**HOUKMI Salaheddine** **M2 Data Engineering**
**ROUANET Clément** **Database NoSQL**

# My Social Networks API

**Utilisons la base de données appelée *Social_Networks_API*, grâce à la requête suivante :**

```
use Social_Networks_API;
```

**Ci-dessous, les différentes collections de notre base de données :**

## Utilisateurs :

**Créer un utilisateur :**

- **Opération Create**
- **Méthode 'POST'**
- **Route '/users/create'**

```
db.createCollection( "users",
  {
    validator: {
      $jsonSchema: {
        bsonType: "object",
        required: ["firstname", "lastname", "email", "age"],
        properties: {
          _id: {
            bsonType: "objectId"
          },
          firstname: {
            bsonType: "string",
            description: "Prénom de l'utilisateur"
          },
          lastname: {
            bsonType: "string",
            description: "Nom de l'utilisateur"
          },
          email: {
            bsonType: "string",
            pattern: ".+@.+\..+",
            description: "Email de l'utilisateur"
          },
          age: {
            bsonType: "int",
            minimum: 16,
            description: "Age de l'utilisateur"
          },
          phonenumber: {
            bsonType: "string",
            pattern: "^[0-9]{10}$",
            description: "Numéro de téléphone de l'utilisateur",
          },
          address :{
            bsonType: "object",
            properties: {
              number: { bsonType: "string" },
              street: { bsonType: "string" },
              zipcode: { bsonType: "string" },
              city: { bsonType: "string" },
              country: { bsonType: "string" }
```

```
            },
            description: "Adresse de l'utilisateur",
          }
        },
        additionalProperties: false
      }
    },
    validationAction: "error",
    validationLevel: "strict"
  }
);


db.users.createIndex({ email: 1 }, { unique: true });
```

**Lire les utilisateurs :**

- **Opération Read**
- **Méthode 'GET'**
- **Route '/users/'**

```
db.users.find({});
```

**Mettre à jour un utilisateur :**

- **Opération Update**
- **Méthode 'PUT'**
- **Route '/users/update/<id>'**

```
db.users.updateOne(
  { _id: ObjectId("<id>") },
  {
    $set: {
      firstname: "<nouvelle valeur>",
      lastname: "<nouvelle valeur>",
      email: "<nouvelle valeur>",
      age: <nouvelle valeur>,
      phonenumber: "<nouvelle valeur>",
      "address.number": "<nouvelle valeur>",
      "address.street": "<nouvelle valeur>",
```

```
      "address.zipcode": "<nouvelle valeur>",
      "address.city": "<nouvelle valeur>",
      "address.country": "<nouvelle valeur>"
    }
  }
);
```

**Supprimer un utilisateur :**

- Opération Delete
- Méthode 'DELETE'
- Route '/users/delete/<id>'

```
db.users.deleteOne({ _id: ObjectId("<id>") });
```

**Rechercher un utilisateur :**

- Opération Search
- Méthode 'GET'
- Route '/users/<id>'

```
db.users.find({ _id: ObjectId("<id>") });
```

**Exemple d'insertion d'un utilisateur :**

```
db.users.insertOne(
  {
    firstname: "Jean",
    lastname: "Dupont",
    email: "jean.dupont@gmail.fr",
    age: 24,
    phonenumber: "0638917176",
    address:
   {
     number: "8",
     street: "Boulevard Voltaire",
     zipcode: "75011",
```

```
      city: "Paris",
      country: "France"
    }
   }
);
```

## Évènements :

Créer un évènement :

- Opération Create
- Méthode 'POST'
- Route '/events/create'

```
db.createCollection("events",
  {
    validator: {
      $jsonSchema: {
        bsonType: "object",
        required: ["name", "description", "startdate", "enddate", "place",
                   "organizers"],
        properties: {
          _id: {
            bsonType: "objectId"
          },
          name: {
            bsonType: "string",
            description: "Nom de l'événement"
          },
          description: {
            bsonType: "string",
            description: "Description de l'événement"
          },
          startdate: {
            bsonType: "date",
            description: "Date de début de l'événement"
          },
          enddate: {
            bsonType: "date",
            description: "Date de fin de l'événement"
          },
          place: {
            bsonType: "string",
            description: "Lieu de l'événement"
          },
          organizers: {
            bsonType: "array",
            items: {
              bsonType: "objectId"
            },
            description: "Liste des organisateurs"
          },
          participants: {
            bsonType: "array",
            items: {
```

```
            bsonType: "objectId"
          },
          description: "Liste des participants"
        }
      },
      additionalProperties: false
    }
  },
  validationAction: "error",
  validationLevel: "strict"
 }
);
```

**Lire les évènements :**

- **Opération Read**
- **Méthode 'GET'**
- **Route '/events/'**

```
db.events.find({});
```

**Mettre à jour un évènement :**

- **Opération Update**
- **Méthode 'PUT'**
- **Route '/events/update/<id>'**

```
db.events.updateOne(
  { _id: ObjectId("<id>") },
  {
   $set: {
     name: "<nouvelle valeur>",
     description: "<nouvelle valeur>",
     startdate: new  Date("<nouvelle valeur>"),
     enddate: new  Date("<nouvelle valeur>"),
     place: "<nouvelle valeur>",
     "organizers.<?>": "<nouvelle valeur>",
     "participants.<?>": "<nouvelle valeur>"
   }
  }
);
```

**Supprimer un évènement :**

- **Opération Delete**
- **Méthode 'DELETE'**
- **Route '/events/delete/<id>'**

```
db.events.deleteOne({ _id: ObjectId("<id>") });
```

**Rechercher un évènement :**

- **Opération Search**
- **Méthode 'GET'**
- **Route '/events/<id>'**

```
db.events.find({ _id: ObjectId("<id>") });
```

**Exemple d'insertion d'un évènement :**

```
db.events.insertOne(
  {
    name: "Gala de fin d'année",
    description: "Gala de fin d'année 2023-2024 pour toute les promotions",
    startdate: new Date("2024-06-22T18:00"),
    enddate: new Date("2024-06-23T05:30"),
    place: "Efrei Paris",
    organizers: [ ObjectId("655e7973b09c637100ad482e") ],
    participants: [
      ObjectId("655e7973b09c637100ad482e"),
      ObjectId("655e7571b09c637100ad482b")
   ]
  }
);
```

## Groupes :

**Créer un groupe :**

- **Opération Create**
- **Méthode 'POST'**
- **Route '/groups/create'**

```
db.createCollection("groups",
  {
    validator: {
      $jsonSchema: {
        bsonType: "object",
        required: ["name", "description", "type", "admins"],
        properties: {
          _id: {
            bsonType: "objectId"
          },
          name: {
            bsonType: "string",
            description: "Nom du groupe"
          },
          description: {
            bsonType: "string",
            description: "Description du groupe"
          },
          type: {
            bsonType: "string",
            enum: ["private", "public", "secret"],
            description: "Type de groupe (public, privé, secret)"
          },
          admins: {
            bsonType: "array",
            items: {
              bsonType: "objectId"
            },
            description: "Liste des administrateurs"
          },
          members: {
            bsonType: "array",
            items: {
              bsonType: "ObjectId"
            },
            description: "Liste des membres"
          }
        },
        additionalProperties: false
      }
    },
```

```
  validationAction: "error",
  validationLevel: "strict"
  }
);
```

**Lire les groupes :**

- **Opération Read**
- **Méthode 'GET'**
- **Route '/groups/'**

```
db.groups.find({});
```

**Mettre à jour un groupe :**

- **Opération Update**
- **Méthode 'PUT'**
- **Route '/groups/update/<id>'**

```
db.groups.updateOne(
  { _id: ObjectId("<id>") },
  {
    $set: {
      name: "<nouvelle valeur>",
      description: "<nouvelle valeur>",
      type: "<nouvelle valeur>",
      "admins.<?>": "<nouvelle valeur>",
      "members.<?>": "<nouvelle valeur>"
    }
  }
);
```

**Supprimer un groupe :**

- **Opération Delete**
- **Méthode 'DELETE'**
- **Route '/groups/delete/<id>'**

```
db.groups.deleteOne({ _id: ObjectId("<id>") });
```

**Rechercher un groupe :**

- **Opération Search**
- **Méthode 'GET'**
- **Route '/groups/<id>'**

```
db.groups.find({ _id: ObjectId("<id>") });
```

**Exemple d'insertion d'un groupe :**

```
db.groups.insertOne(
  {
    name: "M2 Data Engineering",
    description: "Groupe de classe",
    type: "private",
    admins: [ ObjectId("655e7973b09c637100ad482e") ],
    members: [
      ObjectId("655e7973b09c637100ad482e"),
      ObjectId("655e7571b09c637100ad482b"),
      ObjectId("655e7571b09c637100ad482c"),
      ObjectId("655e7571b09c637100ad482t")
    ]
  }
);
```

## Fils de discussions :

**Créer un fil de discussion :**

- **Opération Create**
- **Méthode 'POST'**
- **Route '/discussionThreads/create'**

```
db.createCollection("discussionThreads",
  {
    validator: {
      $jsonSchema: {
        bsonType: "object",
        required: ["linked", "messages"],
        properties: {
          _id: {
            bsonType: "ObjectId"
          },
          linked: {
            bsonType: "object",
            properties: {
              id: { bsonType: "objectId" },
              connected: { bsonType: "string", enum: ["event", "group"] }
            },
            additionalProperties: false,
            description: "Lien vers un événement ou groupe"
          },
          messages: {
            bsonType: "array",
            items: {
              bsonType: "object",
              properties: {
                author: { bsonType: "objectId" },
                content: { bsonType: "string" },
                comments: {
                  bsonType: "array",
                  items: {
                    bsonType: "object",
                    properties: {
                      author: { bsonType: "objectId" },
                      content: { bsonType: "string" }
                    },
                    additionalProperties: false
                  }
                }
              },
              additionalProperties: false,
              description: "Liste des messages du fil de discussion"
            }
```

```
          }
        },
        additionalProperties: false
      }
    },
    validationAction: "error",
    validationLevel: "strict"
  }
);
```

**Lire les fils de discussions :**

- **Opération Read**
- **Méthode 'GET'**
- **Route '/discussionThreads/'**

```
db.discussionThreads.find({});
```

**Mettre à jour un fil de discussion :**

- **Opération Update**
- **Méthode 'PUT'**
- **Route '/discussionThreads/update/<id>'**

```
db.discussionThreads.updateOne(
  { _id: ObjectId("<id>") },
  {
    $set: {
      "linked.id": ObjectId("<nouvelle valeur>"),
      "linked.connected": "<nouvelle valeur>",
      "messages.<?>.author": "<nouvelle valeur>",
      "messages.<?>.content": "<nouvelle valeur>",
      "messages.<?>.comments.<?>.author": "<nouvelle valeur>"
      "messages.<?>.comments.<?>.content": "<nouvelle valeur>"
    }
  }
);
```

**Supprimer un fil de discussion :**

- **Opération Delete**
- **Méthode 'DELETE'**
- **Route '/discussionThreads/delete/<id>'**

```
db.discussionThreads.deleteOne({ _id: ObjectId("<id>") });
```

**Rechercher un fil de discussion :**

- **Opération Search**
- **Méthode 'GET'**
- **Route '/discussionThreads/<id>'**

```
db.discussionThreads.find({ _id: ObjectId("<id>") });
```

**Exemple d'insertion d'un fil de discussion :**

```
db.discussionThreads.insertOne({
  linked: {
    id: ObjectId("655e8661b09c637100ad4838"),
    connected: "group"
  },

  messages: [
    {
      author: ObjectId("655e7973b09c637100ad482e"),
      content: "Quand est le rendu du projet de DataBase NoSQL <?>",
      comments: [
        {
          author: ObjectId("655e7571b09c637100ad482b"),
          content: "Jeudi prochain"
        },
        {
          author: ObjectId("655e7571b09c637100ad482c"),
          content: "Non, c'est le mardi 28 novembre ! ;)"
        }
      ]
    }
  ]
});
```

## Albums photo :

**Créer un album photo :**

- **Opération Create**
- **Méthode 'POST'**
- **Route '/photoAlbums/create'**

```
db.createCollection("photoAlbums",
  {
    validator: {
      $jsonSchema: {
        bsonType: "object",
        required: ["event", "photos"],
        properties: {
          _id: {
            bsonType: "objectId"
          },
          event: {
            bsonType: "objectId",
            description: "Événement lié à l'album photo"
          },
          photos: {
            bsonType: "array",
            items: {
              bsonType: "object",
              properties: {
                author: { bsonType: "objectId" },
                imageUrl: { bsonType: "string" },
                comments: {
                  bsonType: "array",
                  items: {
                    bsonType: "object",
                    properties: {
                      author: { bsonType: "objectId" },
                      content: { bsonType: "string" }
                    },
                    additionalProperties: false
                  }
                },
                additionalProperties: false
              },
              description: "Liste des photos de l'album"
            }
          },
          additionalProperties: false
        }
      },
```

```
    validationAction: "error",
    validationLevel: "strict"
  }
);
```

**Lire les albums photo :**

- **Opération Read**
- **Méthode 'GET'**
- **Route '/photoAlbums/'**

```
db.photoAlbums.find({});
```

**Mettre à jour un album photo :**

- **Opération Update**
- **Méthode 'PUT'**
- **Route '/photoAlbums/update/<id>'**

```
db.photoAlbums.updateOne(
  { _id: ObjectId("<id>") },
  {
    $set: {
      event: ObjectId("<nouvelle valeur>"),
      "photos.<?>.author": ObjectId("<nouvelle valeur>"),
      "photos.<?>.imageUrl": "<nouvelle valeur>",
      "photos.<?>.comments.<?>.author": ObjectId("<nouvelle
   valeur>"),
      "photos.<?>.comments.<?>.content": "<nouvelle valeur>"
    }
  }
);
```

**Supprimer un album photo :**

- **Opération Delete**
- **Méthode 'DELETE'**
- **Route '/photoAlbums/delete/<id>'**

```
db.photoAlbums.deleteOne({ _id: ObjectId("<id>") });
```

**Rechercher un album photo :**

- **Opération Search**
- **Méthode 'GET'**
- **Route '/photoAlbums/<id>'**

```
db.photoAlbums.find({ _id: ObjectId("<id>") });
```

**Exemple d'insertion d'un album photo :**

```
db.photoAlbums.insertOne(
  {
    event: ObjectId("655e80beb09c637100ad4835"),
    photos: [
      {
        author: ObjectId("655e7571b09c637100ad482b"),
        imageUrl: "https://www.google.com/couche_soleil",
        comments: [
          {
            author: ObjectId("655e7973b09c637100ad482e"),
            content: "Trop beau ce couché de soleil! :)"
          },
          {
            author: ObjectId("655e7973b09c637100ad482t"),
            content: "C'est où <?>"
          }
        ]
      }
    ]
  }
);
```

## Sondages :

**Créer un sondage :**

- **Opération Create**
- **Méthode 'POST'**
- **Route '/surveys/create'**

```
db.createCollection("surveys",
  {
    validator: {
      $jsonSchema: {
        bsonType: "object",
        required: ["eventId", "questions"],
        properties: {
          _id: {
            bsonType: "objectId"
          },
          eventId: {
            bsonType: "objectId",
            description: "ID de l'événement associé au sondage"
          },
          questions: {
            bsonType: "array",
            items: {
              bsonType: "object",
              required: ["questionText", "options"],
              properties: {
                questionText: {
                  bsonType: "string",
                  description: "Texte de la question"
                },
                options: {
                  bsonType: "array",
                  items: {
                    bsonType: "object",
                    required: ["optionText"],
                    properties: {
                      optionText: {
                        bsonType: "string",
                        description: "Texte de l'option"
                      },
                      respondents: {
                        bsonType: "array",
                        items: {
                          bsonType: "objectId",
                          additionalProperties: false
                        }
                      }
```

```
                    },
                    additionalProperties: false
                  },
                  uniqueItems: true,
                  description: "Liste des options de réponse pour la question"
                }
              },
              additionalProperties: false
            },
            uniqueItems: true,
            description: "Liste des questions du sondage"
          }
        },
        additionalProperties: false
      }
    },
    validationAction: "error",
    validationLevel: "strict"
  }
);
```

**Lire les sondages :**

- **Opération Read**
- **Méthode 'GET'**
- **Route '/surveys/'**

```
db.surveys.find({});
```

**Mettre à jour un sondage :**

- **Opération Update**
- **Méthode 'PUT'**
- **Route '/surveys/update/<id>'**

```
db.surveys.updateOne(
  { _id: ObjectId("<id>") },
  {
    $set: {
      eventId: ObjectId("<nouvelle valeur>"),
      "questions.<?>.questionText": "<nouvelle valeur>",
      "questions.<?>.options.<?>.optionText": "<nouvelle valeur>",
      "questions.<?>.options.<?>.respondents.<?>": ObjectId("<nouvelle valeur>")
    }
  }
);
```

**Supprimer un sondage :**

- **Opération Delete**
- **Méthode 'DELETE'**
- **Route '/surveys/delete/<id>'**

```
db.surveys.deleteOne({ _id: ObjectId("<id>") });
```

**Rechercher un sondage :**

- **Opération Search**
- **Méthode 'GET'**
- **Route '/surveys/<id>'**

```
db.surveys.find({ _id: ObjectId("<id>") });
```

**Exemple d'insertion d'un sondage :**

```
db.surveys.insertOne(
  {
    eventId: ObjectId("655e80beb09c637100ad4835"),
    questions : [
      {
        questionText: "Ou voulez-vous manger <?>",
        options: [
          {
            optionText: "Au mc Donald",
            respondents: [ ObjectId("655e7571b09c637100ad482b") ]
          },
          {
            optionText: "Au Burger King",
            respondents: []
          }
        ]
      },

      {
        questionText: "Ou voulez-vous partir en vacances <?>",
        options: [
          {
            optionText: "En Italie",
            respondents: [ ObjectId("655e7571b09c637100ad482b") ]
```

```
        },
        {
          optionText: "En Espagne",
          respondents: [
            ObjectId("655e7973b09c637100ad482e"),
            ObjectId("655e7973b09c637100ad482c")
          ]
        },
        {
          optionText: "En Allemagne",
          respondents: []
        }
      ]
    }
  ]
}
);
```

## Billetterie :

**Créer une billetterie :**

- **Opération Create**
- **Méthode 'POST'**
- **Route '/ticketings/create'**

```
db.createCollection("ticketings",
  {
    validator: {
      $jsonSchema: {
        bsonType: "object",
        required: ["event", "tickets"],
        properties: {
          _id: {
            bsonType: "objectId"
          },
          event: {
            bsonType: "objectId",
            description: "Billetterie liée à la l'événement"
          },
          tickets: {
            bsonType: "array",
            items: {
              bsonType: "object",
              properties: {
                ticketType: { bsonType: "string" },
                amount: { bsonType: "double" },
                quantity: { bsonType: "int" }
              },
              additionalProperties: false
            },
            description: "Liste des types de billets disponibles"
          },
          ticketsSold: {
            bsonType: "array",
            items: {
              bsonType: "object",
              properties: {
                ticketType: { bsonType: "string" },
                participant: { bsonType: "objectId" },
                purchaseDate: { bsonType: "date" }
              },
              additionalProperties: false
            },
            description: "Liste des billets vendus"
          }
        },
```

```
        additionalProperties: false
      }
    },
    validationAction: "error",
    validationLevel: "strict"
  }
);
```

**Lire les billetteries :**

- **Opération Read**
- **Méthode 'GET'**
- **Route '/ticketings/'**

```
db.ticketings.find({});
```

**Mettre à jour une billetterie :**

- **Opération Update**
- **Méthode 'PUT'**
- **Route '/ticketings/update/<id>'**

```
db.ticketings.updateOne(
  { _id: ObjectId("<id>") },
  {
    $set: {
      event: ObjectId("<nouvelle valeur>"),
      "tickets.<?>.ticketType": "<nouvelle valeur>",
      "tickets.<?>.amount": <nouvelle valeur>,
      "tickets.<?>.quantity": <nouvelle valeur>,
      "ticketsSold.<?>.ticketType": "<nouvelle valeur>",
      "ticketsSold.<?>.participant": ObjectId("<nouvelle valeur>"),
      "ticketsSold.<?>.purchaseDate": new  Date ("<nouvelle
   valeur>")
    }
  }
);
```

**Supprimer une billetterie :**

- **Opération Delete**
- **Méthode 'DELETE'**
- **Route '/ticketings/delete/<id>'**

```
db.ticketings.deleteOne({ _id: ObjectId("<id>") }) ;
```

**Rechercher une billetterie :**

- **Opération Search**
- **Méthode 'GET'**
- **Route '/ticketings/<id>'**

```
db.ticketings.find({ _id: ObjectId("<id>") });
```

**Exemple d'insertion d'une billetterie :**

```
db.ticketings.insertOne(
  {
    event: ObjectId("655e80beb09c637100ad4835"),
    tickets: [
      {
        ticketType: "Classique",
        amount: 59.99,
        quantity: 100
      },
      {
        ticketType: "VIP",
        amount: 199.99,
        quantity: 20
      }
    ],
    ticketsSold: [
      {
        ticketType: "VIP",
        participant: ObjectId("655e7571b09c637100ad482b"),
        purchaseDate: new Date("2023-11-22T18:00")
      },
      {
        ticketType: "Classique",
```

```
        participant: ObjectId("655e7973b09c637100ad482e"),
        purchaseDate: new Date("2023-11-20T14:30")
      },
      {
        ticketType: "Classique",
        participant: ObjectId("655e7973b09c637100ad482c"),
        purchaseDate: new Date("2023-11-20T14:30")
      }
    ]
  }
);
```

## Listes de course :

Créer une liste de course :

- Opération Create
- Méthode 'POST'
- Route '/shoppingList/create'

```
db.createCollection("shoppingList",
  {
    validator: {
      $jsonSchema: {
        bsonType: "object",
        required: ["event", "items"],
        properties: {
          _id: {
            bsonType: "objectId"
          },
          event: {
            bsonType: "objectId",
            description: "Événement lié à la liste de courses"
          },
          items: {
            bsonType: "array",
            items: {
              bsonType: "object",
              properties: {
                name: { bsonType: "string" },
                quantity: { bsonType: "int" },
                arrivalTime: { bsonType: "date" }
              }
            },
            description: "Liste des éléments pour la liste de courses"
          }
        },
        additionalProperties: false
      }
    },
    validationAction: "error",
    validationLevel: "strict"
  }
);


db.shoppingList.createIndex({"items.name": 1 }, { unique: true });
```

**Lire les listes de course :**

- **Opération Read**
- **Méthode 'GET'**
- **Route '/shoppingList/'**

```
db.shoppingList.find({});
```

**Mettre à jour une liste de course :**

- **Opération Update**
- **Méthode 'PUT'**
- **Route '/shoppingList/update/<id>'**

```
db.shoppingList.updateOne(
  { _id: ObjectId("<id>") },
  {
    $set: {
      event: ObjectId("<nouvelle valeur>"),
      "items.<?>.name": "<nouvelle valeur>",
      "items.<?>.quantity": <nouvelle valeur>,
      "items.<?>.arrivalTime": new Date("<nouvelle valeur>")
    }
  }
);
```

**Supprimer une liste de course :**

- **Opération Delete**
- **Méthode 'DELETE'**
- **Route '/shoppingList/delete/<id>'**

```
db.shoppingList.deleteOne({ _id: ObjectId("<id>") }) ;
```

**Rechercher une liste de course :**

- **Opération Search**
- **Méthode 'GET'**
- **Route '/shoppingList/<id>'**

```
db.shoppingList.find({ _id: ObjectId("<id>") });
```

**Exemple d'insertion d'une liste de course :**

```
db.shoppingList.insertOne(
  {
    event: ObjectId("655e80beb09c637100ad4835"),
    items: [
      {
        name: "Produit 1",
        quantity: 2,
        arrivalTime: new Date("2023-12-01T10:00:00Z")
      },
      {
        name: "Produit 2",
        quantity: 1,
        arrivalTime: new Date("2023-12-01T10:30:00Z")
      }
    ]
  }
);
```

## Covoiturage :

**Créer un covoiturage :**

- **Opération Create**
- **Méthode 'POST'**
- **Route '/carshares/create'**

```
db.createCollection("carshares",
  {
    validator: {
      $jsonSchema: {
        bsonType: "object",
        required: ["event", "cars"],
        properties: {
          _id: {
            bsonType: "objectId"
          },
          event: {
            bsonType: "objectId",
            description: "Événement lié au covoiturage"
          },
          cars: {
            bsonType: "array",
            items: {
              bsonType: "object",
              properties: {
                departurePlace: { bsonType: "string" },
                departureTime: { bsonType: "date" },
                price: { bsonType: "double" },
                availableSeats: { bsonType: "int" },
                maxGapTime: { bsonType: "int" }
              }
            },
            description: "Liste des trajets disponibles pour le covoiturage"
          }
        },
        additionalProperties: false
      }
    },
    validationAction: "error",
    validationLevel: "strict"
  }
);
```

**Lire les covoiturages :**

- **Opération Read**
- **Méthode 'GET'**
- **Route '/carshares/'**

```
db.carshares.find({});
```

**Mettre à jour un covoiturage :**

- **Opération Update**
- **Méthode 'PUT'**
- **Route '/carshares/update/<id>'**

```
db.carshares.updateOne(
  { _id: ObjectId("<id>") },
  {
    $set: {
      event: ObjectId("<nouvelle valeur>"),
      "cars.<?>.departurePlace": "<nouvelle valeur>",
      "cars.<?>.departureTime": new Date("<nouvelle valeur>"),
      "cars.<?>.price": <nouvelle valeur>,
      "cars.<?>.availableSeats": <nouvelle valeur>,
      "cars.<?>.maxGapTime": <nouvelle valeur>
    }
  }
);
```

**Supprimer un covoiturage :**

- **Opération Delete**
- **Méthode 'DELETE'**
- **Route '/carshares/delete/<id>'**

```
db.carshares.deleteOne({ _id: ObjectId("<id>") }) ;
```

**Rechercher un covoiturage :**

- **Opération Search**
- **Méthode 'GET'**
- **Route '/carshares/<id>'**

```
db.carshares.find({ _id: ObjectId("<id>") });
```

**Exemple d'insertion d'un covoiturage :**

```
db.carshares.insertOne(
  {
    event: ObjectId("655e80beb09c637100ad4835"),
    cars: [
      {
        departurePlace: "Adresse de départ 1",
        departureTime: new Date("2023-12-01T08:00:00Z"),
        price: 25.50,
        availableSeats: 3,
        maxGapTime: 30
      },
      {
        departurePlace: "Adresse de départ 2",
        departureTime: new Date("2023-12-01T10:00:00Z"),
        price: 19.90,
        availableSeats: 1,
        maxGapTime: 60
      }
    ]
  }
);
```