L’équipe avait envisagé au départ de créer la base de données avec un moteur non relationnel, mais ce choix n'a finalement pas été retenu et s'oriente dorénavant vers le moteur relationnel MySQL.

Selon vous quelles sont les raisons qui ont poussé l’équipe à changer de stratégie ?

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Une BDD relationnelle MySQL va être davantage cohérente et disponible de manière à relier les données. Celle-ci va permettre de relier les informations entre elles, les stocker dans des tables, des lignes et des colonnes.

Le fait d’utiliser un langage SQL permet d’interroger les données de façon interactive et de les collecter dans le cadre de rapports relationnels.

Il permet aussi la garantie qu’une transaction se fasse de façon fiable (principe ACID) :

* **Atomicité** : Soit l’opération se fait en entier, soit-elle ne se fait pas du tout. La notion d’atomicité sous-entend la possibilité de défaire une opération avortée.
* **Cohérence** : L’opération doit assurer que la base de données sera dans un état valide après l’opération.
* **Isolation** : L’opération doit se faire en toute autonomie sans dépendance à une autre opération.
* **Durabilité** : En cas de problème important (coupure d’électricité), les modifications apportées sont bien enregistrées.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Comparons les avantages de la BDD relationnelle et les inconvénients de la BDD non relationnelle.

Avantages BDD relationnels (SQL) :

1. La technologie est « mature » (création il y a plusieurs dizaines d’années) ce qui fait qu’aujourd’hui le SQL est un langage standard et normalisé.
2. On a une garantie que les transactions sont atomiques, cohérentes, isolées et durables – principe ACID (Atomic, Consistent, Independant, Durable).
3. La possibilité de mettre en œuvre des requêtes complexes (croisement multiple des données).
4. Du fait du nombres d’années d’existence, un large support de documentation est disponible et il existe également de fortes communautés.

Inconvénients BDD non relationnels (NoSQL) :

1. Il n’existe pas de langage d’interrogation standardisé : chaque éditeur a mis en place le sien.
2. La mise en œuvre d’un environnement fortement transactionnel (fort besoin d’écriture) où le séquencement des écritures est important, reste complexe puisque l’architecture est distribuée compliquant l’atomicité et la cohérence des transactions.
3. L’écriture de requêtes complexes est difficile à mettre en œuvre.
4. L’offre NoSQL est segmentée en plusieurs familles où chacune répond à un besoin précis.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Ressources :

Pour SQL :

<https://buzut.net/maitrisez-mysql-en-cli/>

=> Commandes

<https://dev.mysql.com/doc/refman/8.0/en/datetime.html>

=> Doute sur le type des dates en SQL (string ou number)

Pour le regex :

<https://regex101.com/>

=> Entraînements pour bien comprendre le principe et la syntaxe des regex

<https://stackoverflow.com/questions/22061723/regex-date-validation-for-yyyy-mm-dd/22061800>

=> Pour résoudre trouver comment procéder pour les regex de dates

Pour les BDD :

<https://www.veeam.com/blog/fr/sql-vers-le-nosql.html>

=> Principe ACID

<https://www.veeam.com/blog/fr/sql-vers-le-nosql.html>

=> Pour bien comprendre les différences entre BDD relationnelle et non relationnelle