Connexion base de données

```
// connection to DataBase
mongoose.connect(`mongodb://${process.env.DB_USER}:${process.env.DB_PAS
SWORD}@${process.env.DB_HOST}:27017/${process.env.DB}`, {

    useNewUrlParser: true,
    useUnifiedTopology: true,
  })
  .then(() => console.log('Connexion à MongoDB réussie !'))
  .catch(() => console.log('Connexion à MongoDB échouée !'));
```

configuration router

```
app.use('/utilisateur', userRouter);
```

configuration route et contrôleur :

```
// Sign Up route : creates a new user
router.post('/inscription', UserController.user_signup);

// User login route
router.post('/login', UserController.user_login);
```

schéma mongoose :

```
const userSchema = mongoose.Schema({
  // _id: mongoose.Schema.Types.ObjectId,
  email: {
    type: String,
    required: true,
    unique: true,
    // eslint-disable-next-line no-useless-escape
    match:
/[a-z0-9!#$%&'*+\=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+\=?^_`{|}~-]+)*@(?:[a-
z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?/,

  },
  password: { type: String, required: true },
  lastname: { type: String, required: true },
  firstname: { type: String, required: true },
  phone_number: { type: String, required: true },
  address: { type: String },
  postal_code: { type: String },
  city: { type: String },
  role: { type: String, required: ['admin', 'client', 'employé'],
default: 'client' },
  resetLinkToken: { type: String, default: ' ' },
});
```

Configuration contrôleur signup

```
// Sign Up route : creates a new user
exports.user_signup = (req, res) => {
  // Checking if email already exists
  const { email } = req.body;
  User.find({ email })
    .exec()
    .then((user) => {
      if (user.length >= 1) {
        return res.status(409).json({ message: 'Cet email existe déjà'
});}
      bcrypt.hash(req.body.password, 10, (err, hash) => {
        if (err) {
          return res.status(500).json({ error: err });
        }
        const newUser = new User({
          _id: new mongoose.Types.ObjectId(),
          email: req.body.email,
          password: hash,
          lastname: req.body.lastname,
          firstname: req.body.firstname,
          phone_number: req.body.phone_number,
          address: req.body.adsress,
          postal_code: req.body.postal_code,
          city: req.body.city,
          role: req.body.role,
        });
        newUser
          .save()
          .then(() => {
            res.status(201).json({ message: 'Compte créé avec succès'
});
          })
          .catch((error) => {
            res.status(500).json({ message: error });
          });
      });
    });
};
```

Configuration contrôleur login

```
// User Login
exports.user_login = (req, res) => {
  const { email } = req.body;
  User.find({ email })
    .exec()
    .then((user) => {
      if (user.length < 1) {
        return res.status(401).json({ message: 'Echec connexion' });
      }

      bcrypt.compare(req.body.password, user[0].password, (err, result)
=> {
        if (err) {
          return res.status(401).json({ message: 'Echec connexion' });
        }
        if (result) {
          const token = jwt.sign(
            { userId: user[0]._id, role: user[0].role },
            process.env.JWT_PASSWORD,
            { expiresIn: '1h' },
          );

          return res.status(200).json({ message: 'Connexion réussie',
token });
        }
        res.status(401).json({ message: 'Echec connexion' });
      });
    })
    .catch((err) => {
      res.status(500).json({ error: err });
    });
};
```