MAROC ynov CAMPUS | Jobintech

Projet final du module

Les Bases de données

Présenté par:

Lamya Makea

Saad Ouafir

Supervisé par:

Oussama Ettalali

16 October 2025

TABLE DE MATIÈRES

PLAN DÉTAILLÉ

01. Contexte général
02. Idée générale du projet
03. Objectifs et besoins
04. Choix de la technologie et justification
05. Modélisation des données
06. Phases du projet et besoins associés
07. Simulation
08. Conclusion

01.

INTRODUCTION

02.

MySQL – SYSTÈME DE PAIE ET RH

01. Contexte général

GlobalTech Innovations, en pleine croissance, utilise actuellement des feuilles de calcul pour gérer ses ressources humaines et sa paie. Ce système manuel engendre des erreurs de calcul des salaires, complique le suivi des carrières et des équipes, et ne permet aucune analyse approfondie des données. L'entreprise cherche donc à mettre en place une base de données centralisée pour sécuriser, automatiser et analyser ses données RH.

02. Idée générale du projet

- **Créer** une base de données complète pour la gestion des employés et de la paie de entreprise…
- **Ce système** permet d’enregistrer les informations sur les départements, les postes, les employés et leurs fiches de paie mensuelles et il facilite le suivi des salaires, des bonus, et des déductions de manière organisée et fiable.
- **Grâce** à cette base de données relationnelle, l’entreprise peut analyser ses dépenses salariales, suivre la performance du personnel et gérer ses ressources humaines de façon efficace.

03. Objectifs et besoins

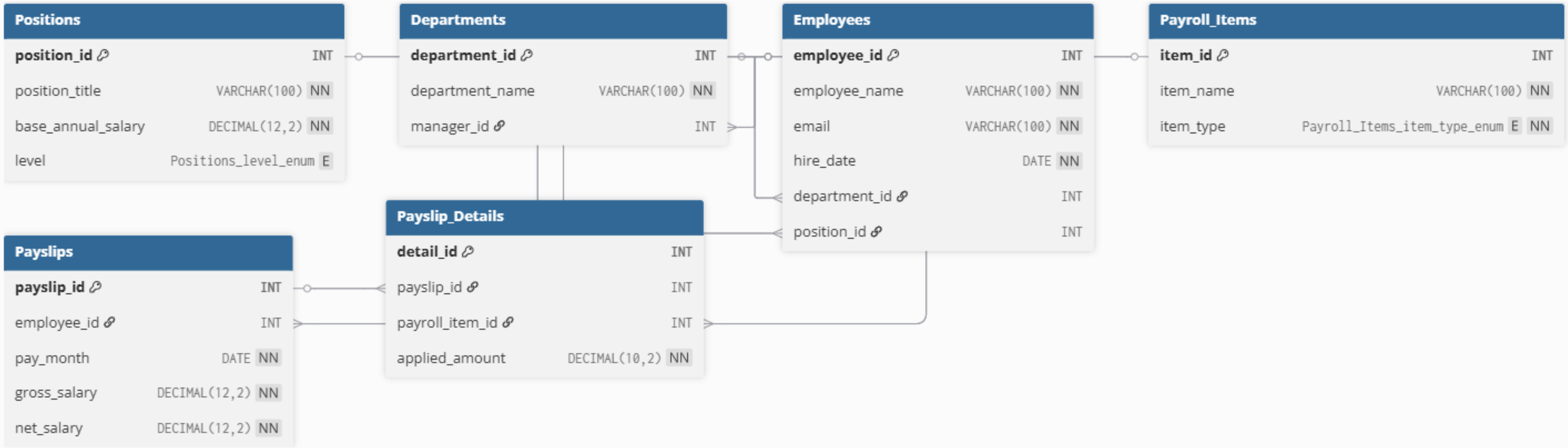
- ✓ **Garantir** la précision des calculs de paie en structurant les salaires, primes et retenues.
- ✓ **Offrir** une vision claire de la structure organisationnelle de l'entreprise.
- ✓ **Permettre** un suivi précis des évolutions de carrière et des changements de poste pour chaque employé.
- ✓ **Fournir** des rapports analytiques pour aider la direction à piloter la masse salariale, analyser la composition des équipes et prendre des décisions stratégiques en ressources humaines.

04. Choix de la technologie

Pourquoi MySQL ?

- ✓ **Créer** des relations plus claires entre les tables.
- ✓ **Utiliser** des clés étrangères (FOREIGN KEY) pour garder la cohérence entre les données.
- ✓ **Ajouter** des contraintes comme CHECK, NOT NULL, UNIQUE, ou AUTO_INCREMENT.
- ✓ **Faire** des jointures (JOIN) complexes entre plusieurs tables.
- ✓ **Faire** des calculs précis (moyenne, somme, classement...) avec GROUP BY, HAVING, AVG(), SUM().
- ✓ **Avoir** une structure fixe et bien définie pour les données.
- ✓ **Exécuter** des requêtes SQL standardisées utilisables dans d'autres SGBD.

05. Modélisation des données



dbdiagram.io

06. Phases du projet et besoins associés

Conception et Analyse

Création de la BD

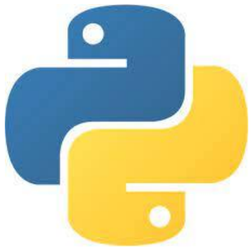
Préparation et Nettoyage de données

Insertion des données

Application des requettes

dbdiagram.io

mockaroo



Python

MySQL Shell

Scripts SQL

07. Simulation

Merci de consulter ce drive contenant la vidéo enregistrée pour la démo

https://drive.google.com/file/d/1kJcRdNr6aAwq938WgMssCregip7kfMNf/view?usp=sharing

-- 1. List all employees hired in 2024, sorted by hire date.

```
mysql> SELECT * FROM Employees WHERE year(hire_date)=2024 ORDER BY hire_date;
+-------------+---------------+-------------------------------+------------+---------------+-------------+
| employee_id | employee_name | email                         | hire_date  | department_id | position_id |
+-------------+---------------+-------------------------------+------------+---------------+-------------+
|          88 | Lorenza       | lyurtsev2f@pinterest.com      | 2024-01-15 |             3 |           8 |
|          51 | Curry         | cturban1e@ovh.net             | 2024-02-03 |             3 |           7 |
|          69 | Cacilie       | cstead1w@furl.net             | 2024-02-06 |             1 |          15 |
|          56 | Drucill       | dsirman1j@weather.com         | 2024-02-08 |             4 |           4 |
|          44 | Danie         | dbricknell17@gizmodo.com      | 2024-03-13 |             2 |           6 |
|          46 | Berte         | bmiddup19@youku.com           | 2024-05-31 |             5 |          12 |
|          30 | Gustie        | gcarasst@i2i.jp               | 2024-06-18 |             1 |           5 |
|          36 | Brigham       | bcamoisz@weather.com          | 2024-06-20 |             2 |           7 |
|          38 | Brigitta      | bgorringe11@engadget.com      | 2024-08-03 |             3 |           7 |
|          25 | Ruthe         | rtremletto@youtube.com        | 2024-08-12 |             2 |           2 |
|          37 | Montgomery    | mnarracott10@cbc.ca           | 2024-09-07 |             3 |           6 |
|          39 | Farris        | fscotchford12@cbslocal.com    | 2024-10-21 |             4 |           9 |
|          73 | Lee           | lloncaster20@google.ru        | 2024-10-23 |             1 |           1 |
|          32 | Nonna         | nvelldenv@amazon.co.jp        | 2024-10-26 |             4 |           5 |
|          83 | Imojean       | ichurchyard2a@baidu.com       | 2024-10-27 |             2 |          14 |
|           9 | Arden         | aferrick8@google.com.au       | 2024-11-11 |             1 |          10 |
|          20 | Gilligan      | gmarronj@altervista.org       | 2024-12-10 |             1 |           1 |
|          76 | Pren          | pbryce23@hhs.gov              | 2024-12-13 |             1 |          14 |
|          29 | Cathy         | caizikovichs@ibm.com          | 2024-12-23 |             3 |           5 |
|          80 | Luella        | lmiell27@intel.com            | 2024-12-31 |             3 |          13 |
+-------------+---------------+-------------------------------+------------+---------------+-------------+
20 rows in set (0.00 sec)
```

-- 2. Find an employee's information by their email address.

```
mysql> SELECT * FROM Employees WHERE email = "smiddup9@tinyurl.com";
+-------------+---------------+-----------------------+------------+---------------+--------------+
| employee_id | employee_name | email                 | hire_date  | department_id | position_id  |
+-------------+---------------+-----------------------+------------+---------------+--------------+
|          10 | Sybyl         | smiddup9@tinyurl.com | 2023-05-18 |             5 |           13 |
+-------------+---------------+-----------------------+------------+---------------+--------------+
1 row in set (0.00 sec)
```

-- 3. List all employees showing their name, position title, and department name.

```
mysql> SELECT e.employee_name, p.position_title, d.department_name FROM Employees e JOIN Positions p ON e.position_id = p.position_
id JOIN Departments d ON e.department_id = d.department_id;
+---------------+----------------+-----------------+
| employee_name | position_title | department_name |
+---------------+----------------+-----------------+
| Nalani        | Mrs            | Engineering     |
| Cam           | Rev            | Engineering     |
| Winna         | Honorable      | Engineering     |
| Kristo        | Rev            | Engineering     |
| Sybyl         | Mrs            | Engineering     |
| Petr          | Honorable      | Engineering     |
| Jeromy        | Rev            | Engineering     |
| Ernie         | Honorable      | Engineering     |
| Berte         | Rev            | Engineering     |
| Max           | Dr             | Engineering     |
| Quent         | Rev            | Engineering     |
| Codee         | Rev            | Engineering     |
| Ofella        | Honorable      | Engineering     |
| Farrell       | Mrs            | Engineering     |
| Cristy        | Rev            | Engineering     |
| Harriett      | Rev            | Engineering     |
| Elly          | Rev            | HelpDesk        |
| Duncan        | Dr             | HelpDesk        |
```

-- 4. List all employees in the 'Engineering' department.

```
mysql> SELECT e.employee_name, e.email, e.hire_date FROM Employees e JOIN Departments d ON e.department_id = d.department_id WHERE
d.department_name = 'Engineering';
+---------------+---------------------------------+------------+
| employee_name | email                           | hire_date  |
+---------------+---------------------------------+------------+
| Nalani        | nguidone0@wikipedia.org         | 2020-04-22 |
| Cam           | crolin1@forbes.com              | 2020-03-04 |
| Winna         | wduiged3@vk.com                 | 2021-04-25 |
| Kristo        | kdwyr5@irs.gov                  | 2022-02-27 |
| Sybyl         | smiddup9@tinyurl.com            | 2023-05-18 |
| Petr          | pgandertonk@blog.com            | 2025-06-05 |
| Jeromy        | jroganm@yolasite.com            | 2022-10-22 |
| Ernie         | elongleyn@rediff.com            | 2023-04-02 |
| Berte         | bmiddup19@youku.com             | 2024-05-31 |
| Max           | mverdey1d@fema.gov              | 2020-06-20 |
| Quent         | qashleigh1f@deliciousdays.com   | 2021-08-28 |
| Codee         | cmorat1u@altervista.org         | 2023-02-10 |
| Ofella        | omccanny21@fda.gov              | 2020-03-10 |
| Farrell       | fgilstin28@usnews.com           | 2025-07-31 |
| Cristy        | cwaliszewski2i@ftc.gov          | 2025-05-01 |
| Harriett      | hgilcriest2o@ebay.co.uk         | 2021-11-01 |
+---------------+---------------------------------+------------+
16 rows in set (0.00 sec)
```

-- 5. Display a detailed payslip for a given employee and month,

```
mysql> select pls.payslip_id, e.employee_name, pls.net_salary, pri.item_name, pri.item_type, pd.applied_amount from payslips pls in
ner join employees e on pls.employee_id=e.employee_id inner join Payslip_Details pd on pls.payslip_id=pd.payslip_id inner join Payr
oll_items pri on pri.item_id=pd.payroll_item_id where e.employee_id=7 and YEAR(pls.pay_month) = 2025 and MONTH(pls.pay_month)=6;
+------------+---------------+------------+-----------+-----------+----------------+
| payslip_id | employee_name | net_salary | item_name | item_type | applied_amount |
+------------+---------------+------------+-----------+-----------+----------------+
|        230 | Bronson       |    5245.90 | Hort      | Deduction |         397.31 |
+------------+---------------+------------+-----------+-----------+----------------+
1 row in set (0.00 sec)

mysql>
```

-- 6. Identify employees who have never received a 'Performance Bonus'.

```
mysql> select pls.payslip_id,e.employee_id, e.employee_name from payslips pls inner join employees e on pls.employee_id=e.employee_
id inner join Payslip_Details pd on pls.payslip_id=pd.payslip_id inner join Payroll_items pri on pri.item_id=pd.payroll_item_id whe
re pri.item_type!="Bonus";
+------------+-------------+---------------+
| payslip_id | employee_id | employee_name |
+------------+-------------+---------------+
|        172 |          80 | Luella        |
|        104 |          93 | Cointon       |
|         10 |          98 | Terrye        |
|         44 |          84 | Jasper        |
|         46 |          37 | Montgomery    |
|        164 |          24 | Ernie         |
|         85 |          32 | Nonna         |
|        225 |          73 | Lee           |
|         76 |          83 | Imojean       |
|          6 |          65 | Elie          |
|         89 |           9 | Arden         |
|        141 |           9 | Arden         |
|        264 |          58 | Edin          |
|        115 |          75 | Lira          |
|         96 |          35 | Ossie         |
|        191 |          12 | Daune         |
|        183 |          43 | Elga          |
|        204 |          80 | Luella        |
|        125 |          95 | Lay           |
|        254 |          30 | Gustie        |
|         21 |          16 | Berkley       |
|        117 |          19 | Trever        |
|        251 |          52 | Quent         |
|        241 |          79 | Bertine       |
|        157 |          74 | Ofella        |
|        228 |          53 | Lynnette      |
|        137 |          65 | Elie          |
|        194 |          93 | Cointon       |
```

-- 7. Identify departments that currently have no employees.

```
mysql> SELECT d.department_id, d.department_name FROM Departments d LEFT JOIN Employees e ON d.department_id = e.department_id WHER
E e.employee_id IS NULL;
+---------------+-----------------+
| department_id | department_name |
+---------------+-----------------+
|             6 | Law             |
+---------------+-----------------+
1 row in set (0.00 sec)

mysql>
```

-- 8. Count the number of employees in each department.

```
mysql> SELECT d.department_name, COUNT(e.employee_id) AS employee_count FROM Departments d LEFT JOIN Employees e ON d.department_id
 = e.department_id GROUP BY d.department_name;
+--------------------+----------------+
| department_name    | employee_count |
+--------------------+----------------+
| Engineering        |             16 |
| HelpDesk           |             19 |
| Human Resources    |             23 |
| IT                 |             21 |
| Law                |              0 |
| Sales & Mareketing |             21 |
+--------------------+----------------+
6 rows in set (0.00 sec)

mysql>
```

-- 9. Calculate the average base annual salary for each position level.

```
mysql> SELECT p.level, AVG(p.base_annual_salary) AS average_salary FROM Positions p GROUP BY p.level;
+-----------+-----------------+
| level     | average_salary  |
+-----------+-----------------+
| Junior    |   42830.053333  |
| Director  |  218974.586667  |
| Manager   |  161159.256667  |
| Mid-Level |   68201.680000  |
| Senior    |  127550.810000  |
+-----------+-----------------+
5 rows in set (0.00 sec)
```

-- 10. List departments whose total annual payroll (sum of base_annual_salary) is greater than €500,000.

```
mysql> SELECT d.department_name, SUM(p.base_annual_salary) AS total_payroll FROM Employees e JOIN Departments d ON e.department_id
= d.department_id JOIN Positions p ON e.position_id = p.position_id GROUP BY d.department_name HAVING total_payroll > 500000;
+------------------+---------------+
| department_name  | total_payroll |
+------------------+---------------+
| Engineering      |    1503129.76 |
| HelpDesk         |    1724594.00 |
| Human Resources  |    2551184.88 |
| IT               |    2335698.01 |
| Sales & Mareketing |    1973531.57 |
+------------------+---------------+
5 rows in set (0.00 sec)
```

-- 11. Calculate the total gross payroll paid by the company for each month.

```
mysql> SELECT DATE_FORMAT(pay_month, '%Y-%m') AS month, SUM(gross_salary) AS total_gross_payroll FROM Payslips GROUP BY DATE_FORMAT
(pay_month, '%Y-%m') ORDER BY month;
+---------+----------------------+
| month   | total_gross_payroll |
+---------+----------------------+
| 2021-01 |             30175.52 |
| 2021-02 |             37615.86 |
| 2021-03 |             15565.66 |
| 2021-04 |              2065.11 |
| 2021-05 |             34670.14 |
| 2021-06 |             28903.22 |
| 2021-07 |             38186.38 |
| 2021-08 |             30605.64 |
| 2021-09 |              4497.61 |
| 2021-10 |             22235.11 |
| 2021-11 |             16506.66 |
| 2021-12 |             15872.32 |
| 2022-01 |             20992.03 |
| 2022-02 |             31321.28 |
| 2022-03 |             22050.39 |
| 2022-05 |             19168.98 |
| 2022-06 |             26448.74 |
| 2022-07 |             14886.22 |
| 2022-08 |             27513.58 |
| 2022-09 |             31131.05 |
| 2022-10 |             22046.02 |
| 2022-11 |             13926.97 |
| 2022-12 |             24950.36 |
| 2023-01 |             23349.51 |
| 2023-02 |             30618.05 |
```

-- 12. Find the top 5 employees with the highest net salary on the last recorded payslip.

```
mysql> SELECT e.employee_name, psl.net_salary, psl.pay_month FROM Employees e INNER JOIN Payslips psl ON psl.employee_id = e.employ
ee_id WHERE YEAR(psl.pay_month)=YEAR(CURDATE()) ORDER BY psl.net_salary  DESC LIMIT 5;
+---------------+------------+------------+
| employee_name | net_salary | pay_month  |
+---------------+------------+------------+
| Betteanne     |    9581.80 | 2025-01-31 |
| Trever        |    9471.01 | 2025-08-11 |
| Gamaliel      |    9292.16 | 2025-06-26 |
| Berte         |    8793.93 | 2025-02-01 |
| Lee           |    8566.39 | 2025-01-20 |
+---------------+------------+------------+
5 rows in set (0.00 sec)

mysql>
```

-- increase 'Senior Developer' base_annual_salary by 5%.

```
mysql> UPDATE Positions SET base_annual_salary = base_annual_salary * 0.05 WHERE level = 'Senior';
Query OK, 1 row affected, 1 warning (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 1

mysql>
```

-- 2. UPDATE: An employee is promoted.

```
mysql> UPDATE Employees SET position_id = (SELECT position_id FROM Positions WHERE level = 'Manager' LIMIT 1) WHERE employee_id = 1
1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

-- 3. DELETE: An employee has left the company.

```
mysql> DELETE FROM Payslip_Details
    -> WHERE payslip_id IN (
    ->   SELECT payslip_id FROM Payslips WHERE employee_id = 15
    -> );
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> DELETE FROM Payslips WHERE employee_id = 15;
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> DELETE FROM Employees WHERE employee_id = 15;
Query OK, 1 row affected (0.01 sec)

mysql>
```

08. Conclusion

En conclusion, ce projet démontre comment une base de données relationnelle peut gérer efficacement les informations sur les employés, leurs postes et leurs paies. Il permet de garantir la cohérence des données, de suivre les salaires et bonus, et de faciliter l’analyse financière et la prise de décision au sein de l’entreprise.

03.

MongoDB – BACKEND POUR GESTION DE DONNÉES IOT

01. Idée générale du projet

- **SmartInfra Solutions** installe des milliers de capteurs IoT dans les bâtiments intelligents.
- **Les systèmes classiques** ne suffisent plus à gérer ces volumes ni cette diversité
- **L'entreprise** recherche une base **NoSQL** adaptée aux données temporelles

02. Contexte général

- **Le projet** concerne **la gestion des données IoT** dans les **bâtiments intelligents.**
- **L’ampleur et la diversité** des flux générés rendent les systèmes traditionnels inadaptés.
- **L’adoption d’une base NoSQL** est ainsi nécessaire pour traiter ces données en temps réel et favoriser la maintenance prédictive ainsi que l’optimisation énergétique.

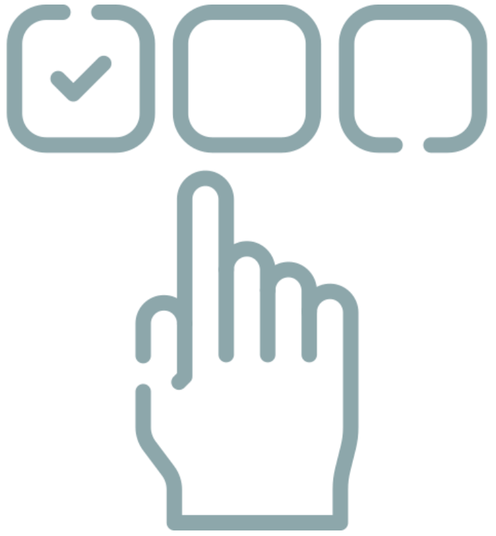
03. Objectifs et besoins

Concevoir et implémenter une base de données documentaire avec MongoDB.

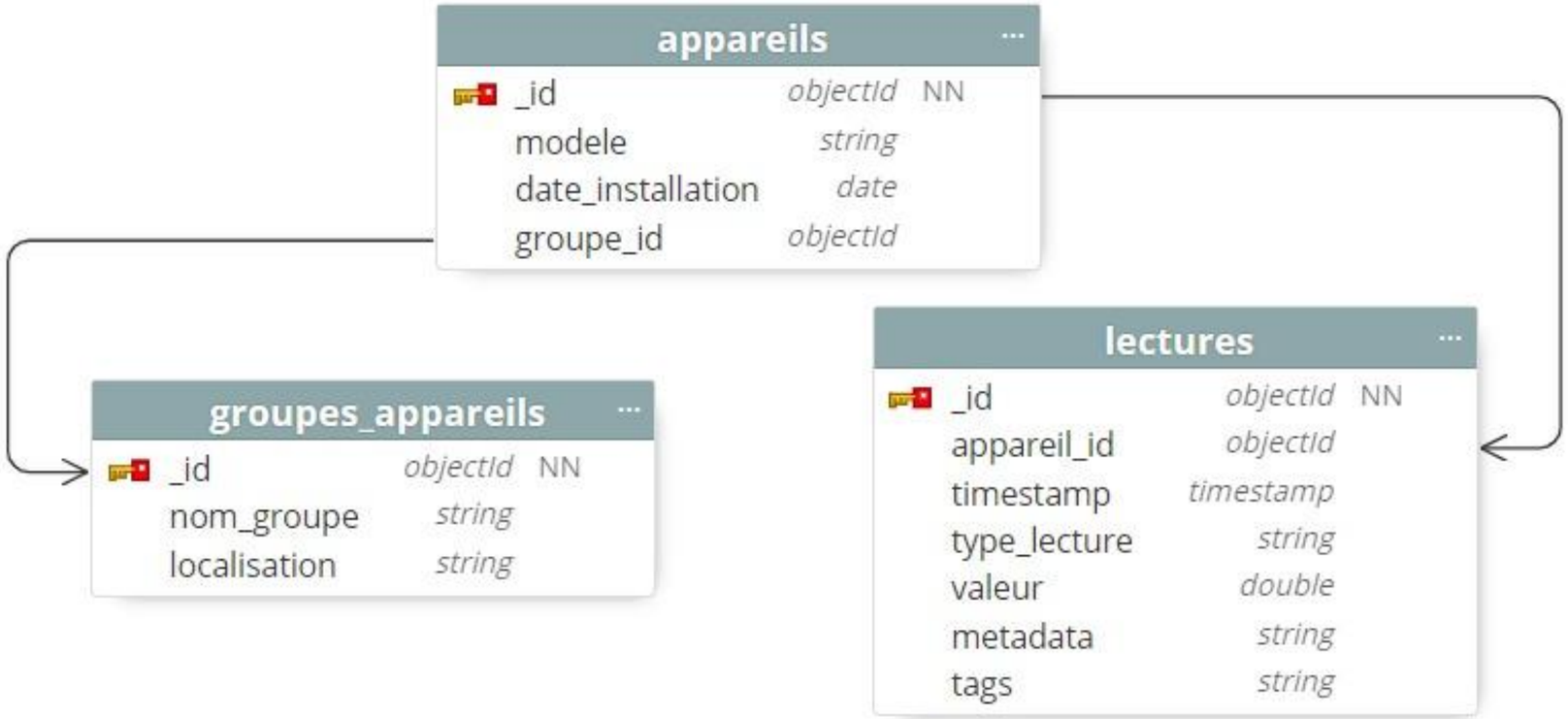
- ✓ Modélisation
- ✓ Gestion
- ✓ Permission des requêtes rapides
- ✓ Fourniture des outils analytiques

04. Choix de la technologie et justification

- ✓ Flexibilité et adaptation
- ✓ Scalabilité et performance
- ✓ Analytique en temps réel



05. Modélisation des données



06. Phases du projet et besoins associés

- **Phase 1**
- Phase 2
- Phase 3
- Phase 4

🎯 **Objectif :** Initialiser la base de données avec les collections de référence.

🎯 **Besoins associés**

- **groupes_appareils**: regroupe les différents ensembles d'appareils (groupes de capteurs).
- **appareils** : contient les métadonnées de chaque appareil (identifiant, type, emplacement, etc.).

```
>_MONGOSH
smartinfra> db.groupes_appareils.insertMany([
              {
                nom_groupe: "Batiment-A-Etage-1",
                localisation: "Paris, France"
              },
              {
                nom_groupe: "Batiment-A-Etage-2",
                localisation: "Paris, France"
              },
              {
                nom_groupe: "Batiment-B-RDC",
                localisation: "Lyon, France"
              },
              {
                nom_groupe: "Datacenter-Principal",
                localisation: "Marseille, France"
              },
              {
                nom_groupe: "Entrepot-Logistique",
                localisation: "Lille, France"
              }
            ]);

            // Affichage des groupes insérés
            db.groupes_appareils.find().pretty();
```

```
>_MONGOSH
smartinfra> db.appareils.insertMany([
              // Batiment-A-Etage-1 (5 appareils)
              { modele: "TempSensor-V2", date_installation: ISODate("2024-10-01"), groupe_id: ObjectId('68ecfad57b80b3506a64af48') },
              { modele: "HumiditySensor-X1", date_installation: ISODate("2024-10-01"), groupe_id: ObjectId('68ecfad57b80b3506a64af48') },
              { modele: "MotionDetector-Pro", date_installation: ISODate("2024-10-02"), groupe_id: ObjectId('68ecfad57b80b3506a64af48') },
              { modele: "TempSensor-V2", date_installation: ISODate("2024-10-05"), groupe_id: ObjectId('68ecfad57b80b3506a64af48') },
              { modele: "CO2Sensor-Advanced", date_installation: ISODate("2024-10-07"), groupe_id: ObjectId('68ecfad57b80b3506a64af48') },

              // Batiment-A-Etage-2 (4 appareils)
              { modele: "TempSensor-V2", date_installation: ISODate("2024-10-03"), groupe_id: ObjectId('68ecfad57b80b3506a64af49') },
              { modele: "HumiditySensor-X1", date_installation: ISODate("2024-10-03"), groupe_id: ObjectId('68ecfad57b80b3506a64af49') },
              { modele: "MotionDetector-Pro", date_installation: ISODate("2024-10-04"), groupe_id: ObjectId('68ecfad57b80b3506a64af49') },
              { modele: "TempSensor-V3", date_installation: ISODate("2024-10-08"), groupe_id: ObjectId('68ecfad57b80b3506a64af49') },

              // Batiment-B-RDC (5 appareils)
              { modele: "TempSensor-V2", date_installation: ISODate("2024-09-20"), groupe_id: ObjectId("68ecfad57b80b3506a64af4a") },
              { modele: "HumiditySensor-X1", date_installation: ISODate("2024-09-21"), groupe_id: ObjectId("68ecfad57b80b3506a64af4a") },
              { modele: "MotionDetector-Pro", date_installation: ISODate("2024-09-22"), groupe_id: ObjectId("68ecfad57b80b3506a64af4a") },
              { modele: "TempSensor-V3", date_installation: ISODate("2024-09-25"), groupe_id: ObjectId("68ecfad57b80b3506a64af4a") },
              { modele: "CO2Sensor-Advanced", date_installation: ISODate("2024-09-28"), groupe_id: ObjectId("68ecfad57b80b3506a64af4a") },

              // Datacenter-Principal (4 appareils)
              { modele: "TempSensor-V3", date_installation: ISODate("2024-09-15"), groupe_id: ObjectId("68ecfad57b80b3506a64af4b") },
              { modele: "HumiditySensor-X2", date_installation: ISODate("2024-09-16"), groupe_id: ObjectId("68ecfad57b80b3506a64af4b") },
              { modele: "TempSensor-V3", date_installation: ISODate("2024-09-18"), groupe_id: ObjectId("68ecfad57b80b3506a64af4b") },
              { modele: "PowerMonitor-Elite", date_installation: ISODate("2024-09-20"), groupe_id: ObjectId("68ecfad57b80b3506a64af4b") },

              // Entrepot-Logistique (2 appareils)
              { modele: "TempSensor-V2", date_installation: ISODate("2024-09-10"), groupe_id: ObjectId("68ecfad57b80b3506a64af4c") },
```

06. Phases du projet et besoins associés

- **Phase 1**
- Phase 2
- Phase 3
- Phase 4

🎯 **Résultats:** Création et insertion de la collection '**appareils**'


```
acknowledged: true,
insertedIds: {
  '0': ObjectId('68ecfd4c7b80b3506a64af4d'),
  '1': ObjectId('68ecfd4c7b80b3506a64af4e'),
  '2': ObjectId('68ecfd4c7b80b3506a64af4f'),
  '3': ObjectId('68ecfd4c7b80b3506a64af50'),
  '4': ObjectId('68ecfd4c7b80b3506a64af51'),
  '5': ObjectId('68ecfd4c7b80b3506a64af52'),
  '6': ObjectId('68ecfd4c7b80b3506a64af53'),
  '7': ObjectId('68ecfd4c7b80b3506a64af54'),
  '8': ObjectId('68ecfd4c7b80b3506a64af55'),
  '9': ObjectId('68ecfd4c7b80b3506a64af56'),
  '10': ObjectId('68ecfd4c7b80b3506a64af57'),
  '11': ObjectId('68ecfd4c7b80b3506a64af58'),
  '12': ObjectId('68ecfd4c7b80b3506a64af59'),
  '13': ObjectId('68ecfd4c7b80b3506a64af5a'),
  '14': ObjectId('68ecfd4c7b80b3506a64af5b'),
  '15': ObjectId('68ecfd4c7b80b3506a64af5c'),
  '16': ObjectId('68ecfd4c7b80b3506a64af5d'),
  '17': ObjectId('68ecfd4c7b80b3506a64af5e'),
  '18': ObjectId('68ecfd4c7b80b3506a64af5f'),
  '19': ObjectId('68ecfd4c7b80b3506a64af60')
```

06. Phases du projet et besoins associés

- **Phase 1**
- Phase 2
- Phase 3
- Phase 4

🎯 **Résultats:** Création et insertion de la collection ‘groupes_appareils’

```
acknowledged: true,
insertedIds: {
  '0': ObjectId('68ecfad57b80b3506a64af48'),
  '1': ObjectId('68ecfad57b80b3506a64af49'),
  '2': ObjectId('68ecfad57b80b3506a64af4a'),
  '3': ObjectId('68ecfad57b80b3506a64af4b'),
  '4': ObjectId('68ecfad57b80b3506a64af4c')
}
```

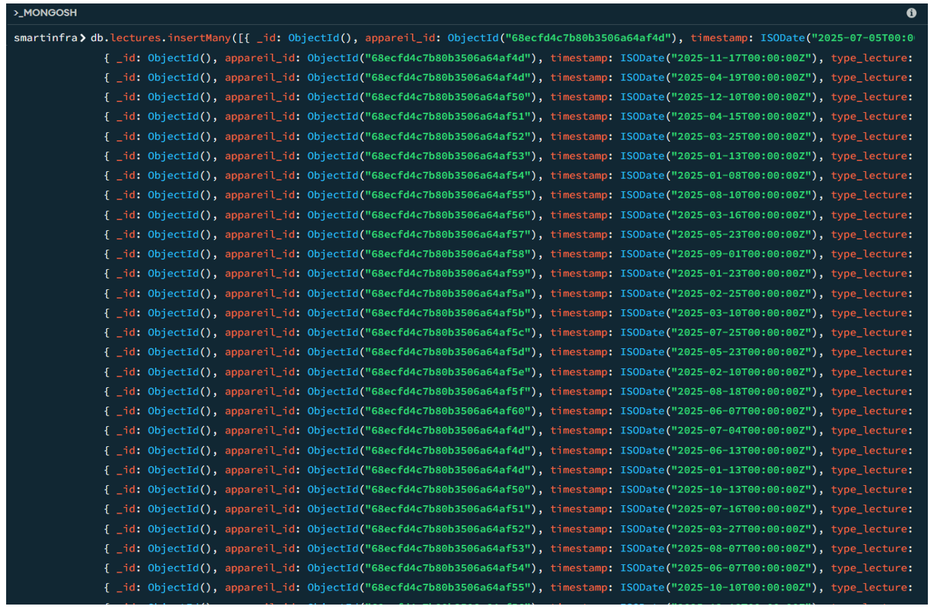
06. Phases du projet et besoins associés

- Phase 1
- **Phase 2**
- Phase 3
- Phase 4

🎯 **Objectif :** Générer un grand volume de lectures de capteurs.

🎯 **Besoins associés**

📩 lectures: Utiliser db.readings.insertMany() pour insérer les données.



06. Phases du projet et besoins associés

- Phase 1
- **Phase 2**
- Phase 3
- Phase 4

🎯 **Résultats: Contenu** de la collection ‘**lectures**’

>_MONGOSH

```
> db.lectures.find().pretty();
< {
    _id: ObjectId('68ed537d7b80b3506a64af61'),
    appareil_id: ObjectId('68ecfd4c7b80b3506a64af4d'),
    timestamp: 2025-07-05T00:00:00.000Z,
    type_lecture: 'pressure',
    valeur: 900,
    metadata: {
      unite: 'hPa'
    },
    tags: [
      'abnormal'
    ]
  }
  {
    _id: ObjectId('68ed537d7b80b3506a64af62'),
    appareil_id: ObjectId('68ecfd4c7b80b3506a64af4d'),
    timestamp: 2025-11-17T00:00:00.000Z,
    type_lecture: 'light',
    valeur: 89700,
    metadata: {
      unite: 'lux'
    },
    tags: [
      'high_alert'
    ]
  }
  {
    _id: ObjectId('68ed537d7b80b3506a64af63'),
```

06. Phases du projet et besoins associés

- Phase 1
- **Phase 2**
- Phase 3
- Phase 4

🎯 **Résultats:** Utilisation de **countDocuments()** pour vérifier le nombre de champs dans les collections

```
>_MONGOSH

> db.appareils.countDocuments()
< 20
> db.groupes_appareils.countDocuments()
< 5
> db.lectures.countDocuments()
< 1000
smartinfra>
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

🎯 **Objectif :** Effectuer des requêtes et agrégations **MongoDB** pour répondre à des questions précises.

🎯 **Besoins associés**

📥 Utilisation des opérateurs : **$match, $lookup, $group, $sort, $limit, $sum**

1ère Requête : Lister toutes les lectures de capteurs enregistrées au cours des dernières 24 heures

```
>_MONGOSH
> db.lectures.find({ timestamp: { $gte: new Date(Date.now() - 24*60*60*1000) } })
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

🎯 **Objectif :** Effectuer des requêtes et agrégations **MongoDB** pour répondre à des questions précises.

🎯 **Besoins associés**

📩 Utilisation des opérateurs : **$match, $lookup, $group, $sort, $limit, $sum**

1ère Requête : Lister toutes les lectures de capteurs enregistrées au cours des dernières 24 heures

```
{
  _id: ObjectId('68ed537d7b80b3506a64af62'),
  appareil_id: ObjectId('68ecfd4c7b80b3506a64af4d'),
  timestamp: 2025-11-17T00:00:00.000Z,
  type_lecture: 'light',
  valeur: 89700,
  metadata: {
    unite: 'lux'
  },
  tags: [
```

06. Phases du projet et besoins associés

2ème Requête : Trouver les métadonnées d'un appareil spécifique en utilisant son _id

```
>_MONGOSH

> db.appareils.findOne({ _id: ObjectId("68ecfd4c7b80b3506a64af4d") })

< {
    _id: ObjectId('68ecfd4c7b80b3506a64af4d'),
    modele: 'TempSensor-V2',
    date_installation: 2024-10-01T00:00:00.000Z,
    groupe_id: ObjectId('68ecfad57b80b3506a64af48')
  }
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

3ème Requête : Lister tous les appareils en affichant leurs métadonnées ainsi que le nom du groupe auquel ils appartiennent **($lookup)**

>_MONGOSH

```
> db.appareils.aggregate([
    {
      $lookup: {
        from: "groupes_appareils",
        localField: "group_id",
        foreignField: "_id",
        as: "group_info"
      }
    },
    {
      $project: {
        _id: 1,
        model: 1,
        install_date: 1,
        group_id: 1,
        group_name: { $arrayElemAt: ["$group_info.group_name", 0] },
        location: { $arrayElemAt: ["$group_info.location", 0] }
      }
    }
  ])
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

3ème Requête : Lister tous les appareils en affichant leurs métadonnées ainsi que le nom du groupe auquel ils appartiennent **($lookup)**

```
< {
    _id: ObjectId('68ecfd4c7b80b3506a64af4d')
  }
  {
    _id: ObjectId('68ecfd4c7b80b3506a64af4e')
  }
  {
    _id: ObjectId('68ecfd4c7b80b3506a64af4f')
  }
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

4ème Requête : Lister toutes les lectures qui ont été étiquetées comme **'high_alert'**

```
>_MONGOSH

> db.lectures.find({ tags: "high_alert" })
```

```
< {
    _id: ObjectId('68ed537d7b80b3506a64af62'),
    appareil_id: ObjectId('68ecfd4c7b80b3506a64af4d'),
    timestamp: 2025-11-17T00:00:00.000Z,
    type_lecture: 'light',
    valeur: 89700,
    metadata: {
      unite: 'lux'
    },
    tags: [
      'high_alert'
    ]
  }
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

5ème Requête : Afficher les 10 lectures les plus récentes pour un **appareil_id** spécifique, triées par **timestamp** décroissant

```
>_MONGOSH
> db.lectures.find({ appareil_id: ObjectId("68ecfd4c7b80b3506a64af4d") })
    .sort({ timestamp: -1 })
    .limit(10)
```

```
< {
    _id: ObjectId('68ed537d7b80b3506a64b105'),
    appareil_id: ObjectId('68ecfd4c7b80b3506a64af4d'),
    timestamp: 2025-12-26T00:00:00.000Z,
    type_lecture: 'pressure',
    valeur: 900,
    metadata: {
      unite: 'hPa'
    },
    tags: [
      'abnormal'
    ]
  }
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

6ème Requête : Identifier les appareils qui n'ont jamais envoyé de lecture **($lookup)**

>_MONGOSH

```
> db.appareils.aggregate([
    {
      $lookup: {
        from: "lectures",
        localField: "_id",
        foreignField: "device_id",
        as: "device_readings"
      }
    },
    {
      $match: {
        device_readings: { $size: 0 }
      }
    },
    {
      $project: {
        _id: 1,
        model: 1,
        install_date: 1,
        group_id: 1
      }
    }
  ])
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

6ème Requête : Identifier les appareils qui n'ont jamais envoyé de lecture **($lookup)**

```
{
  _id: ObjectId('68ecfd4c7b80b3506a64af4d')
}
{
  _id: ObjectId('68ecfd4c7b80b3506a64af4e')
}
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

7ème Requête : Identifier les groupes d'appareils qui ne contiennent aucun appareil enregistré **($lookup).**

>_MONGOSH

```
> db.groupes_appareils.aggregate([
    {
      $lookup: {
        from: "appareils",
        localField: "_id",
        foreignField: "group_id",
        as: "devices_in_group"
      }
    },
    {
      $match: {
        devices_in_group: { $size: 0 }
      }
    },
    {
      $project: {
        _id: 1,
        group_name: 1,
        location: 1
      }
    }
  ])
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

7ème Requête : Identifier les groupes d'appareils qui ne contiennent aucun appareil enregistré **($lookup).**

```
{
  _id: ObjectId('68ecfad57b80b3506a64af48')
}
{
  _id: ObjectId('68ecfad57b80b3506a64af49')
}
{
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

8ème Requête : Compter le nombre d'appareils dans chaque groupe **($group)**

>_MONGOSH

```
> db.appareils.aggregate([
    {
      $group: {
        _id: "$groupe_id",
        device_count: { $sum: 1 },
        devices: { $push: "$modele" }
      }
    },
    {
      $sort: { device_count: -1 }
    }
  ])
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

8ème Requête : Compter le nombre d'appareils dans chaque groupe **($group)**

```
< {
    _id: ObjectId('68ecfad57b80b3506a64af4a'),
    device_count: 5,
    devices: [
      'TempSensor-V2',
      'HumiditySensor-X1',
      'MotionDetector-Pro',
      'TempSensor-V3',
      'CO2Sensor-Advanced'
    ]
  }
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

9ème Requête : Calculer le nombre total de lectures pour chaque type de lecture (**type_lecture**) **($group)**

>_MONGOSH

```
> db.lectures.aggregate([
    {
      $group: {
        _id: "$type_lecture",
        total_readings: { $sum: 1 },
        avg_value: { $avg: "$valeur" },
        min_value: { $min: "$valeur" },
        max_value: { $max: "$valeur" }
      }
    },
    {
      $sort: { total_readings: -1 }
    }
  ])
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

9ème Requête : Calculer le nombre total de lectures pour chaque type de lecture (**type_lecture**) **($group)**


```
{
  _id: 'humidity',
  total_readings: 177,
  avg_value: 48.96079096045197,
  min_value: 0.25,
  max_value: 99.54
}
{
  _id: 'co2',
  total_readings: 174,
  avg_value: 1018.048275862069,
  min_value: 300,
  max_value: 1990
}
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

10ème Requête : Lister les appareils ayant envoyé plus de **100** lectures au total **($group, $match)**

```
>_MONGOSH
      $match: {
        reading_count: { $gt: 100 }
      }
    },
    {
      $lookup: {
        from: "appareils",
        localField: "_id",
        foreignField: "_id",
        as: "device_info"
      }
    },
    {
      $project: {
        device_id: "$_id",
        reading_count: 1,
        model: { $arrayElemAt: ["$device_info.model", 0] },
        install_date: { $arrayElemAt: ["$device_info.install_date", 0] }
      }
    },
    {
      $sort: { reading_count: -1 }
    }
  ])
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

10ème Requête : Lister les appareils ayant envoyé plus de **100** lectures au total **($group, $match)**

```
< {
    _id: ObjectId('68ecfd4c7b80b3506a64af4d'),
    reading_count: 101,
    device_id: ObjectId('68ecfd4c7b80b3506a64af4d')
  }
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

11ème Requête : Calculer le nombre total de lectures enregistrées par jour **($group, $sum)**

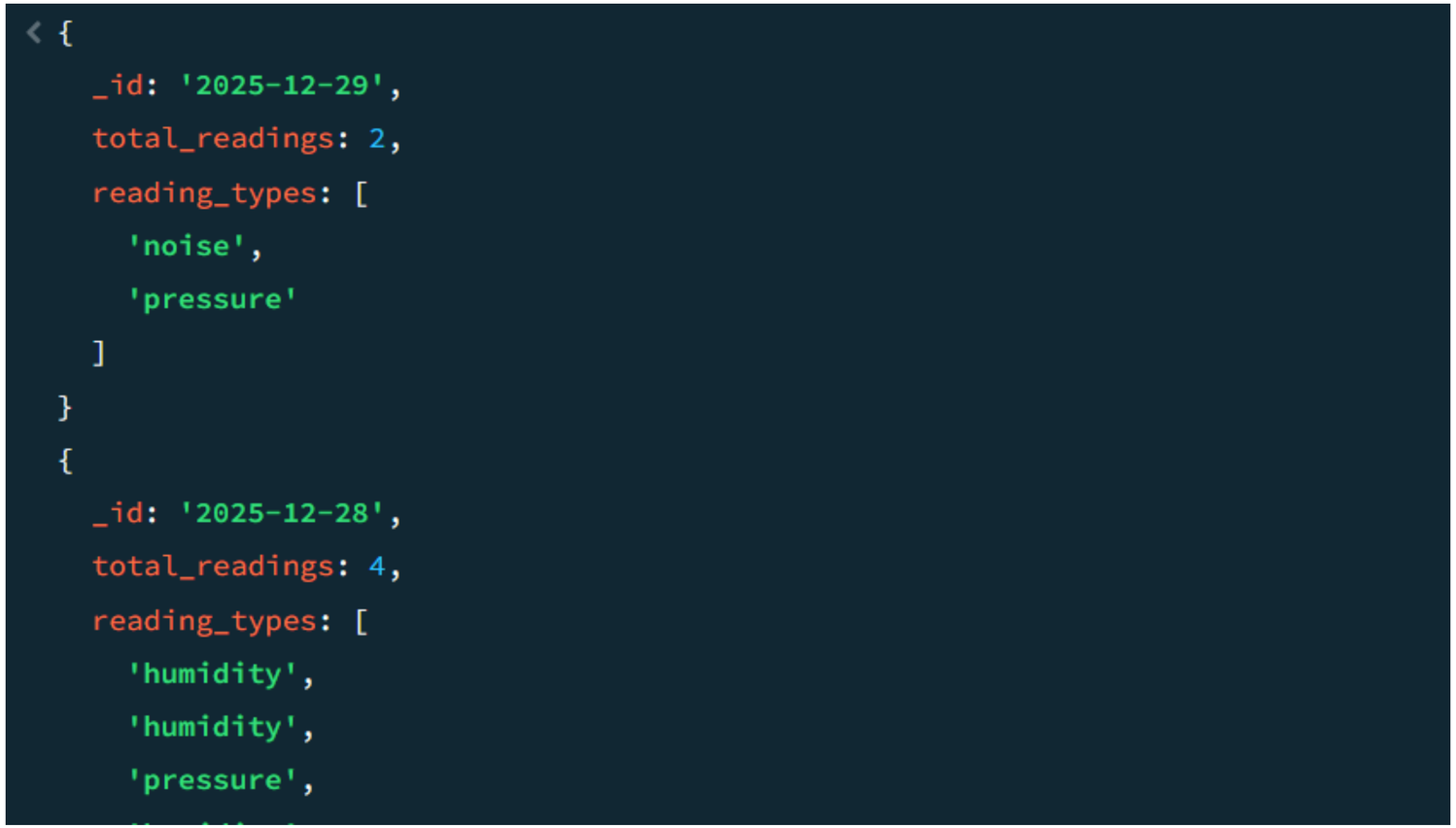
>_MONGOSH

```
> db.lectures.aggregate([
    {
      $group: {
        _id: {
          $dateToString: { format: "%Y-%m-%d", date: "$timestamp" }
        },
        total_readings: { $sum: 1 },
        reading_types: { $push: "$type_lecture" }
      }
    },
    {
      $sort: { _id: -1 }
    }
  ])
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

11ème Requête : Calculer le nombre total de lectures enregistrées par jour **($group, $sum)**



06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

12ème Requête : Trouver les 5 appareils ayant la valeur moyenne la plus élevée pour les lectures de type **'temperature' ($group, $sort, $limit**).

```
>_MONGOSH

> db.lectures.aggregate([
    {
      $match: {
        type_lecture: "temperature"
      }
    },
    {
      $group: {
        _id: "$appareil_id",
        avg_temperature: { $avg: "$valeur" },
        reading_count: { $sum: 1 }
      }
    },
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

12ème Requête : Trouver les 5 appareils ayant la valeur moyenne la plus élevée pour les lectures de type **'temperature' ($group, $sort, $limit**).

```
{
  $lookup: {
    from: "appareils",
    localField: "_id",
    foreignField: "_id",
    as: "device_info"
  }
},
{
  $project: {
    device_id: "$_id",
    avg_temperature: { $round: ["$avg_temperature", 2] },
    reading_count: 1,
    model: { $arrayElemAt: ["$device_info.model", 0] }
  }
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

12ème Requête : Trouver les 5 appareils ayant la valeur moyenne la plus élevée pour les lectures de type **'temperature' ($group, $sort, $limit**).

```
    },
    {
      $sort: { avg_temperature: -1 }
    },
   {
     $limit: 5
   }
])
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

12ème Requête : Trouver les 5 appareils ayant la valeur moyenne la plus élevée pour les lectures de type **'temperature' ($group, $sort, $limit**).

```
< {
    _id: ObjectId('68ecfd4c7b80b3506a64af54'),
    reading_count: 6,
    device_id: ObjectId('68ecfd4c7b80b3506a64af54'),
    avg_temperature: 48.49
  }
  {
    _id: ObjectId('68ecfd4c7b80b3506a64af4e'),
    reading_count: 7,
    device_id: ObjectId('68ecfd4c7b80b3506a64af4e'),
    avg_temperature: 48.06
  }
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- **Phase 3**
- Phase 4

12ème Requête : Trouver les 5 appareils ayant la valeur moyenne la plus élevée pour les lectures de type **'temperature' ($group, $sort, $limit**).

```
< {
    _id: ObjectId('68ecfd4c7b80b3506a64af54'),
    reading_count: 6,
    device_id: ObjectId('68ecfd4c7b80b3506a64af54'),
    avg_temperature: 48.49
  }
  {
    _id: ObjectId('68ecfd4c7b80b3506a64af4e'),
    reading_count: 7,
    device_id: ObjectId('68ecfd4c7b80b3506a64af4e'),
    avg_temperature: 48.06
  }
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- Phase 3
- **Phase 4**

🎯 **Objectif :** Appliquer des opérations de mise à jour et de suppression.

🎯 **Besoins associés**

📩 Utilisation des opérateurs : **updateOne(), $set, $push, deleteMany()**

1ère Requête : Ajouter une étiquette '**critical_alert**' à une lecture (**updateOne, $push**)

>_MONGOSH

```
> db.lectures.updateOne(
    { _id: ObjectId("68ed537d7b80b3506a64af72") },
    { $push: { tags: "critical_alert" } }
  )
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- Phase 3
- **Phase 4**

🎯 **Objectif :** Appliquer des opérations de mise à jour et de suppression.

🎯 **Besoins associés**

📩 Utilisation des opérateurs : **updateOne(), $set, $push, deleteMany()**

1ère Requête : Ajouter une étiquette '**critical_alert**' à une lecture (**updateOne, $push**)

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- Phase 3
- **Phase 4**

🎯 **Objectif :** Appliquer des opérations de mise à jour et de suppression.

🎯 **Besoins associés**

📩 Utilisation des opérateurs : **updateOne(), $set, $push, deleteMany()**

2ème Requête : Mettre à jour le **appareil_id** d'un appareil qui a été déplacé (**updateOne, $set**).

>_MONGOSH

```
> db.appareils.updateOne(
    { _id: ObjectId("68ecfd4c7b80b3506a64af54") },
    { $set: { groupe_id: ObjectId("68ecfad57b80b3506a64af48") } }
  )
```

06. Phases du projet et besoins associés

- Phase 1
- Phase 2
- Phase 3
- **Phase 4**

🎯 **Objectif :** Appliquer des opérations de mise à jour et de suppression.

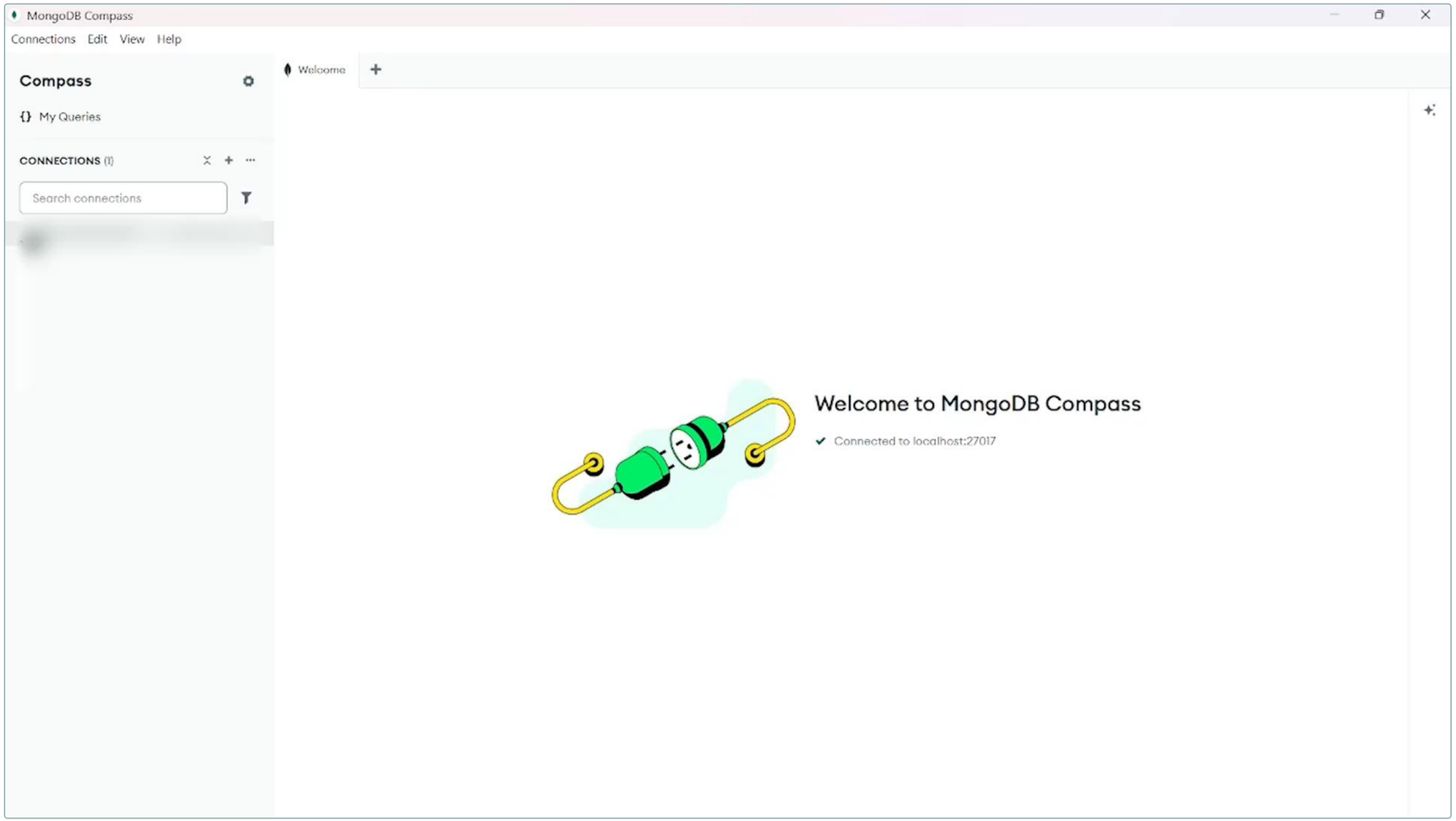
🎯 **Besoins associés**

📥 Utilisation des opérateurs : **updateOne(), $set, $push, deleteMany()**

3ème Requête : Supprimer toutes les lectures datant de plus de 90 jours pour archivage (**deleteMany**).

```
>_MONGOSH
> db.lectures.deleteMany(
    { timestamp: { $lt: new Date(Date.now() - 90*24*60*60*1000) } }
  )
< {
    acknowledged: true,
    deletedCount: 542
  }
```

07. Simulation



08. Conclusion

- ✓ **Efficacité : MongoDB** permet une collecte et un stockage fiables de volumes massifs de données IoT.
- ✓ **Exploitation intelligente :** Les outils analytiques intégrés facilitent la maintenance prédictive et l'optimisation énergétique.
- ✓ **Évolutivité :** La solution offre une base solide et adaptable pour accompagner la croissance des infrastructures intelligentes.

04.

CONCLUSION

MERCI POUR VOTRE ATTENTION