PRESTI Louis

SOUCHET Mathéo

COSSEC Alban

LEFEVRE Lilian

MILLET Arthur

Rapport SAE R4.03 Qualité au delà du relationnel

1 - Notre site et ses données :

- **1.1 - Objectif du site**
  - Nous avons choisi de faire un site web récupérant des données présentes sur des API. L'objectif de notre site web était de réaliser une carte permettant d'afficher les toilettes et les stations de vélos dans la ville de Nantes

- **1.2 - Sources des données**
  - Afin d'atteindre notre objectif nous avons choisi de récupérer les données depuis des API de Nantes métropole.
  - lien : https://data.nantesmetropole.fr
  - lien de nos API :
    - **API Toilettes** : https://data.nantesmetropole.fr/explore/dataset/244400404_toilettes-publiques-nantes-metropole/table/?disjunctive.pole&disjunctive.commune
    - **API Vélos** : https://data.nantesmetropole.fr/explore/dataset/244400404_stations-velos-libre-service-nantes-metropole-disponibilites/table/

- **1.3 - Explication du choix de notre source de données**
  - **1.3.1 - Fiabilité** : En effet, les données sont collectées et vérifiées par la ville de Nantes, Nantes Métropole, le Département de Loire-Atlantique, la Région des Pays de la Loire

et leurs partenaires. Nous pouvons donc en conclure que ce sont des données fiables ce qui va permettre de les exploiter sans avoir à craindre les fausses données.

- **1.3.2 - Qualité** : La fiabilité des sources de données entraîne la qualité. La collectivité s'engage à fournir des données pertinentes et cohérentes comme le mentionne cet extrait des mentions légales du site :

  "La Collectivité s'engage à ce que les données collectées soient adéquates, pertinentes, exactes, protégées et conservées pour une durée limitée strictement nécessaire à la finalité du traitement."

- **1.3.3 - Cohérence** : Comme mentionné précédemment, la collectivité s'engage à collecter des données cohérentes. De plus sans prendre en compte cet aspect les données présentes dans l'API correspondaient parfaitement à l'utilisation prévue. Par exemple, nous souhaitions réaliser une carte donc nous avions besoin des coordonnées. Pour plus de lisibilité pour l'utilisateur nous avions également besoin d'informations complémentaires comme le nom de la station/toilette ou le nombre de vélos disponibles.

  Les données sont également régulièrement actualisées comme le montre cet extrait des mentions légales :

  "Le Site peut faire l'objet de mises à jour fréquentes."

  La base de données des vélos contient également un attribut date de mise à jour ce qui permet de suivre l'avancement de l'évolution des données.

- **1.3.4 - Redondance** : Enfin en ce qui concerne la redondance des données, nous n'avons pas observés de doublons dans les données manipulées.

2 - Données brutes :

2.1 - Table Toilette :

https://data.nantesmetropole.fr/explore/embed/dataset/244400404_toilettes-publiques-nantes-metropole/table/?disjunctive.pole&disjunctive.commune

2.2 - Table Vélo :

https://data.nantesmetropole.fr/explore/embed/dataset/244400404_stations-velos-libre-service-nantes-metropole-disponibilites/table/

3 - Modification des données brutes :

- **3.1 - Choix des données :**
  - Parmi toutes les données proposées par l'API nous n'avions pas toutes les données pour l'utilisation voulue et certaines données ne servaient pas.
  - **3.1.1 - Choix pour la table Toilette :**
    - Pour la table Toilet, nous avons décidé d'enlever différents attributs de la table qui ne nous étaient pas utiles ou que nous ne comprenions pas.
    - Les attributs que nous avons enlevés sont : Pôle, Marque, Propriétaire, Exploitant, Nettoiement, Maintenance.
    - Les données supprimées sont toujours obtenables via l'API si jamais dans le futur nous souhaitons les utiliser.
    - Nous avons également décidé de séparer l'attribut géométrie en deux attributs distincts : latitude et longitude.
    - Nous obtenons donc cette table :

identifiant	**Nom**	**Commune**	**latitude**	**longitude**

  - **3.1.2 - Choix pour la table Vélo:**
    - Idem que pour la table toilette nous avons choisi de garder seulement les attributs dont nous avions besoin.
    - Les attributs que nous avons enlevés sont : banking, bonus, Contract Name, Bike Stands.
    - Nous avons également décidé de séparer l'attribut position en deux attributs distincts : latitude et longitude.
    - Nous obtenons donc cette table :

number	**name**	**adress**	**latitude**	**longitude**	**status**	**Available Bike Stands**	**Avail Bikes**

- **3.2 - Utilisation des données :**

- 3.2.1 - Table toilette :
    - **identifiant** sera la clé primaire de notre table. Il sera donné par un Number.
    - **nom** sera le nom des toilettes. Il sera donné par une VarChar
    - **commune** sera la ville dans laquelle se situe les toilettes. Elle sera donné par un Number
    - **latitude longitude** se passent d'explication

  L'identifiant nous permettra d'identifier chaque toilette, le nom et la commune sont des informations complémentaires que nous donnerons à l'utilisateur. Enfin la géométrie va nous permettre d'afficher les toilettes sur la carte.

- 3.2.2 - Table Vélo :
    - **number** sera la clé primaire de notre table. Il sera donné par un Number.
    - **name** sera le nom de la station de vélo. Il sera donné par une VarChar.
    - **adress** sera l'adresse de la station. Elle sera donnée par une VarChar.
    - **latitude longitude** se passent d'explication
    - **status** sera l'attribut donnant l'état de la station : open ou closed. Il sera donné par une VarChar.
    - **Available Bike Stands** sera les bornes vides. Il sera identifié par un Number.
    - **Available Bikes** sera les vélos disponibles à la station. Il sera identifié par un Number.
    - **Last Update** est la dernière mise à jour de la base de données. Il sera identifié par une Date.

  Le number nous permettra d'identifier chaque station, le nom et l'adresse, le status, le nombre de bornes et de vélos disponibles sont des informations complémentaires que nous donnerons à l'utilisateur. Enfin la position va nous permettre d'afficher les stations sur la carte.

4 - Ajout de tables et modification

- 4.1 - Table Avis

- Pour les besoins de notre site, nous souhaitions que les utilisateurs puissent laisser un avis sur les toilettes et cette donnée n'étant pas présente dans l'API nous avons décidé de rajouter une table Avis.
- Voici la table :
idAvis	#idToilette	ranking
- **idAvis** sera la clé primaire de la table donné par un Number. **idToilette** est une clé étrangère faisant référence à la table toilette et **ranking** est la note donné sur une échelle de 1 à 5.

- **4.2 - Table User**
  - Pour donner des avis il faut être identifié à notre site. Pour cela nous avions donc besoin d'une table User.
login	password	avisdonne
  - **idUser** est la clé primaire de notre table donné par un Number. **login** et **password** permettront d'identifier l'utilisateur grâce à une VarChar. Enfin **avisdonne** sera un document contenant les avis données par chaque utilisateur.

5 - La base de données:

- **5.1 - Type de base de données**
  - Nous obtenons donc une base de données orienté document que nous allons implémenter à l'aide de mongodb.

- **5.2 - Pourquoi cette modélisation ?**
  - Nous avons décidé d'utilisé une base de données orienté document car sa structure est très flexible, notre base évoluant dans le temps cela nous facilitera la tâche dans le futur.
  - De plus les bases orientés document offrent généralement de bonnes performances lorsque les données sont lues à répétition ce qui est le cas dans notre site.
  - Enfin nous trouvé l'accessibilité et la modélisation des données plus naturelle dans une base de données orienté document.

- **5.3 - Formes normales**
    - Nos tables sont toutes en 1NF car chaque attribut contient une seule valeur.
    - Elles sont également en 2NF car elles sont en 1NF et toutes les dépendances fonctionnelles sont basées sur la clé primaire.
    - Enfin elles sont également en 3NF car elles sont en 2NF et aucun attribut ne dépend transitivement de la clé. En effet on pourrait penser que la localisation donne forcément un couple unique mais ce n'est pas forcément le cas car certaines toilettes ont la même localisation et sont pourtant identifiées différemment. Cela est dû à plusieurs facteurs comme le fabricant.
    - Enfin elles sont en 3NFBC car toutes les dépendances non triviales sont entièrement déterminées par la clé candidate.

6 - Utilisation/Création des données dans le site

- **6.1 - findAllToilets, findAllBikes et findAllUsers**
    - Pour les toilettes ainsi que les stations de vélo, nous avons une méthode qui nous permet de retourner toutes les valeurs que contient la base de données.

```
//Requête à la base :

MongoWC.find({}, {_id: 0, __v: 0})
MongoBike.find({},{_id:0,__v:0})
MongoUser.find({},{_id:0})
```

- **6.2 - deleteAllToilets, deleteAllBikes et deleteAllUsers**
    - Ces deux méthodes vont faire un appel à la base de données afin de supprimer tout ce qu'elle contient. Cela nous permet ensuite de réinsérer les données de provenant de l'API. Nous utilisons cette méthode pour mettre régulièrement à jour nos données.

```
MongoWC.deleteMany({})
MongoBike.deleteMany({})
MongoUser.deleteMany({})
```

- **6.3 - findToiletById et findUserById**

- Nous avons une méthode qui nous permet de renvoyer l'objet "toilet" par son identifiant. Nous nous servons de cette méthode afin d'ajouter une note à un objet toilet spécifique ainsi que pour vérifier le login et mot de passe de l'utilisateur.

```
MongoWC.findOne({identifiant: idWC}, {_id: 0})
MongoUser.findOne({login: login})
```

- **6.4 - addToilet, addUser et addAvis**
  - La méthode addToilet permet d'ajouter un objet toilet à la base de données. Pour notre site cette méthode n'est pas utile car nous n'ajoutons pas nous même de toilette, en revanche elle va nous servir pour nos tests.
  - La méthode addAvis va nous permettre d'ajouter un avis émis par un utilisateur dans le document avis.
  - La méthode addUser va nous permettre d'ajouter un utilisateur qui vient de s'enregistrer sur le site.

```
//addToilet
MongoWC.collection.insertOne({
            identifiant: Toilette.identifiant,
            nom: Toilette.nom,
            latitude: Toilette.latitude,
            longitude: Toilette.longitude,
            ranking: Toilette.ranking,
            nbr_avis: Toilette.nbr_avis
        })
//AddUser
const login =user.login
            const hashedPassword = await bcrypt.hash(user.password, 10)
            const newUser = new User({login:login,password:
hashedPassword,avisdonnees:[]})
            const mongoUser = new MongoUser({...newUser})
            await mongoUser.save()

//addAvis
let tab = data.avisdonnees
tab.push({idToilette:idToilette,ranking:note})
MongoUser.updateOne({login:data.login}, {$set:{avisdonnees:tab}})
```

7 - Qualité et évolution dans le temps

- Afin de garder des données cohérentes, nous nous assurons de faire en sorte d'éviter la redondance des données au maximum. De plus nous conservons uniquement les données qui nous intéressent pour éviter la surabondance de données inutiles.
- Notre source de données met régulièrement à jour ces dernières ce qui nous permet de maintenanir la fiabilité et la cohérence.