PROJET TUTORÉ

**Rapport n°1**

**Définition des rôles, élaboration des spécifications techniques et fonctionnelles**

ligne horizontale

# 

ligne horizontale

Responsable : POUZIN Pierre-Emmanuel Groupe n°1

Table des Matières

[**1) Contexte du projet et de ses objectifs**](#_r7v84nl4swla) **3**

[**2) Fonctionnalités et interface**](#_s97yfbjc3ve6) **3**

[2.1 - Fonctionnalités envisagées](#_3zlwjx8eb5f1) 3

[2.2 - Fonctionnalités de l'interface](#_jgy32fi5j922) 4

[**3) Dépendances entre les fonctionnalités**](#_dfenqykh883k) **6**

[**4) Organisation du groupe**](#_dvn1s5dxt1rq) **6**

[4.1 - Attribution des rôles](#_1ayyno2xnux4) 6

[4.2 - Répartition des rôles sur chaque fonctionnalité et affectation des tâches](#_a9ajl3s0phbo) 7

[4.3 - Estimation du temps](#_on7azfyma5go) 8

[**5) Choix techniques et outils utilisés**](#_ro5cdf8q837o) **8**

[**6) Stockage des données**](#_2w5qa6tf2g6o) **9**

[6.1 - Contexte](#_mfhrtjvdz1c1) 9

[6.2 - Architecture générale du schéma de stockage](#_r16s2nvo47yq) 10

[6.3 - Technologie utilisée](#_rqri0rvoafe) 11

[6.4 - Processus d’ingestion des données](#_22db0kiuli8q) 11

## 

## 

## 1) Contexte du projet et de ses objectifs

Le but de ce projet est de développer une application autour des échecs, capable de fournir une aide à la décision durant une partie. Pour chaque coup joué, le système exploitera les données basées sur une grande quantité de parties existantes obtenues depuis la base de données de Lichess, et donnera des indications sur le coup et des recommandations sur les coups suivants possibles.

L'application proposera également un moteur permettant de disputer des parties contre des joueurs, assisté ou non par d'autres moteurs de jeu comme Stockfish. De plus, elle permettra de détecter les tricheurs en se basant sur l'analyse des coups.

L'application implémentera un client permettant d'interagir avec L'API de Lichess afin d'obtenir les statistiques des joueurs et des parties, et de réaliser la connexion entre les comptes des utilisateurs de l'application et leurs comptes Lichess.

Pour inclure la collection de données de l’API à notre base de données, nous avons besoin de développer un programme ETL (*Extract, Transform, Load)* afin de filtrer les données entrantes, et de garder celles qui nous seront utiles.

## 2) Fonctionnalités et interface

### 2.1 - Fonctionnalités envisagées

Comme explicité dans la partie précédente, l'application implémentera les fonctionnalités principales suivantes.

**L'affichage des parties des joueurs :** L'application permettra de visualiser les parties réalisées par les joueurs sur Lichess, par l'intermédiaire de l'API.

**Le calcul des statistiques générales des joueurs et des parties :** À l'aide des données de Lichess, l'application pourra établir des statistiques générales sur le comportement des joueurs et sur le déroulé des parties à différents niveaux d'ELO.

**Un système d'aide à la décision :** Un moteur de jeu pourra analyser la partie et afficher des indications et des recommandations de coups en fonction de l'état de l'échiquier.

**Un système d'anti-triche :** Le moteur de jeu analysera les coups joués par les joueurs, et détectera les tricheurs qui utilisent des robots. Le nombre de coups « excellents » joués, et les coups « anormaux » par rapport aux coups habituellement joués au niveau de l'ELO du joueur pourraient faire partie des paramètres à prendre en compte.

Nous ne savons pas encore si la détection se fera en cours de partie ou en fin de partie.

En effet, si nous devons détecter en cours de partie, on pourrait “ban” la personne dès X bons coups joués alors que si on le fait en fin de partie, se serait sur un Ratio nb\_coup\_total/nb\_bon\_coup×100.

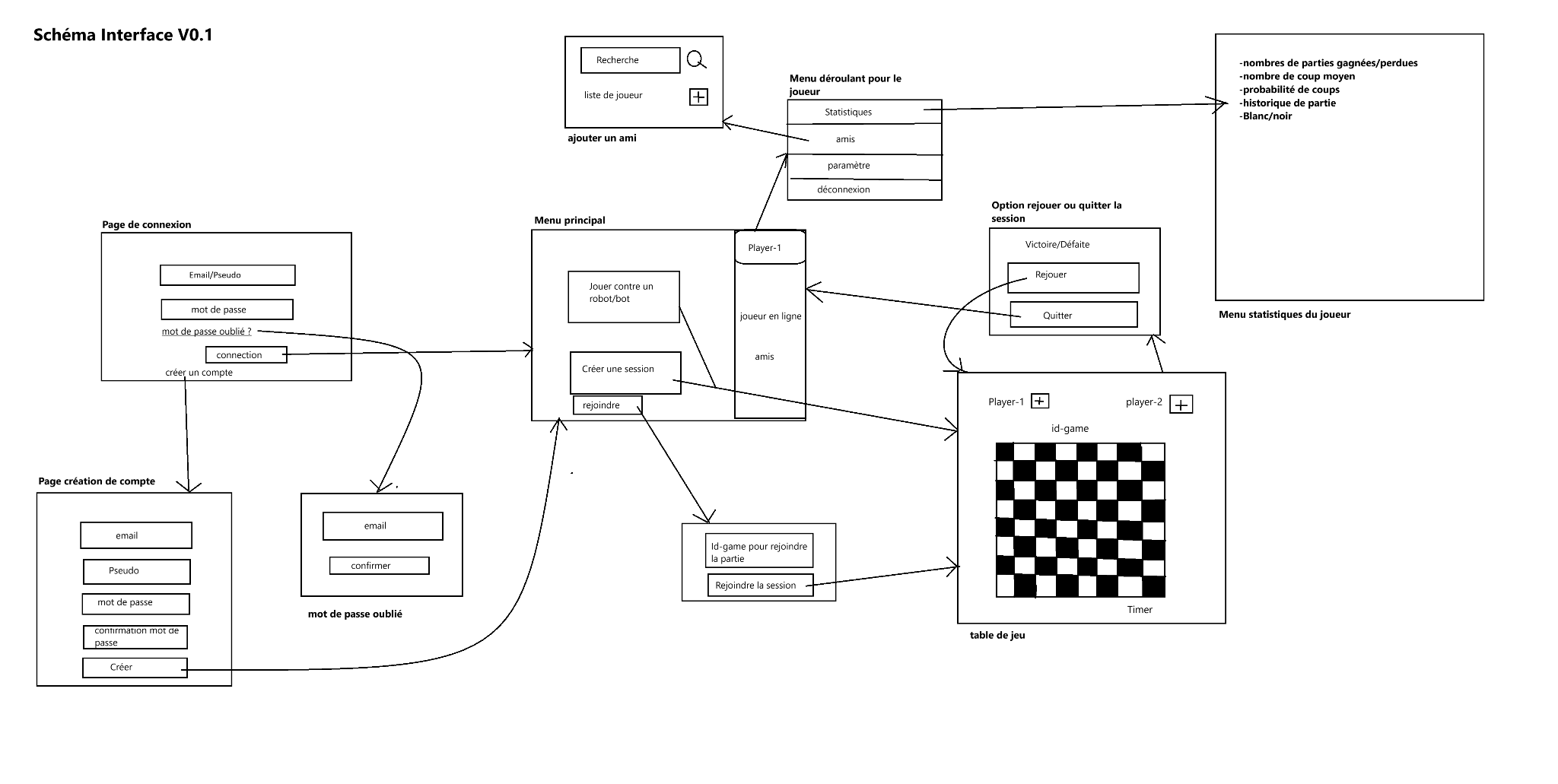
**Le jeu contre l'ordinateur :** Les utilisateurs pourront disputer des parties contre le moteur de jeu de l'application, qui sera assisté ou non par d'autres moteurs (comme Stockfish). Le système d'anti-triche présenté ci-dessus pourrait rendre le moteur plus réaliste en lui faisant jouer des coups proches de ceux que jouent habituellement les humains.

### 2.2 - Fonctionnalités de l'interface

L'interface de l'application disposera de plusieurs fonctionnalités supplémentaires, mises à disposition de l'utilisateur.

* L'utilisateur aura la possibilité de se connecter ou de se créer un compte lié à l'API Lichess.
* En plus de pouvoir jouer contre un robot, l'utilisateur pourra affronter un autre humain.
* Le joueur pourra consulter ses statistiques et son historique de partie.

Éventuellement, d'autres fonctionnalités supplémentaires pourront être implémentées, telles que les listes d'amis, la mise en place d'un chat écrit, ou une présentation des règles du jeu. Il faut cependant souligner que ces fonctionnalités ne sont pas définitives, de la même manière que le schéma de l'interface ci-dessous n'en est qu'à sa première version.



## 3) Dépendances entre les fonctionnalités

Certaines fonctionnalités dépendent d'autres pour leur mise en œuvre. Par exemple, le développement des statistiques doit être achevé avant de pouvoir implémenter les recommandations de coups. De plus, l'intégration de l'API LiChess doit être fonctionnelle pour que la détection de triche soit efficace.

De la même manière, le jeu contre l'ordinateur nécessite que le système de recommandations de coups et d'aide à la décision soit finalisé. Cette fonctionnalité pourrait aussi s'appuyer sur le système d'anti-triche, si l'on souhaite améliorer le robot pour qu'il joue des coups plus proches de ceux des humains (cf 2.1 Fonctionnalités envisagées).

## 4) Organisation du groupe

### 4.1 - Attribution des rôles

| NOM / PRÉNOM | RÔLE |
| --- | --- |
| POUZIN Pierre-Emmanuel | Responsable de groupe |
| GUIBARD Théo | Référent développeur |
| POULET Alexandre | Animateur de réunion/Gardien du temps |
| GIE Jimmy | Référent qualité |
| STIZ Romain | Référent données |
| ROUSSEAU Emile | Devops |
| VOUETTE Maxence | Secrétaire |
| GOURMAND Nicolas | Alternant |
| TURIN Maxime | Alternant |
| REMBERT Lucas | Alternant |

### 4.2 - Répartition des rôles sur chaque fonctionnalité et affectation des tâches

Les premières tâches ont déjà été planifiées pour préparer l'environnement de travail, le traitement des données et la conception de l'application.

**Spécification du format ETL**

La tâche a été attribuée à **Romain STIZ (Référent données).** En tant que référent données, Romain est responsable de la gestion des aspects liés à la manipulation et au traitement des données. Le processus ETL étant une tâche critique pour récupérer, filtrer, et insérer les données dans la base, cela correspond parfaitement à son rôle de spécialiste des données.

**Initialisation GitLab et création du projet Spring Boot**

La tâche a été attribuée à **Pierre-Emmanuel POUZIN (Responsable de groupe).** En tant que responsable de groupe, Pierre-Emmanuel supervise l'ensemble du projet et est donc naturellement impliqué dans les étapes d'infrastructure initiales, telles que la configuration de GitLab et la création du projet backend avec Spring Boot. Cela permet de mettre en place un environnement de développement solide pour le reste de l'équipe.

**Implémentation de la première route GET**

La tâche a été attribuée à **Théo GUIBARD (Référent développeur).** Théo, en tant que référent développeur, est le mieux placé pour commencer à implémenter les fonctionnalités backend, comme la première route GET. Cette tâche nécessite de bonnes connaissances en développement backend, ce qui correspond à son expertise.

**Développement du frontend Home**

La tâche a été attribuée à **Maxence VOUETTE (Secrétaire).** Bien que Maxence soit secrétaire, cela n'empêche pas de lui attribuer des tâches techniques comme le développement de l'interface front end. S'il a des compétences en développement frontend, cette tâche lui permet de contribuer efficacement tout en respectant son rôle organisationnel dans le groupe.

**Setup Docker pour React et Spring Boot**

La tâche a été attribuée à **Emile ROUSSEAU (DevOps).** Emile est responsable DevOps, il est donc naturellement chargé de la mise en place des environnements de développement et production avec Docker. Son expertise dans la gestion des conteneurs et des environnements de déploiement le rend le plus qualifié pour cette tâche.

**Script de migration BD et schéma**

La tâche a été attribuée à **Romain STIZ (Référent données).** Étant également référent pour les données, Romain est chargé de concevoir et de mettre en œuvre le schéma de la base de données ainsi que le script de migration. Cette tâche s'inscrit directement dans son rôle, car elle concerne la structuration et l'organisation des données au sein de la base de données.

**Référent qualité et développer frontend,backend React**

La tâche a été attribuée à **Jimmy GIE est Référent qualité** du projet tout en étant chargé de développer à la fois le frontend (avec React.js) et le backend. Il devra s'assurer que les fonctionnalités sont conformes aux standards de qualité, effectuer des tests unitaires et d'intégration, et garantir une architecture robuste et maintenable.

### 4.3 - Estimation du temps

Le projet a été divisé en plusieurs phases avec une estimation de temps pour chaque fonctionnalité :

* Spécification ETL : 1 semaine
* Initialisation GitLab + Spring Boot : 2 jours
* Route GET + tests : 3 jours
* Frontend Home : 1 semaine
* Mise en place Docker : 3 jours
* Script de migration et schéma BD : 4 jours

À noter que les délais présentés ci-dessus ne sont pas définitifs et peuvent encore évoluer en fonction des choix effectués et des problèmes rencontrés.

## 5) Choix techniques et outils utilisés

**Spécifications ETL**

Le processus ETL permet de récupérer les données de l'API LiChess, de les filtrer pour n'inclure que les statistiques pertinentes et de les insérer dans la base de données PostgreSQL. L'ETL est implémenté en Java et permet d'automatiser la collecte et l'intégration des données dans le système

**Initialisation GitLab et Spring Boot**

GitLab a été utilisé pour la gestion de version et la collaboration. Un projet Spring Boot a été créé pour gérer le backend, avec une architecture prévue pour le traitement des données des parties d'échecs.

**Mise en place de Docker**

Docker a été utilisé pour containeriser les applications frontend (React) et backend (Spring Boot). Chaque service a son propre Dockerfile, et Docker Compose est utilisé pour orchestrer les différents conteneurs.

**Schéma de la base de données et script de migration**

Un schéma de base de données a été conçu pour stocker les informations relatives aux parties, joueurs et statistiques. Un script de migration a été mis en place pour automatiser la création des tables et des relations dans la base de données PostgreSQL.

**API**

**oui**

## 

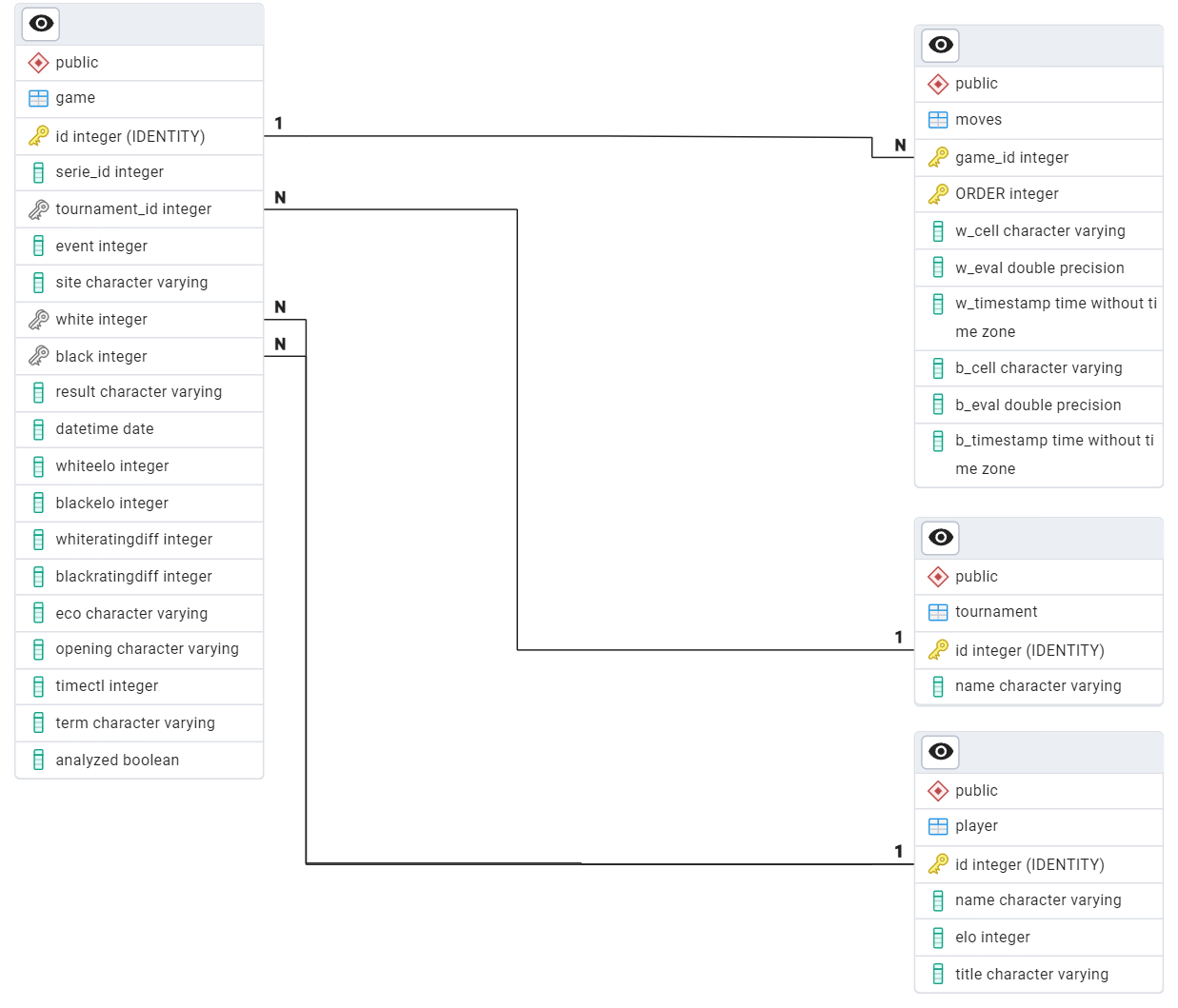
## 6) Stockage des données

### 6.1 - Contexte

Notre projet portant sur un jeu d'échecs, il est important de concevoir un stockage des données qui permet de supporter les futures fonctionnalités de l’application. Nous devons pouvoir les réutiliser à des fins de statistiques, de métriques de coups,...

L'enjeu est d'établir une structure permettant d'accueillir plusieurs giga-octets de données, qui bénéficiera de performances convenables.

### 6.2 - Architecture générale du schéma de stockage



L’objectif de ce schéma est de mettre l’accent sur le tri des données réalisé au préalable des insertions. En effet, ces données, qu’on doit collecter, doivent pouvoir être insérées dans la BD facilement et rapidement. La description du schéma de la base de données est disponible sur le wiki.

Id integer (identity) PK

Nom varchar

Date inscription : date

Titre: varchar

Elo : integer

Nb parties joués : integer

Pourcentage victoire blanc / noir / total : float

Ouverture pref : varchar

Cmb de coup on gagne la partie en moyenne : float

Meilleur mode de jeu: varchar

…

### 6.3 - Technologie utilisée

Nous avons décidé de partir sous PostgreSQL à des fins d’utiliser JDBC (*Java Database Connectivity*) pour relier la base de données à nos programmes Java en backend. Pour cela, nous avons installé la version 16.4 de PostgreSQL et la version 42.7.4 du JDBC à des fins de compatibilité.

Nous avons décidé de choisir ces technologies qui sont plutôt simples à mettre en place et à comprendre. Tout le monde peut travailler dessus.

### 6.4 - Processus d’ingestion des données

Les données récupérées depuis la base de données libre d'accès de LiChess ([lichess.org open database](https://database.lichess.org/#:~:text=lichess.org%20open%20database.%20Lichess%20games%20and%20puzzles%20are%20released%20under)) seront de la forme suivante :

[Event "Rated Bullet tournament https://lichess.org/tournament/yc1WW2Ox"]

[Site "https://lichess.org/PpwPOZMq"]

[Date "2017.04.01"]

[Round "-"]

[White "Abbot"]

[Black "Costello"]

[Result "0-1"]

[UTCDate "2017.04.01"]

[UTCTime "11:32:01"]

[WhiteElo "2100"]

[BlackElo "2000"]

[WhiteRatingDiff "-4"]

[BlackRatingDiff "+1"]

[WhiteTitle "FM"]

[ECO "B30"]

[Opening "Sicilian Defense: Old Sicilian"]

[TimeControl "300+0"]

[Termination "Time forfeit"]

1. e4 { [%eval 0.17] [%clk 0:00:30] } 1... c5 { [%eval 0.19] [%clk 0:00:30] }

2. Nf3 { [%eval 0.25] [%clk 0:00:29] } 2... Nc6 { [%eval 0.33] [%clk 0:00:30] }

3. Bc4 { [%eval -0.13] [%clk 0:00:28] } 3... e6 { [%eval -0.04] [%clk 0:00:30] }

4. c3 { [%eval -0.4] [%clk 0:00:27] } 4... b5? { [%eval 1.18] [%clk 0:00:30] }

5. Bb3?! { [%eval 0.21] [%clk 0:00:26] } 5... c4 { [%eval 0.32] [%clk 0:00:29] }

6. Bc2 { [%eval 0.2] [%clk 0:00:25] } 6... a5 { [%eval 0.6] [%clk 0:00:29] }

7. d4 { [%eval 0.29] [%clk 0:00:23] } 7... cxd3 { [%eval 0.6] [%clk 0:00:27] }

8. Qxd3 { [%eval 0.12] [%clk 0:00:22] } 8... Nf6 { [%eval 0.52] [%clk 0:00:26] }

9. e5 { [%eval 0.39] [%clk 0:00:21] } 9... Nd5 { [%eval 0.45] [%clk 0:00:25] }

10. Bg5?! { [%eval -0.44] [%clk 0:00:18] } 10... Qc7 { [%eval -0.12] [%clk 0:00:23] }

11. Nbd2?? { [%eval -3.15] [%clk 0:00:14] } 11... h6 { [%eval -2.99] [%clk 0:00:23] }

12. Bh4 { [%eval -3.0] [%clk 0:00:11] } 12... Ba6? { [%eval -0.12] [%clk 0:00:23] }

13. b3?? { [%eval -4.14] [%clk 0:00:02] } 13... Nf4? { [%eval -2.73] [%clk 0:00:21] } 0-1

Pour pouvoir les insérer dans nos tables, un programme ETL sera nécessaire afin de filtrer les données.

Comme nous le voyons avec l'exemple du dessus, les données sont regroupées par partie jouée.

Nous allons de ce fait importer une grosse partie de la BD LiChess afin de remplir notre propre BD.

Quant aux informations des joueurs (tels que les statistiques), nous allons plutôt utiliser l’API.