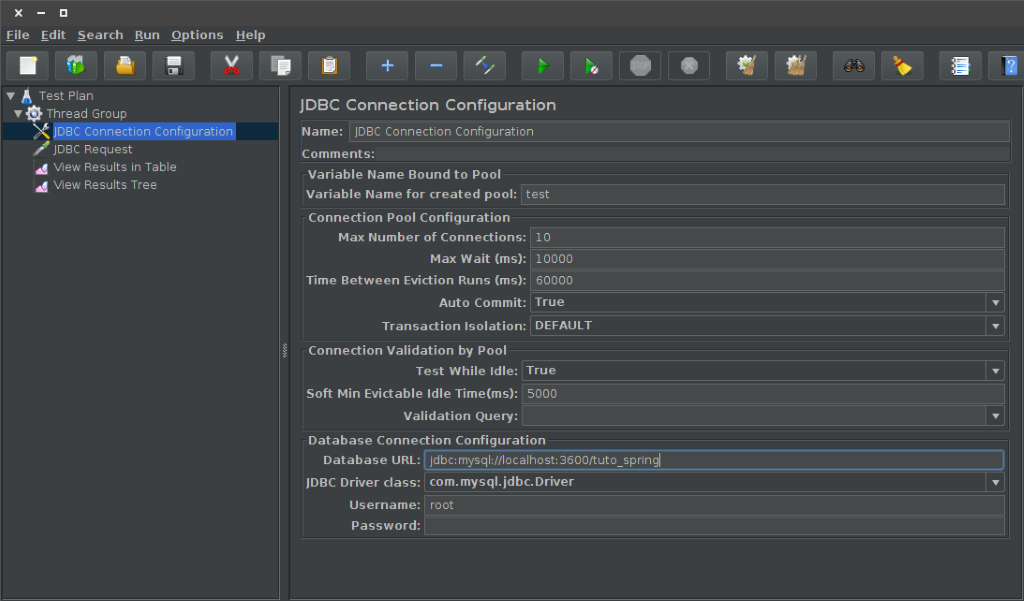


Figure 1 : configuration de Jmatre

**Faker**:

**C'est la bibliothèque qui on a déjà présenté dans la partie présidant**

**On revient à la bibliothèque Faker qui on a présenté dans la partie des outils de génération des données**

**Cette bibliothèque permit après la génération des donnés de les insérer directement dans la base de données concerné, et on cas d’erreur dans l’insertion nous a donne la description exacte pour l’exception. Cela permet de vérifier la préséance des tables, la préséance des columens, vérifier les types des données et vérifier la langueur des champs.**

**Et ce n’est pas seulement des propriétés spécifiques de cette bibliothèque mais la plus par des outils de génération des données permettons de fais ce genre de vérification.**

1.4 Tester les Base des données :

**En tant que développeur ou éditeur d’un programme informatique nous avons plusieurs défis, soit au moment de la réalisation ou après l’achèvement de projet, et l’un de ces défis les plus important de cycle de vie du programme est le teste qui nous assure le bon fonctionnement de l’application, augmenter la qualité et réduire au maximum le nombre des bugs.[2]**

**Est l’un des choses les plus important à tester dans un projet informatique est les bases des données. Ces teste doit comprend principalement la validation des schémas, des tables, des clés étrangères, des clés primaires, des index, des triggers et des columens,**

**Cela nous amener à créer de nombreuses requêtes et les exécuté sur la base de données soit pour des requêtes pour charge, insert, modifie ou supprimer le contenu de la base de données, et vérifier la réaction, le temps, le résultat obtenir.[3]**

**Les teste à effectuer durant le processus de teste sont :**

1. **Pour les tables et les columens :**
   * **Vérification de la présence de la table et des columens**
   * **Vérification le type des données**
   * **Vérification de la langueur des champs et c’est ils sont suffisont**
   * **Vérifie les champs obligatoires**
2. **Pour les index et les clés**
   * **Vérification la présence des index pour les clés primaires et les clés étrangères pour chaque table.**
   * **Contrôler si les références des clés étrangères sont disponibles**
   * **Contrôler que les type des clés étrangère et les type de leur référence sont les mêmes**
3. **Pour les opérations (SELECT, DELETE, UPDATE)**
   * **Vérifier le temps d’exécution pour une ou plusieurs requêtes**
   * **Vérifier qui les résulta obtenir de ces opérations sont bien les mêmes résultats attendus**
   * **Vérifier les l’ordre des résultats pour les opération Selecte**

[1]: <https://jmeter.apache.org/>

[2]: <https://phpunit.readthedocs.io/fr/latest/database.html>

[3]:<https://www.guru99.com/data-testing.html>

## Outils et environnement de développement :

La réalisation de notre application est basée sur les outils et langages suivants :

### Langage de programmation :

Java : La particularité de ce langage est que les logiciels sont très facilement [portables](https://fr.wikipedia.org/wiki/Portabilit%C3%A9_(informatique)) sur plusieurs [systèmes d’exploitation](https://fr.wikipedia.org/wiki/Syst%C3%A8me_d%27exploitation) [2]. Contrairement à la plupart des autres langages, Java met à la disposition du développeur une API très riche lui permettant de faire de très nombreuses choses que de coder par lui-même des fonctionnalités basiques. Il nous propose à peu près tout ce que nous avons besoin directement dans le JDK. Ceci est un énorme avantage, qui augmente encore une fois grandement la productivité de développement.

Java et sa JVM aussi permettent au programmeur d'avoir un très haut niveau d'abstraction par rapport à la machine et donc pas besoin de s'occuper de la compatibilité matérielle.

### Environnements de développement :

Nous avons développé notre application à l’aide d’IDE (Integrated development environment) suivant la structure MAVEN.

**NetBeans** considéré comme [3]:

* L’environnement de développement le plus puissant et le plus adapté pour Java
* Large communauté, utilisé pour le développement d’applications web, desktop et mobile
* Adaptation avec différents systèmes d’exploitation tel que Windows, Mac et Linux.

**Scene builder** : est un produit oracle permettant de créer des interfaces graphiques en s’appuyant sur le Drag and Drop.

**Apache Maven** : un outil de gestion de projet informatique permettant de gérer les dépendances et automatiser la construction du projet java.

Il nous offre les fonctionnalités suivantes :

* Compilation et déploiement des applications Java (JAR, WAR)
* Gestion des librairies requises par l'application
* Génération des documentations du projet
* Intégration dans différents IDE

Git : **Git** est un [logiciel de gestion de versions](https://fr.wikipedia.org/wiki/Logiciel_de_gestion_de_versions) [décentralisé](https://fr.wikipedia.org/wiki/Gestion_de_versions#Gestion_de_versions_d.C3.A9centralis.C3.A9e). C'est un [logiciel libre](https://fr.wikipedia.org/wiki/Logiciel_libre) créé par [Linus Torvalds](https://fr.wikipedia.org/wiki/Linus_Torvalds), auteur du [noyau Linux](https://fr.wikipedia.org/wiki/Noyau_Linux), et distribué selon les termes de la [licence publique générale GNU](https://fr.wikipedia.org/wiki/Licence_publique_g%C3%A9n%C3%A9rale_GNU) version 2. En 2016, il s’agit du [logiciel de gestion de versions](https://fr.wikipedia.org/wiki/Logiciel_de_gestion_de_versions)le plus populaire qui est utilisé par plus de douze millions de personnes[3](https://fr.wikipedia.org/wiki/Git#cite_note-3).[REF]

### Bibliothèque :

**JavaFX** : est une bibliothèque développée par oracle permettant de fournir des composants modernes afin de mettre en place des interfaces ergonomiques et agréables à utiliser [4].

**DataFaker** : est une bibliothèque riche qui offre plusieurs types de données réelles sur demande. De plus elle fournit des valeurs très spécifique par exemple

**La bibliothèque contient plusieurs Class qui nous fournissent ...**

tel que des adresses mail et des prénoms [5].

## Méthodologie de travail

Dans ce qui suit, nous menons une étude comparative des différentes méthodologies du travail utilisées dans la littérature. Cette étude nous aidera à décider quant au choix de la méthodologie à poursuivre pour atteindre nos objectifs fixés au départ.

### Etudes comparatives :

Dans le tableau suivant nous présentons les principales différences entre les deux types de méthodes : les méthodes classiques et la méthode agile.

|  |  |  |
| --- | --- | --- |
| **Thème** | **Approche traditionnelle** | **Approche agile** |
| Cycle de vie | En cascade ou en V, sans rétroaction possible, phase séquentielle. | Itératif et incrémental. |
| Planification | Prédictive, caractérisée par des plans plus ou moins détaillés sur la base d’un périmètre et d’exigences définies et stables au début du projet. | Adaptative avec plusieurs niveaux de planification (macro et micro planification) avec ajustements si nécessaires au fil de l’eau en fonction des changements survenus. |
| Documentation | Produite en quantité importante comme support de communication, de validation et de contractualisation. | Réduite au strict nécessaire au profit d’incréments fonctionnels opérationnels pour obtenir le feedback du client. |
| Equipe | Une équipe avec des ressources spécialisées, dirigées par un chef de projet. | Une équipe responsabilisée ou l’initiative et la communication sont privilégiées, soutenue par le chef de projet. |
| Qualité | Contrôle qualité à la fin du cycle de développement. Le client découvre le produit fini. | Un contrôle qualité précoce et permanent, au niveau du produit et du processus. Le client visualise les résultats tôt et fréquemment. |
| Changement | Résistance voire opposition au changement.  Processus lourds de gestion des changements acceptés. | Accueil favorable au changement inéluctable, intégré dans le processus. |
| Suivi de l’avancement | Mesure de la conformité aux plans initiaux.  Analyse des écarts. | Un seul indicateur d’avancement : le nombre de fonctionnalités implémentées et le travail restant à faire. |
| Gestion des risques | Processus distinct, rigoureux, de gestion des risques. | Gestion des risques intégrée dans le processus global, avec responsabilisation de chacun dans l’identification et la résolution des risques. |

# Présentation du travail déjà réalisé

une **table** est un ensemble de données organisées sous forme d'un tableau où les [colonnes](https://fr.wikipedia.org/wiki/Colonne_(tableau)) correspondent à des catégories d'information (une colonne peut stocker des numéros de téléphone, une autre des noms...) et les [lignes](https://fr.wikipedia.org/wiki/Ligne) à des [enregistrements](https://fr.wikipedia.org/wiki/Enregistrement_(informatique)), également appelés entrées[REF].

une **clé primaire** est la donnée qui permet d'identifier de manière unique un [enregistrement](https://fr.wikipedia.org/wiki/Enregistrement_(base_de_donn%C3%A9es)) dans une [table](https://fr.wikipedia.org/wiki/Table_(base_de_donn%C3%A9es)). Une clé primaire peut être composée d'une ou de plusieurs colonnes de la table[REF].

Une clé étrangère identifie une colonne ou un ensemble de colonnes d'une [table](https://fr.wikipedia.org/wiki/Table_(base_de_donn%C3%A9es)) comme référençant une colonne ou un ensemble de colonnes d'une autre table (la table référencée). Les colonnes de la table référencée doivent faire partie d'une contrainte de [clé primaire](https://fr.wikipedia.org/wiki/Cl%C3%A9_primaire) ou d'une [contrainte d'unicité](https://fr.wikipedia.org/w/index.php?title=Contrainte_d%27unicit%C3%A9&action=edit&redlink=1)[REF].

## Analyse des besoins technique :

Afin de pouvoir construire le schéma de la base de données, nous avons utilisé une API (Application programming interface) pour récupérer l’information de la base de données. Cette dernière est une tâche difficile à résoudre (problème de complication). Pour cela, l’API JDBC (JAVA DataBase Connectivity) est un besoin technique.

Nous avons utilisé L’API JavaFaker comme une dépendance de l’application afin de générer les différents types de données réels (email, bâtiment, rue, …).

Les Graphes

## Conception :

Nous présentons dans ce chapitre les différents points essentiels au développement de notre projet, la principale tâche sur laquelle se base notre application est l’algorithme de génération de données et l’analyse et teste de schéma de la base de données que nous expliquons ci-dessous.

Avant d’entamer le développement de l’algorithme, nous définissons d’abord les différentes classes utilisées dans notre code. Ces dernières sont regroupées dans des packages en fonction de leur classification.

**Le *package* *Connection :***

**La classe *SQLConnection* : permet d’établir une connexion à la base de données de trois manières différentes, pour enfin nous pouvons récupérer les métadonnées de la base de données et exécuter des requêtes.**

***Le package beans :***

**La classe *SQLSchema* : représente un schéma SQL c’est-à-dire un ensemble de tables, identifiée par un nom, la liste des tables qu’il contient et la liste des attributs qu’ils n'ont pas de références trier par le nombre de lignes à générer.**

**La classe *Table* : représente une table SQL identifiée par un nom, ensemble d’attributs (les colonnes), le nombres de lignes à générer,** ~~le pourcentage de valeurs nulle dans cette table, une clé primaire(PrimaryKey )~~

La classe Attribute : permet de représenter une colonne d’une table d’une base de données. Elle est spécialement composée d’un nom, d’un type de données, d’une liste des instances possibles, d’une liste des attributs qui lui fait référence, d’une liste des attributs qui les référence, la table de cet attribut et un générateur de données DataFaker que nous expliquons ci-dessous. Nous indiquons aussi si l’attribut est unique ou est une clé primaire et s’il accepte les valeurs nulles, s’il a déjà généré et s’il appartient à un schéma circulaire.

La classe ForeignKey : ??

~~La classe PrimaryKey : permet d’identifier de manière unique un enregistrement dans une table. Elle désigne une contrainte de clé primaire où elle regroupe l’ensemble des attributs.~~

**Le *package* *Pre\_condition :***

**La classe PreCondition : permet de vérifier la configuration donner par l’utilisateur pour éviter les erreurs au moment de la génération.**

**Le *package* *Faker* :**

La classe *DataFaker* : est une classe abstraite qui permet de générer une liste d’instances d’un attribut. Elle est composée d’un intervalle de génération du type spécifique (entier, adresse, âge, département...), et l’attribut qu’il va lui générer et une expression régulière des types indéfinis.

Les classes *IntegerDataFaker*, *DateDataFaker*, *TextDataFaker,* BlobDataFaker, BooleanDataFaker et RealDataFaker héritent de la classe *DataFaker* afin d’implémenter la méthode abstraite generateValue() qui génère les données.

**Le *package* Util *:***

La classe FileUtile : permet de sérialiser et enregistrer le schéma dans un fichier, et d’importer et désérialiser le fichier déjà enregistrer pour sauvegarder l’état de l’application.

L’association réflexive *references* de la classe Attribut représente la contrainte de clé primaire.

* Schéma SQL dans un fichier texte.
* Exportation d’une base de données sous forme d’un fichier binaire (.db).
* Connexion à une base de données dans un serveur.

## Diagramme de classes :

Nous avons pris les principales classes qui permettent de modéliser l’application figurées ci-dessous.



Figure 6 : Diagramme de classes

## Algorithme de génération

**Pour générer les données d’un schéma SQL, nous avons créé un algorithme qui génère la plupart des cas que nous avons croisé pendant notre travail. Il respecte bien les formats des schémas de la base de données. Dans ces derniers, nous trouvons la forme SQL à suivre pour la génération des données ainsi que les contraintes de clés primaires et clés étrangères entre les tables, les contraintes d’unicité. Notre algorithme supporte les relations cycliques entre les tables.**

Pour pouvoir générer les données, notre algorithme suit les étapes suivantes :

### Construction du schéma SQL :

Nous avons utilisé l’API JDB capable de se connecter à des serveurs de [bases de données](https://www.commentcamarche.net/contents/104-bases-de-donnees-introduction) ([SGBD](https://www.commentcamarche.net/contents/104-bases-de-donnees-introduction)) pour pouvoir récupérer le format du fichier à générer et plus précisément les objets possédant les métadonnées.

D’abord, nous commençons à récupérer les différents noms des tables.  
Etant donné qu'il existe différents types de bases de données, et que chacune d'entre-elles possède sa propre façon de traiter les requêtes SQL et de renvoyer les résultats, L’API JDBC permet d'obtenir des résultats identiques quel que soit le type de base de données.

Ensuite, pour chaque nom *name* dans la liste des noms des tables, nous créons une nouvelle instance de chaque classe *Table.*

Puis, pour chaque table T dans le schéma, nous récupérons tous les attributs qui appartiennent à cette dernière avec les caractéristiques nécessaires (NOT NULL, NULLABLE, TYPE, …) et nous créons une nouvelle instance de chaque classe *Attribute.*

Une fois que les instances des tables sont créées et pour chaque table *T* dans le schéma, nous récupérons en premier les noms des attributs qui font partie de la clé primaire et nous créons une instance de la classe PrimaryKey en y ajoutant ces attributs.

Puisque toutes les références des tables existent dans l’objet *SQLSchema*, nous procédons après à la spécification des clés étrangères.

Enfin, pour chaque table *T*, et pour chaque attribut qui possède une référence, nous récupérons l’attribut référencé.

### L’entrée de l’algorithme de génération :

Notre application demande à l’utilisateur les entrées suivantes :

* Schéma SQL : contenant la structure de la base de données.
* Le nombre de lignes à générer pour chaque table dépend de chaque utilisateur, mais tant que le nombre est grand, la génération de vient de plus en plus lente.
* Une configuration par défaut pour les différents types de génération : l’algorithme prend en considération les entiers, les booléens, les flottants, les réels, les numériques et les blobs qui sert à stocker les fichiers dans SQLite comme les images par exemple. Cette configuration est définie par des intervalles pour les entiers, une longueur minimale et maximale pour les chaines, ainsi que les *dateMin* et *dateMax* pour les dates.

Cette configuration peut être faite pour n’importe quel attribut séparément.

### Préconditions de l’algorithme de génération de données :

Avant de commencer la génération de données, nous vérifions si les contrainte suivant ont respecté : ( ~~nous devons respecter )~~

* Schéma SQL valide (qui compile par le SGBD).
* Chaque attribut A1 unique ou primaire d’une table T1 qui référence un autre attribut A2 de la table T2 implique que le nombre de lignes à générer pour la table T2 soit inférieur à celui de la table T1.
* L’intervalle de génération d’un attribut (nombre de cas possibles) doit être supérieur au nombre de lignes demandé. Par exemple, si l’utilisateur demande n valeurs alors l’intervalle [min, max] correspondant à cet attribut doit contenir plus de n valeurs.
* Circulaire

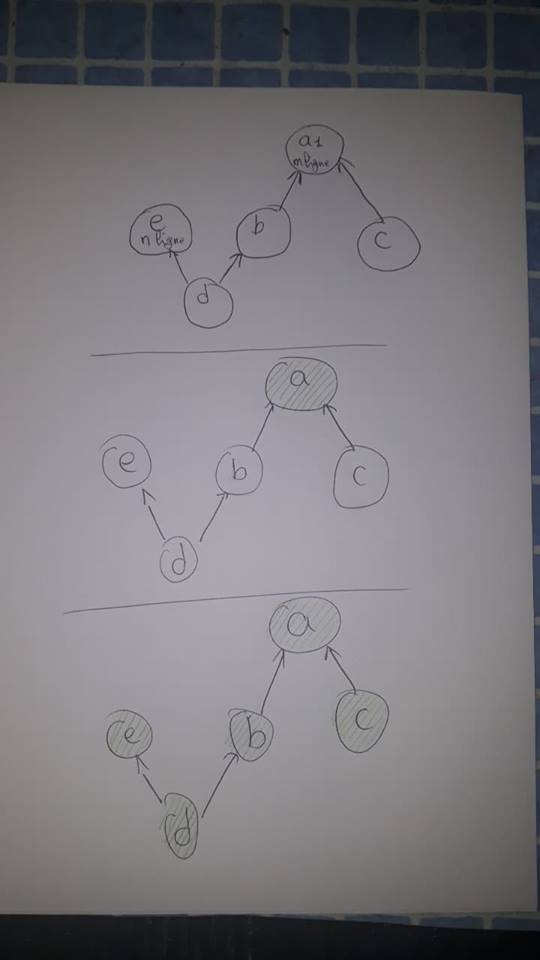
### **Algorithme de génération de données :**

Avant de commencer à discuter la manière que notre application génère les données, nous avons pris en considération les clés primaires et la contrainte d’unicité multicolonne comme des contrainte scélérate pour chaque attribut présente dedans.

**Nous présentons maintenant les différents cas que nous pouvons obtenir d’après le schéma SQL passé en entrer.**

#### **Cas non circulaire :**

**Prenons ce schéma comme exemple :**



* **Les attributs b et c référence a.**
* **L’attribut d référence b et e.**

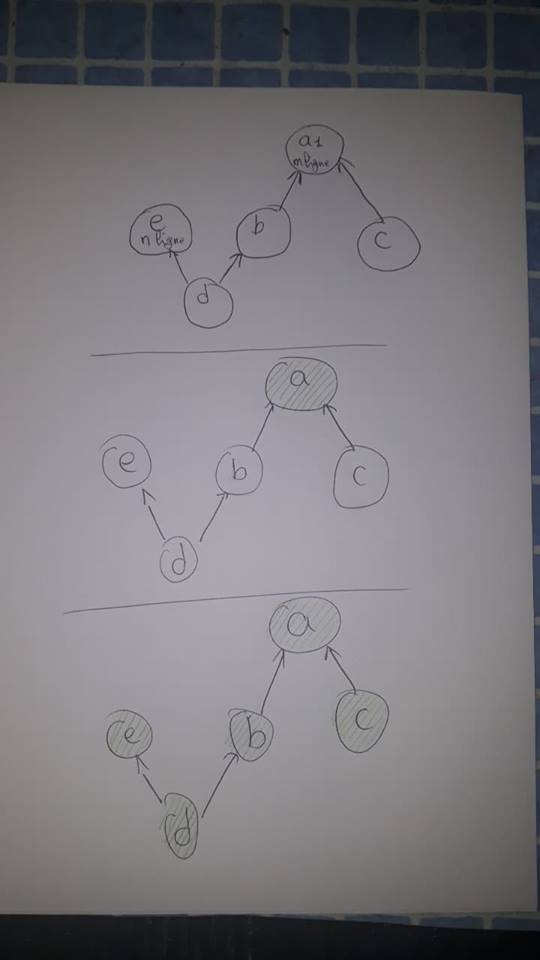
**Pour chaque attribut dans la liste des attribut route dans la classe SQLSchema :**

**Supposons m>n, donc notre liste route est :**

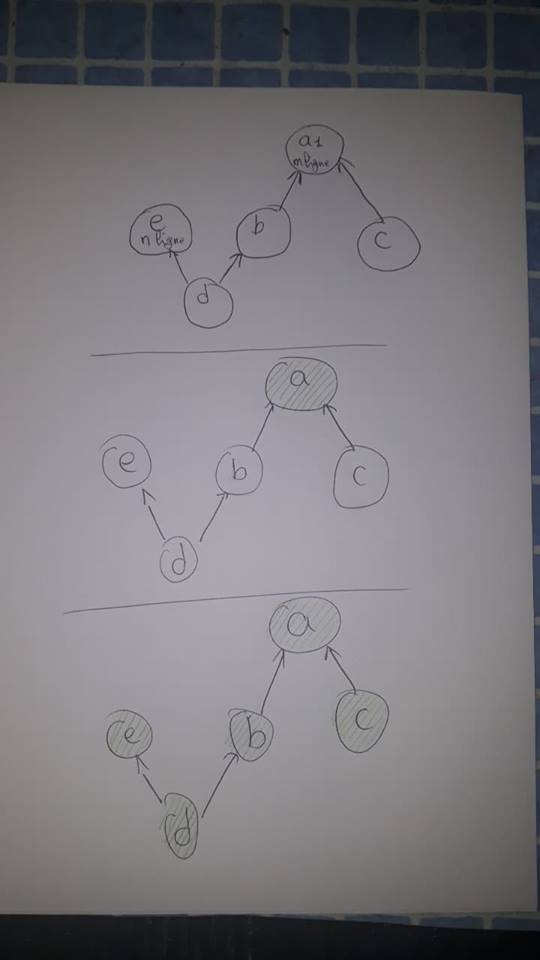


**1 vérification si l’attribut déjà généré sinon :**

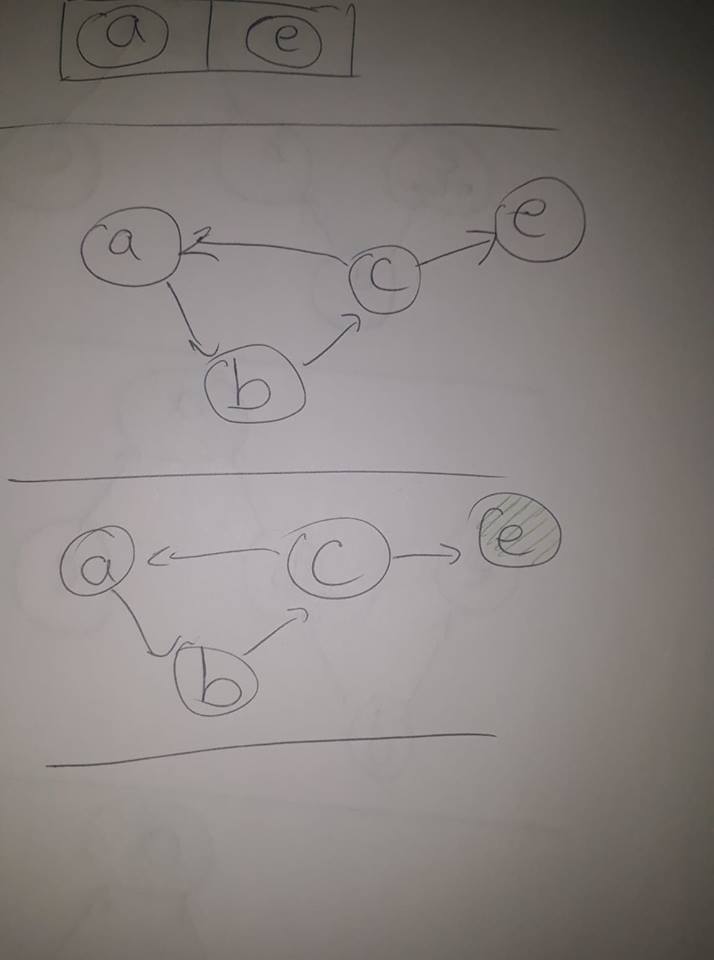
**1.1 Génération la liste des instances de cet attribut**



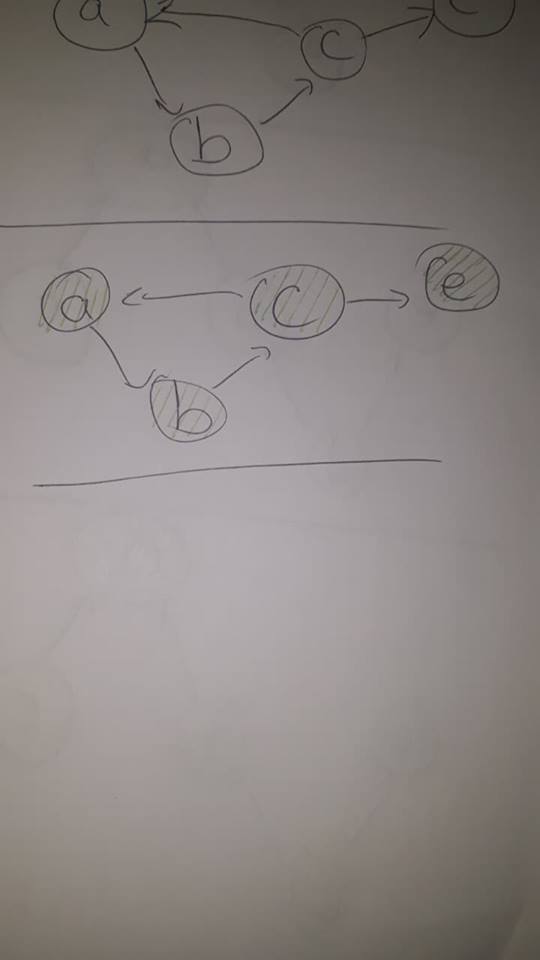
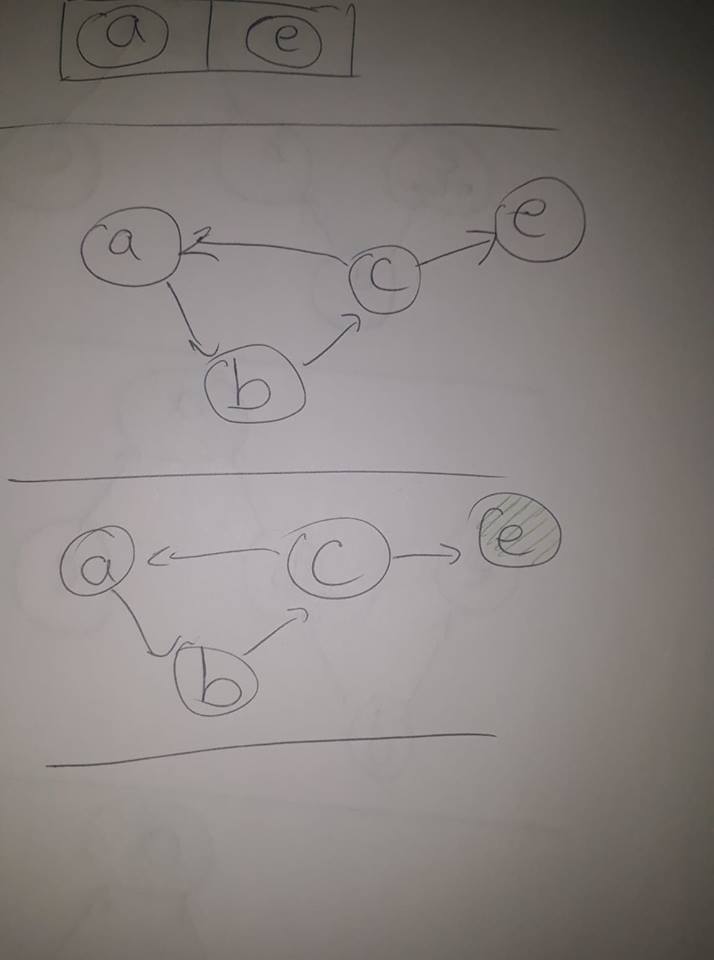
**1.2 Coller la liste d’instance pour chaque attribut des listes referenceMe et reference d’une façon récursive en vérifiant si l’attribut a déjà généré**



**Cet algorithme génère même les schémas circulaires qui ont une ou plusieurs relations avec un attribut root**



* **L’attribut a référence b.**
* **L’attribut b référence c.**
* **L'attribut c référence a et e.**



**2 Pour chaque attribut dans T possède des références alors nous fixons la taille de liste d’instance (on prend une sous ensemble ou bien on duplique la liste à fin d’avoir le nombre demandé)**

#### **Cas circulaire :**

Dans ce cas, la liste d’attribut route est vide. Pour chaque Table T dans le schéma SQL.

1 Pour chaque attribut dans T dont  ne possède aucune référence alors nous générons la liste des instances.

1.1 referenceMe (: Pour chaque attribut qui référence , alors nous donnons la liste des instances de à .

1.2 Pour chaque attribut qui référence, referenceMe( référence {{ceci est faite récursivement}}

2 Pour chaque attribut dans T possède des références alors nous fixons la taille de liste d’instance (on prend une sous ensemble ou bien on duplique la liste à fin d’avoir le nombre demandé)

Pour chaque Table T dans le schéma SQL.

3 Pour chaque attribut dans T où avec une liste des instances vides (nous somme dans une relation circulaire)

Nous générons la liste des instances et ferons les mêmes étapes que nous avons mention ci-dessus.

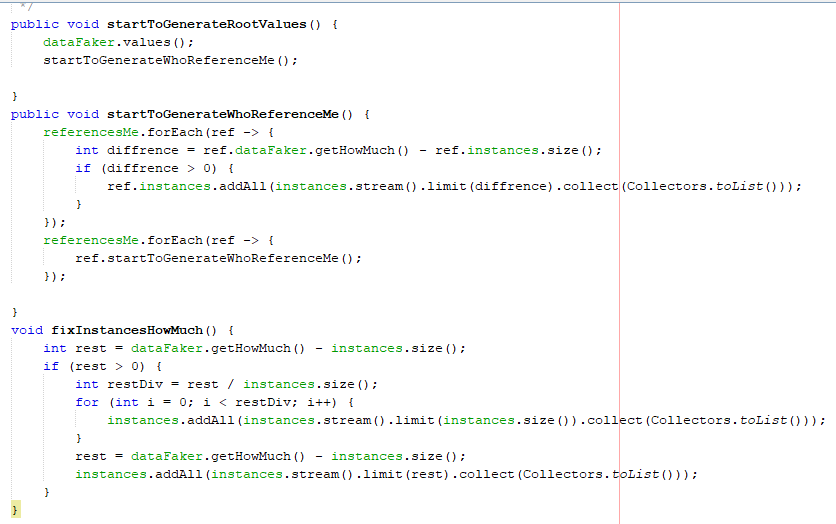


Figure 7 : Les principales méthodes pour l’algorithme de génération

Nous proposons un graphe qui permet de schématiser un schéma SQL de trois tables A, B et C



Figure 8 : Schéma SQL sous forme d'un graphe avant la génération

L’attribut c1 référence b1, qui lui-même référence a1

L’attribut a2 référence b2, qui lui-même référence c2, qui lui-même référence a2 (cas d’une relation circulaire)



Figure 9 : Schéma SQL sous forme d'un graphe après la génération

Les attributs de couleur grise désignent le point 1 dans notre algorithme.

Les attributs de couleur violette désignent le point 1.1 dans notre algorithme.

Les attributs de couleur bleue désignent le point 3 dans notre algorithme.

### Génération des instances :

Pour un attribut A donné, nous récupérons la liste vide des instances de A.

Si A est un attribut primaire (appartient à une clé primaire) ou possède la contrainte d’unicité. Pour i de 1 jusqu’au nombre de valeurs demandées :

* Pour les entiers et pour les réels, nous calculons le pas ([a,b], nombre) = (b-a/nombre) et nous ajoutons la liste des instances une valeur V = V+ aléatoire(1,pas), ou V commence de a.
* Sinon, nous générons une valeur V avec la configuration donnée (type, intervalle, ...) tant que V n’appartient pas à la liste des instances. Et à la fin nous ajoutons V dans la liste.

Sinon, pour i de 1 jusqu’au nombre de valeurs demandées :

Nous générons une valeur V avec la configuration donnée (type, intervalle, ...) et nous ajoutons V à la liste

### Estimation de temps pour la génération de données

Nous avons pour l’instant testé de représenter le temps nécessaire pour la génération d’un nombre de tuples souhaité par l’utilisateur pour une seule table possédant une clé primaire.

Nous avons constaté que la génération d’informations uniques (clé primaire) a un impact considérable sur le temps de génération.

Nous notons aussi, le nombre de tables ou de tuples n’a pas de lourdes conséquences sur le temps de génération, c’est-à-dire que ce dernier a une relation directe avec les propriétés des tables (clé primaire ou unique).

## Complexité

Après avoir présenté notre algorithme de génération, il est clair que la façon de générer des instances pour les attributs est d’une complexité non déterministe car à chaque fois que nous générons une valeur, nous testons si elle existe. Cette manière de génération est juste pour les types complexes comme les chaines.

Cependant, la génération de données pour les nombres, c’est-à-dire les entiers et les réels, est d’une complexité linéaire O(n).

## Limites de l’algorithme :

Notre algorithme ne prend pas en charge les points suivants :

* Les différentes bases de données Mysql, Oracle, ......
* La contrainte CHECK de validation.

### Exemples :

**3\_tables\_inheritance.sql** : est un exemple qui permet de générer des données pour trois tables. Chacune des tables est en relation avec celle qui la suit. C’est une relation d’héritage. Nous trouvons aussi les contraintes de clé primaire et de clé étrangère.

**3\_tables\_with\_all\_type.sql** : est un fichier contenant tous les types de variables que nous pouvons générer avec notre algorithme et qui sont respectés par Sqlite.

**Circuler.sql** : est un exemple montrant les relations circulaires entre les tables.

**errorSchema.sql** : est un exemple non valide car il contient des erreurs syntaxiques mais qui est accepté par notre algorithme.

**oneToThree.sql** : est un autre cas accepté par notre algorithme, il s’agit d’un seul attribut qui fait référence à trois attributs chacun d’une table différente à la fois.

**tables\_not\_exist.sql**: il s’agit d’un exemple où notre algorithme permet de détecter si un attribut fait référence à une table qui n'existe pas.

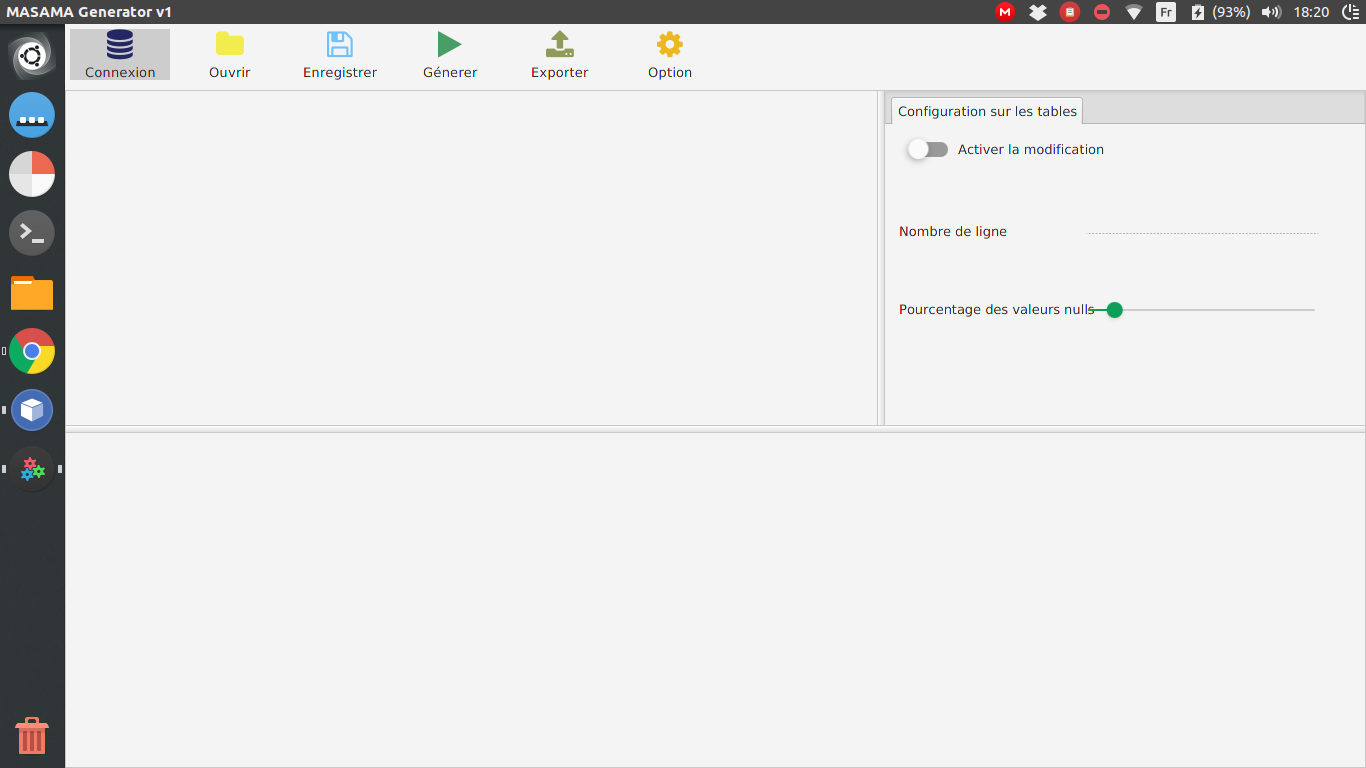
**table\_withe\_data.sql** : notre application prend en considération les données existantes déjà dans la BD mais ne respecte pas les contraintes de clé primaire et les contraintes d’unicité dans ce cas.

**OneToTowTablesToOne(bug).sql** : dans cet exemple, un seul attribut fait référence à deux attributs différents et l'un de ces derniers lui-même fait référence à un autre attribut d’une quatrième table.

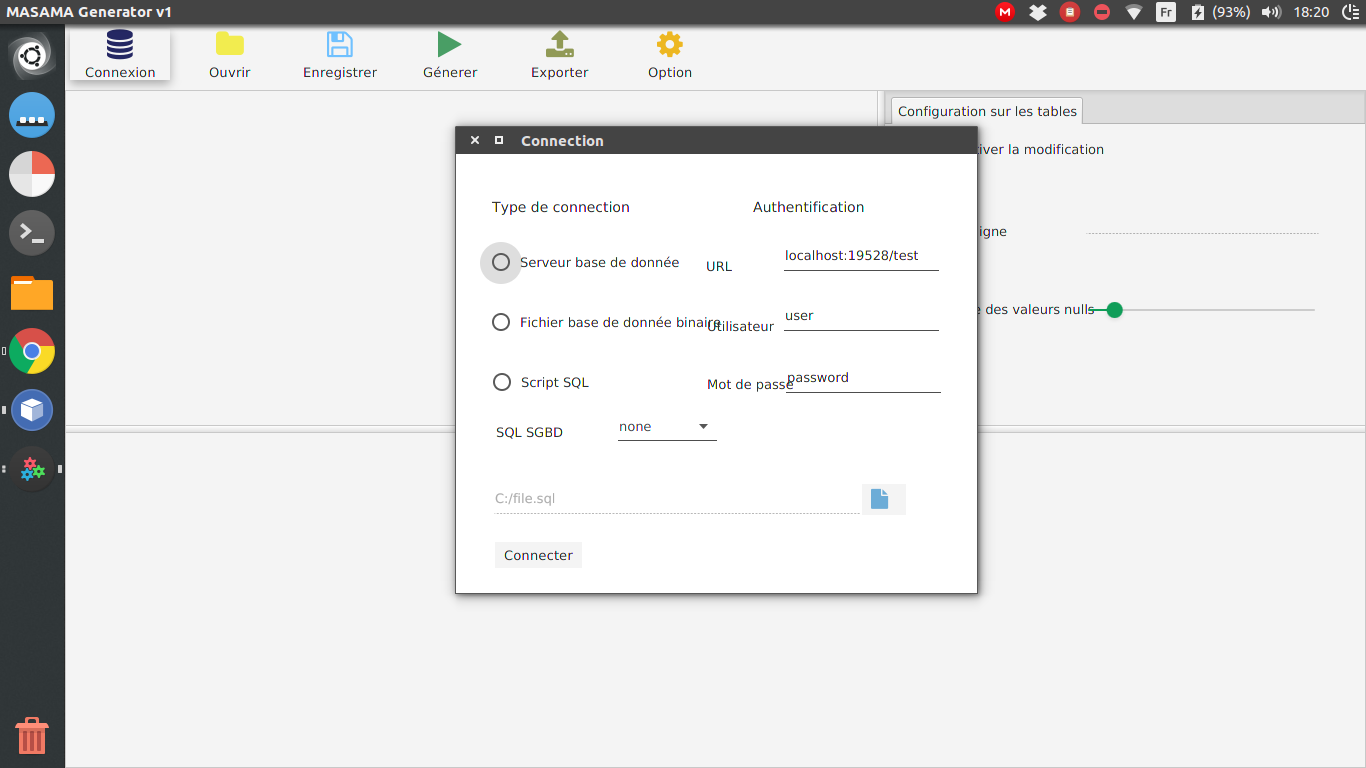
## L’interface graphique

L’interface graphique de l’application est divisée en quatre parties :

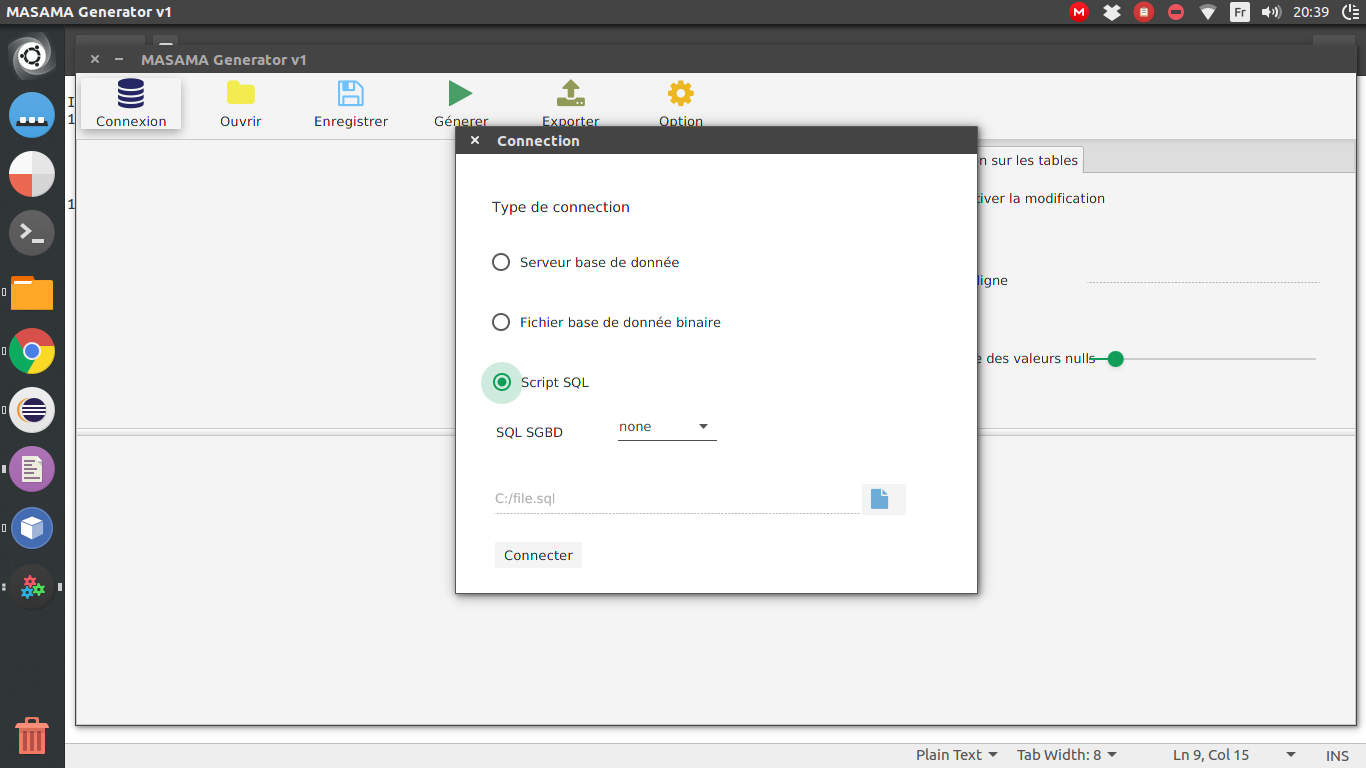
* Header où nous trouvons les différentes fonctionnalités offertes par l’application.
* Body : Contient deux parties :
  + Partie gauche : dans cette partie on trouve tous les tableaux que nous avons lus depuis le fichier importé avec leurs configurations pour les attributs.
  + Partie droite : on trouve les configurations des tableaux lus et les lignes des tuples à générer.
* Footer : pour voir le tableau body et ses données générées.



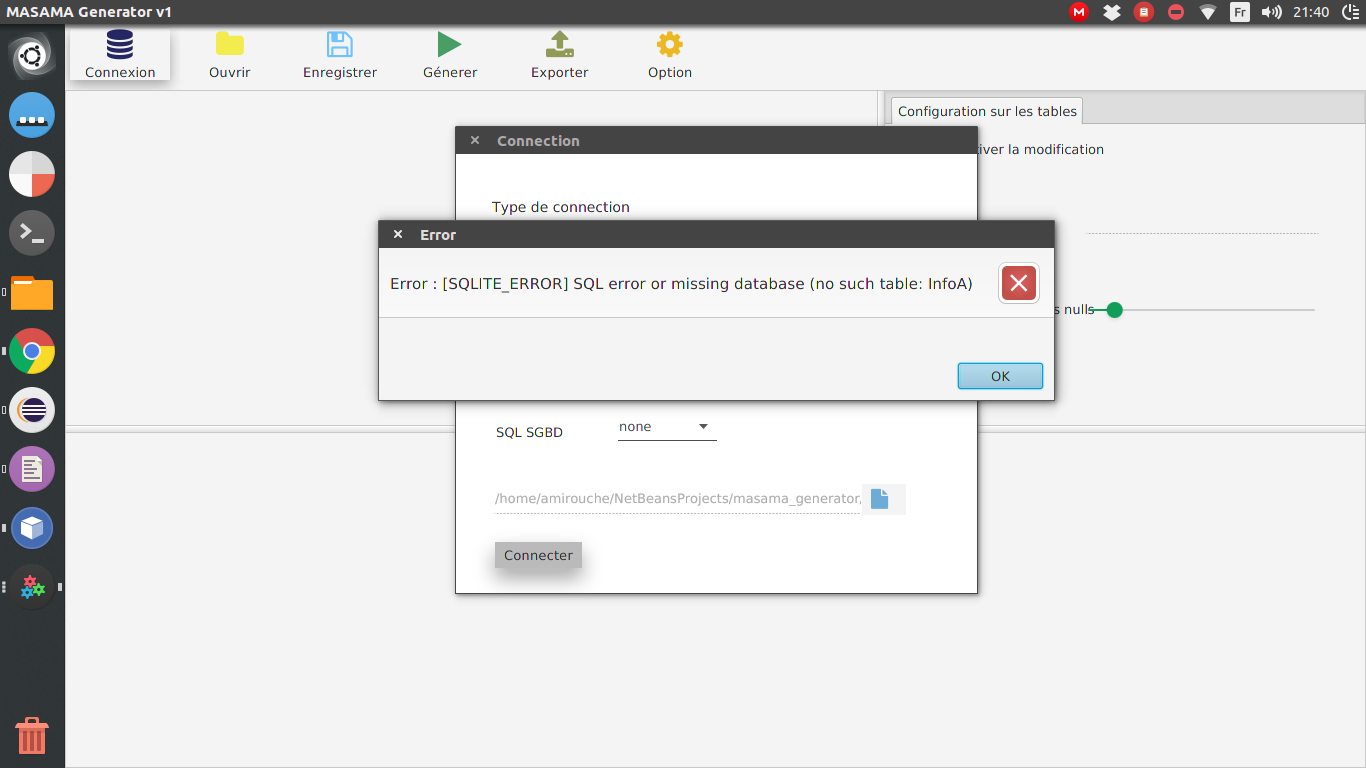
* Connexion : nous avons fait trois types de connexion
  + Un serveur d’une base de données, et dans ce cas nous vous demandons de remplir les cases d’Url pour le lien de serveur, nom d’utilisateur et le mot de passe du serveur.



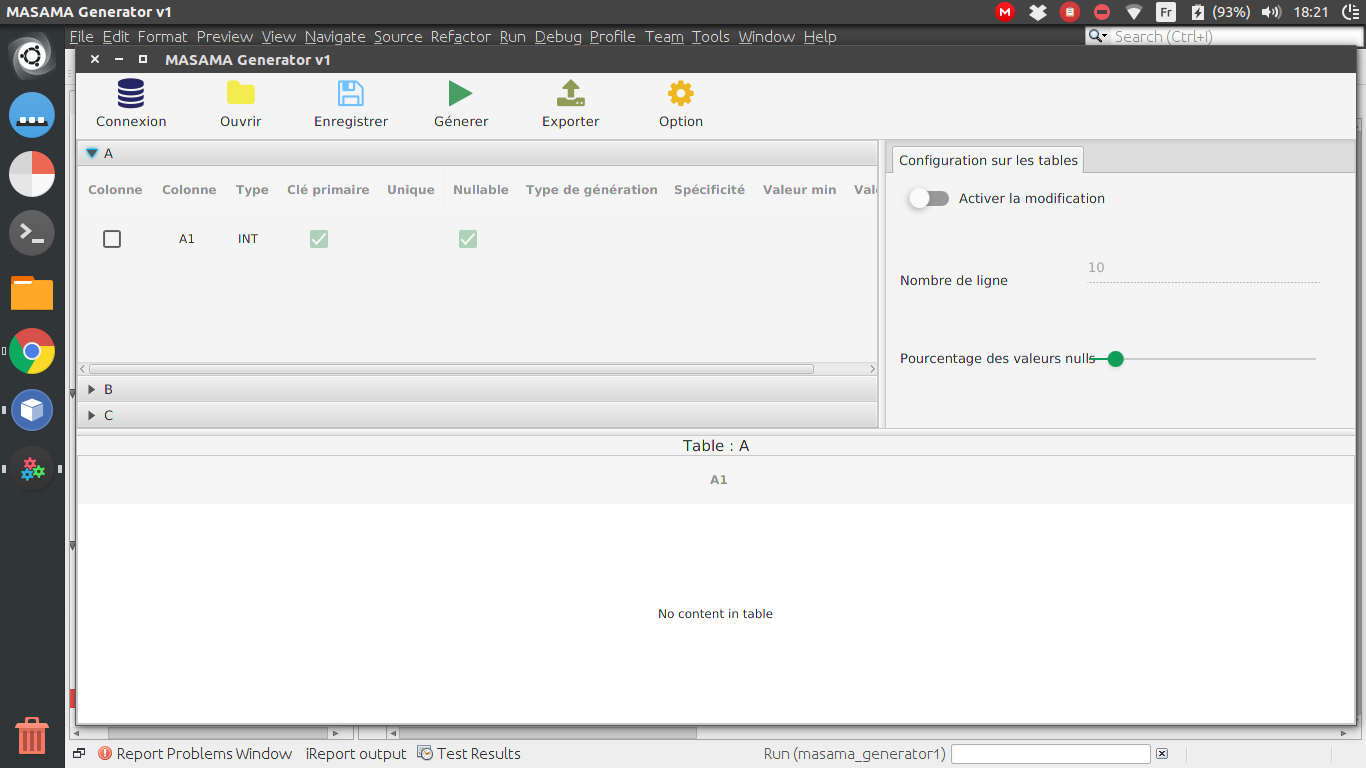
* + Un fichier binaire ou un script sql en spécifiant le SGBD, pour importer un fichier (il y a un dossier où vous trouverez des fichiers .sql pour tester l’application /masama\_generator/SQL)



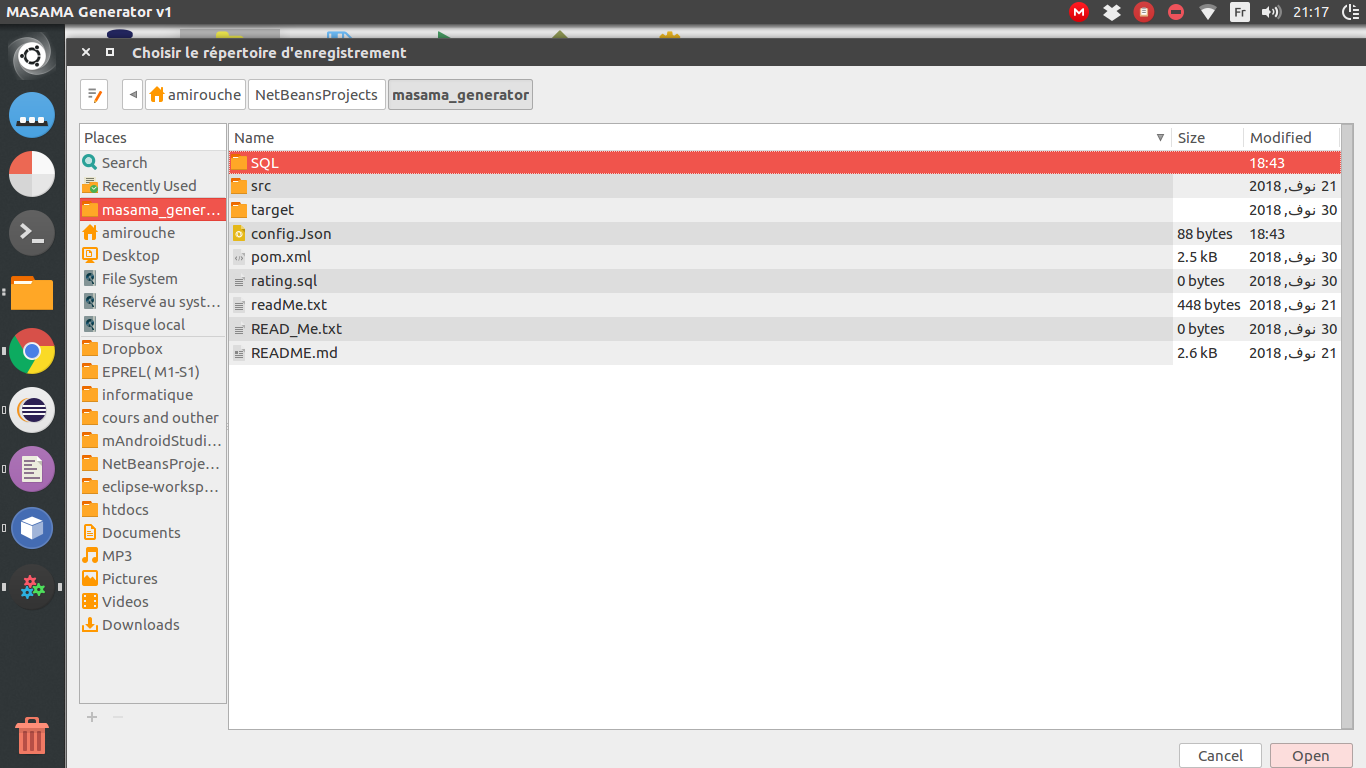
En cliquant dans le bouton Connecter, notre application va lire le script. S’il y a une erreur dans ce dernier d’après les préconditions que nous avons implémentées, un message d’erreur s’affiche.



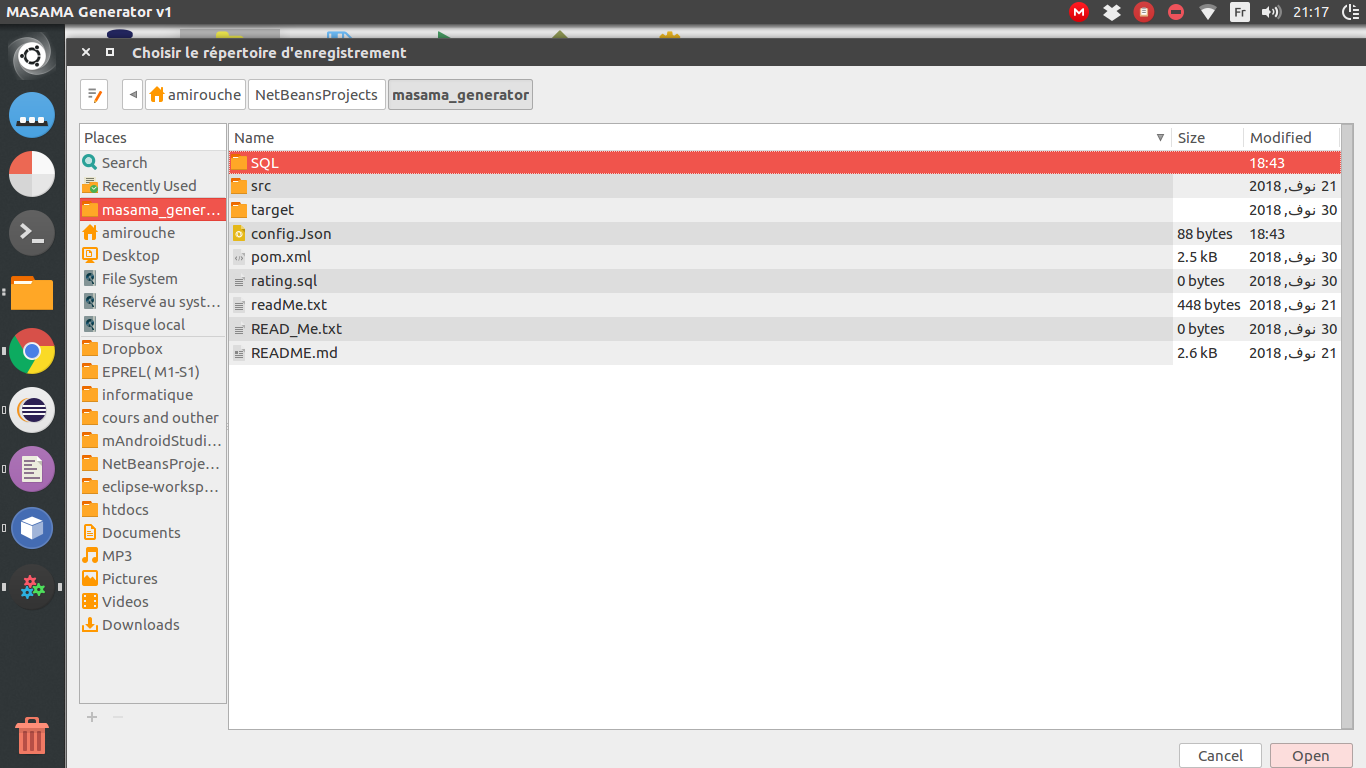
Sinon, après une connexion réussie, un nouvel affichage apparait permettant la configuration des différentes tables et ses attributs.



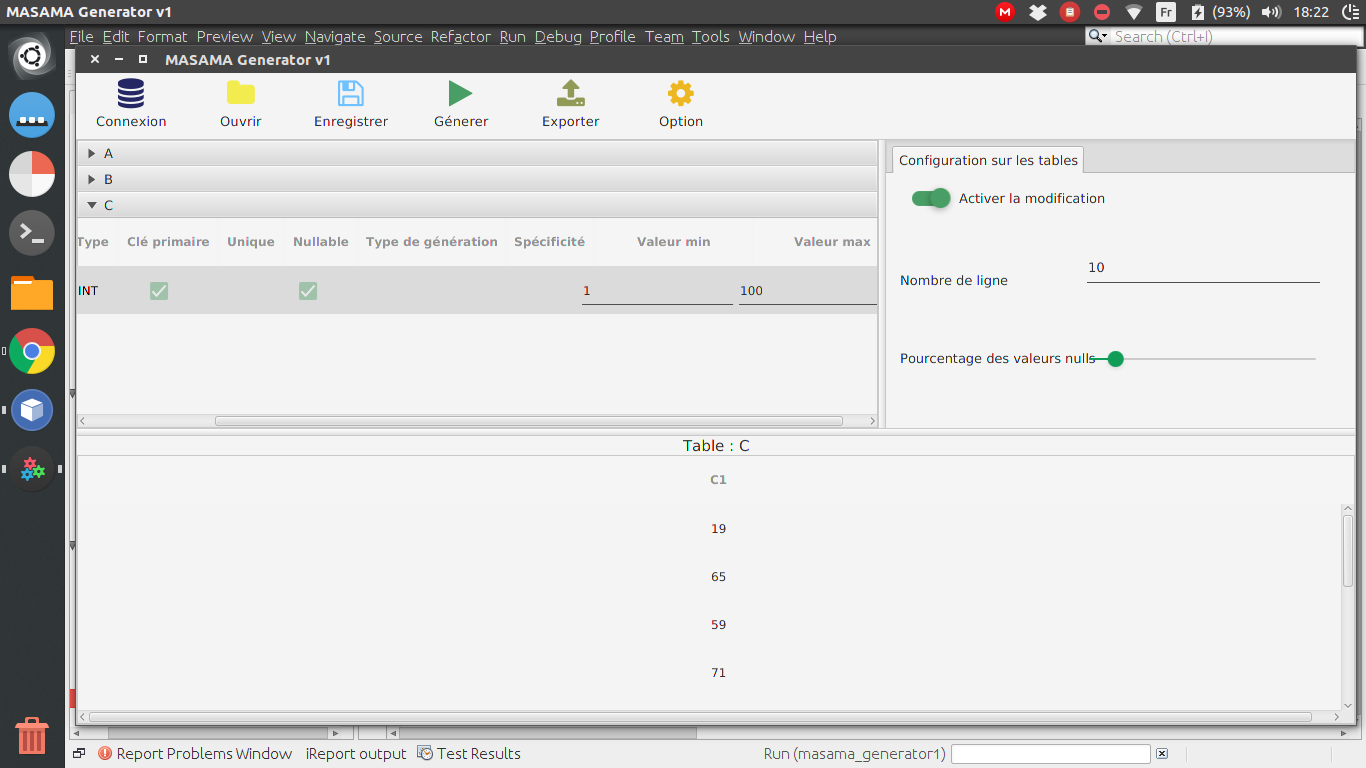
* Enregistrer : pour enregistrer le dernier état de l’application avec toutes les données déjà saisies, le fichier enregistré à l’extension. masama



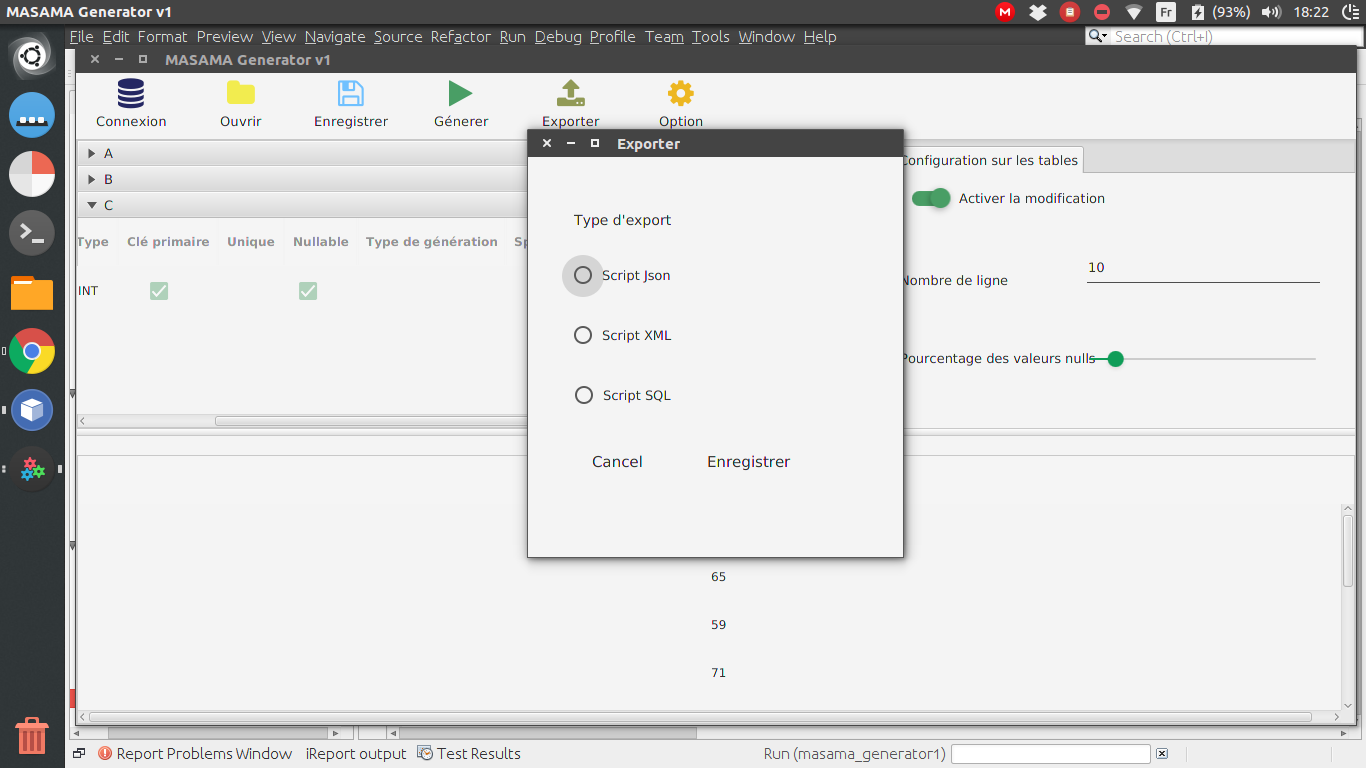
* Ouvrir : pour ouvrir le dernier état enregistré(le bouton “Enregistrer”), le fichier sélectionné doit avoir l’extension .masama



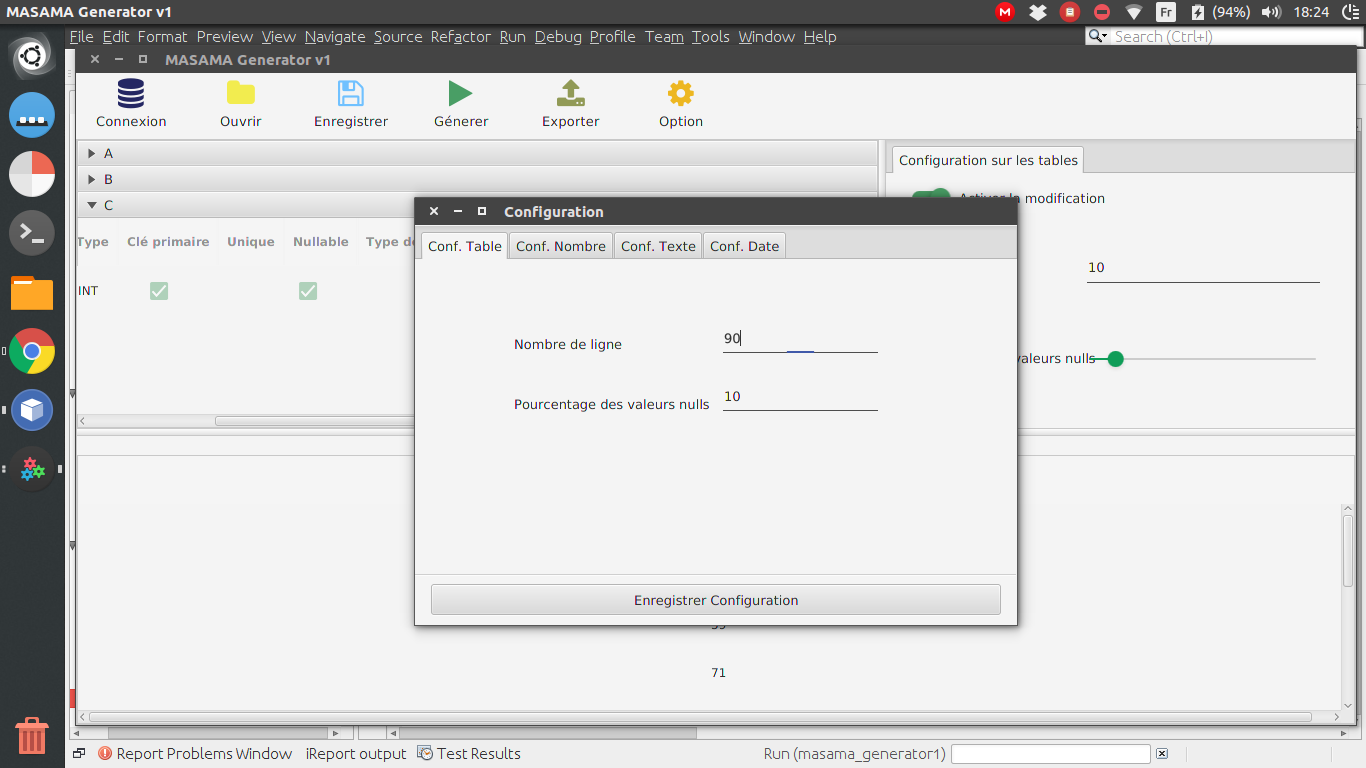
* Générer : permet la génération des données et les afficher ensuite dans des tableaux.



* Exporter : permet d’exporter les fichiers sous plusieurs formats : sql, xml et json.



* Option : pour modifier les configurations des données générées. Nous trouvons :
* Le nombre de lignes à générer pour tous les tableaux.
* Le pourcentage des valeurs nulles.
* Le nombre minimal et maximal pour les attributs réels.
* La longueur minimale et maximale pour les chaines de caractère.
* La date minimale et maximale pour les dates.



# Organisation du travail

## Pilotage du projet avec Scrum :

La méthode Scrum ou plus exactement le cadre méthodologique est de loin la méthode Agile la plus utilisée dans le monde. Elle bénéficie aujourd’hui de nombreux retours d’expérience [6].

Le principe de base de Scrum est de focaliser l’équipe de façon itérative sur un ensemble de fonctionnalités à réaliser [5]. Il s'appuie sur le découpage d'un projet en boîtes de temps, appelés « sprints ». Chaque sprint possède un but à atteindre, défini par le directeur de produit, à partir duquel sont choisies les fonctionnalités à implémenter.

Dans cette partie nous définissons le *backlog* du produit, les différents rôles attribués aux membres de l’équipe, le plan du sprint et le *timebox* du projet.

### Backlog de produit :

Le *backlog* de produit est la liste des fonctionnalités attendues d'un produit. Plus exactement, au-delà de cet aspect fonctionnel, il contient tous les éléments qui vont nécessiter du travail pour l'équipe. Les éléments y sont classés par priorité ce qui permet de définir l'ordre de réalisation [2].

Le tableau suivant présente le *backlog* de produit de notre application.

|  |  |  |
| --- | --- | --- |
| **En tant que** | **On souhaite** | **Priorité** |
| Utilisateur | Effectuer la lecture du schéma d’une base de données. | 1 |
| Utilisateur | Générer des données aléatoires pour les tables. | 1 |
| Utilisateur | Sauvegarder les données générées et la configuration des paramètres utilisés. | 1 |
| Utilisateur | Tester/Scenario | 2 |
| Utilisateur | Amélioration et optimisation | 3 |

Tableau 4 : Backlog de produit de notre application

## Plan des sprints :

Après avoir découpé notre projet final en tâches, il nous a fallu regrouper ces tâches en sprints. Un sprint, bien qu’étant une partie incomplète du projet doit être un livrable exécutable et réutilisable. Il faut donc délimiter les sprints de manière à respecter cette règle.

Après le regroupement, nous avons obtenu 3 sprints à réaliser :

* **Sprint 1 :** Génération des données
* Effectuer la lecture du schéma d’une base de données existante.
  + Générer des données aléatoires pour les tables.
  + Sauvegarder les données générées et la configuration des paramètres utilisés.
* **Sprint 2 :** Tester/scenario
* **Sprint 3 :** Amélioration et optimisation

### Déroulement du premier sprint « Génération des données » :

#### Backlog du premier sprint :

Le tableau suivant présente les tâches à effectuer dans le premier sprint et qui vont durer 30 jours du 28/10/2018 au 05/12/2018. La revue de ce sprint est considérée le 30/11/2018.

|  |  |  |
| --- | --- | --- |
| **Fonctionnalité** | **Les taches** | **Durée (Jour)** |
| Lecture du schéma de la base de données. | - Lecture d'un schéma SQL et création de l'interface adéquate.  - Création de la connexion aux différents BD (Mysql, SQL Server) | 5 |
| Générer les données aléatoires. | - Création des beans.  - Algorithme de génération des données  - Création de l'interface adéquate.  - Exportation des données (sql, xml, json).  - Progression avec les threads pour la génération des données asynchrones.  - Vérification des préconditions de la génération.  - Amélioration du design de l'interface avec JavaFX.  - Optimisation du code.  - Documentation de toutes les méthodes déjà existantes. | 30 |
| Sauvegarder les paramètres et les données générées. | - Enregistrer et charger l'état actuel de projet (fichier binaire).  - Enregistrer et charger la configuration de la génération. | 5 |

Tableau 5 : Tâches à effectuer dans le premier sprint

#### Tableau des tâches du premier sprint :

Le tableau suivant résume les tâches à effectuer durant la réalisation du sprint « Génération des données » ainsi que l’état de chaque tâche (to do, In progress, done) et l’estimation des tâches par jour.

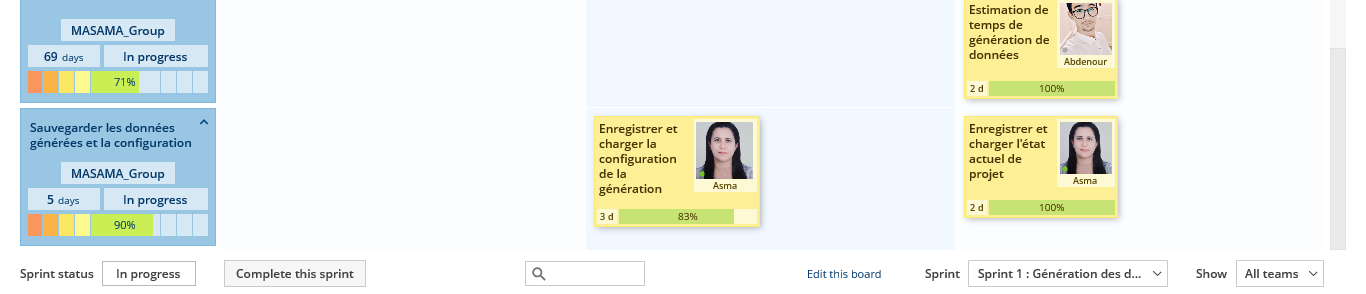
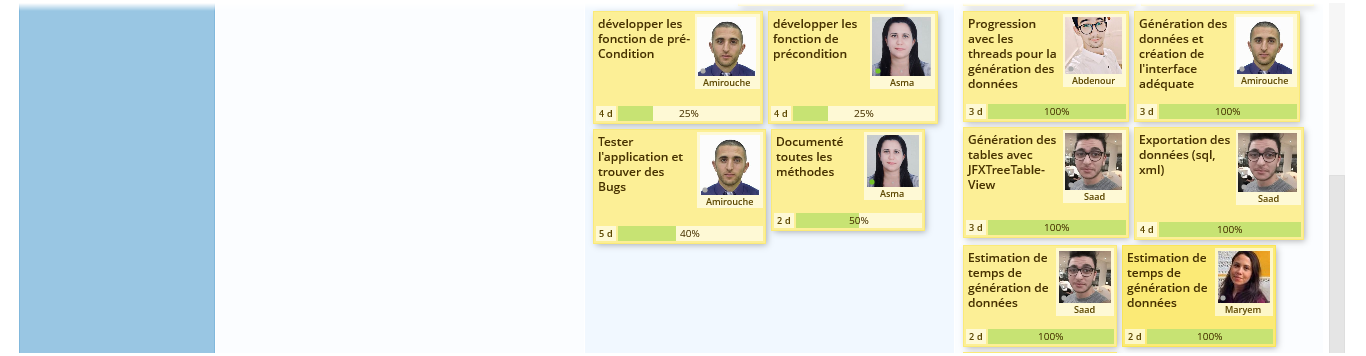
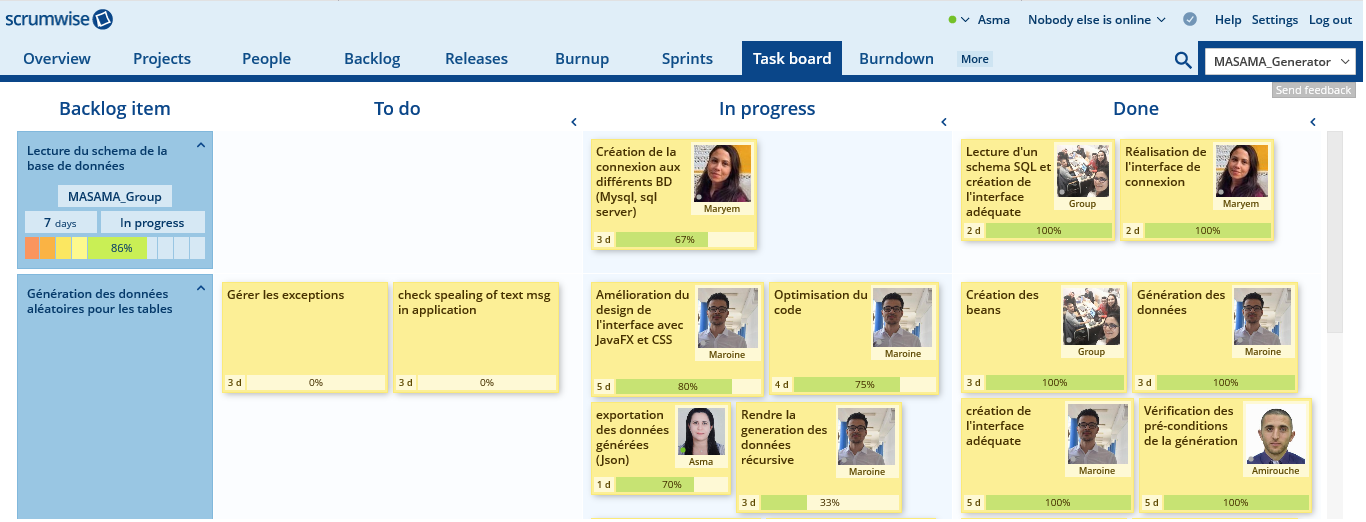


Figure 10 : Tableau des tâches à effectuer durant la réalisation

Conclusion

Pendant la réalisation de notre projet, nous avons travaillé avec des logiciels open sources pour avoir une plateforme base de données riches en services et facilement extensibles.

Pour cela, nous avons procédé, en premier lieu, à une recherche théorique ciblée afin de comprendre les notions de base que traite ce projet, à savoir les étapes principales à suivre pour le bon développement de notre application.

En deuxième lieu, nous avons procédé à la conception des différentes parties du projet, c’est-à-dire la création de notre diagramme de classe et surtout le développement des différents algorithmes sur lesquels se base notre code.

Le cadre de ce projet nous a permis aussi de nous confronter à plusieurs types de problèmes d’organisation, de compréhension, répartition des tâches, respect du temps et surtout de les résoudre avec patience et courage.

Il nous a appris également comment optimiser notre temps et travailler en groupe dans le but d’apprendre et de réaliser au mieux notre projet.

6 .https://github.com/fzaninotto/Faker

6 .https://github.com/fzaninotto/Faker

Références

|  |  |
| --- | --- |
| 1 | https://fr.wikipedia.org/wiki/G%C3%A9n%C3%A9rateur\_de\_nombres\_al%C3%A9atoire. |
| 2 | https://fr.wikipedia.org/wiki/Java\_(langage). |
| 3 | https://netbeans.org/kb/. |
| 4 | https://openclassrooms.com/fr/courses/1469021-les-applications-web-avec-javafx/1469344-presentation-de-linterface-graphique-en-javafx. |
| 5 | https://github.com/DiUS/java-faker. |