
فهاد الجزء الثاني من Mern Social Network من بعد ما زدنا فالجزء السابق ل models ديالنا دبا غادي ندوزوا ل controllers لي غادي يكونوا فيهم les fonctions لي غادي نحتاجوا ولي غادي يمكنوا المستخدمين باش يتسجلوا و يزيدوا les tweets ديالهم.

1- إضافة ل user controller

دائما ف dossier backend زيد dossier جديد سميه controllers فيه غادي نزيدو fichier user.js هنا غادي يكونوا عندي des fonctions لي غادي يمكنوا باش نزيد مستخدم جديد فقاعدة البيانات ونسترجع مستخدم بل id ديالو ونسترجع مستخدم ونسترجع جميع المستخدمين وندير تعديل على مستخدم ونحذف مستخدم ونسترجع صورة مستخدم ولهاد الغرض غادي تزيد dossier images ف dossier backend وفيه زيد صورة ديال مستخدم ممكن تطيليشارجيها من الأنترنيت وسميها user.png باش يلا مكانش عند المستخدم image غادي تكون هي الصورة par défaut.

وعندي أيضا les fonctions لي غادي يمكنوا المستخدم باش يتابع أو يلغي متابعة مستخدم آخر.

الكود لي غادي تزيد هو :

```
                                        const User =
require("../models/user");
const _ = require("lodash");
const formidable = require("formidable");
const fs = require("fs");

const createUser = (req, res) => {
  const { name, email, password } = req.body;
  const user = new User({ name, email, password });
  user.save((err, user) => {
    if (err) return res.json({ error: err });
    user.hashed_password = undefined;
    user.salt = undefined;
    res.json(user);
  });
};

const getUserById = (req, res, next, id) => {
  User.findById(id)
    .populate("following", "_id name")
    .populate("followers", "_id name")
    .exec((err, user) => {
```

```
      if (err || !user) return res.json({ error: "Aucun profile trouvé" });
      req.profile = user;
      next();
    });
};

const getUser = (req, res) => {
  req.profile.hashed_password = undefined;
  req.profile.salt = undefined;
  res.json(req.profile);
};

const getAllUsers = (req, res) => {
  User.find((err, users) => {
    if (err || !users) return res.json({ error: err });
    res.json(users);
  }).select("name email about image createdAt");
};

const updateUser = (req, res) => {
  console.log(req.body);
  let form = new formidable.IncomingForm();
  form.keepExtensions = true;
  form.parse(req, (err, fields, files) => {
    if (err)
      return res.json({ error: "Impossible d'ajouter le fichier séléctionné"
});
    let user = req.profile;
    user = _.extend(user, fields);
    if (files.image) {
      user.image.data = fs.readFileSync(files.image.path);
      user.image.contentType = files.image.type;
    }
    user.save((err, result) => {
      if (err) return res.json({ error: err });
      result.hashed_password = undefined;
      result.salt = undefined;
      result.image = undefined;
      res.json(result);
    });
  });
};

const deleteUser = (req, res) => {
  let user = req.profile;
  user.remove((err, deletedUser) => {
    if (err) res.json({ error: err });
    res.json({ message: "Compte supprimé" });
  });
};
```

```
const getUserPhoto = (req, res) => {
  if (req.profile.image.data) {
    res.set("Content-Type", req.profile.image.contentType);
    return res.send(req.profile.image.data);
  } else {
    return res.sendFile(
      "c:/xampp/htdocs/react-social-network-channel/backend/images/user.png"
    );
  }
};

const addFollowing = (req, res, next) => {
  User.findByIdAndUpdate(
    req.body.userId,
    { $push: { following: req.body.followId } },
    (err, result) => {
      if (err) return res.json({ error: err });
      next();
    }
  );
};

const addFollower = (req, res) => {
  User.findByIdAndUpdate(
    req.body.followId,
    { $push: { followers: req.body.userId } },
    { new: true }
  )
    .populate("following", "_id name ")
    .populate("followers", "_id name ")
    .exec((err, result) => {
      if (err) return res.json({ error: err });
      result.hashed_password = undefined;
      result.salt = undefined;
      result.image = undefined;
      res.json(result);
    });
};

const removeFollowing = (req, res, next) => {
  User.findByIdAndUpdate(
    req.body.userId,
    { $pull: { following: req.body.followId } },
    (err, result) => {
      if (err) return res.json({ error: err });
      next();
    }
  );
};

const removeFollower = (req, res) => {
```

```
  User.findByIdAndUpdate(
    req.body.followId,
    { $pull: { followers: req.body.userId } },
    { new: true }
  )
    .populate("following", "_id name ")
    .populate("followers", "_id name ")
    .exec((err, result) => {
      if (err) return res.json({ error: err });
      result.hashed_password = undefined;
      result.salt = undefined;
      result.image = undefined;
      res.json(result);
    });
};

module.exports = {
  createUser,
  getUserById,
  getUser,
  getAllUsers,
  updateUser,
  deleteUser,
  getUserPhoto,
  addFollowing,
  addFollower,
  removeFollowing,
  removeFollower,
};
```

2- إضافة ل auth controller

دائما ف dossier controllers زيد fichier جديد سميه auth.js فيه غادي نزيدو des fonctions لي غادي يمكنوا

المستخدم باش يتكونيكطا ويديكونيكطا أيضا غادي نزيدو des middlewares لي هما عبارة على des fonctions لي كيتحققوا من واحد الشرط قبل ما يخليو المستخدم يدير واحد ل action.

هنا عندنا requireSignin لي كتأكد بلي المستخدم مكونيكطي الدور ديالها ميمكن للمستخدم يزيد شي حاجة مثلا tweet حتى يكون مكونيكطي وكنخدم بديك JWT_SECRET لي سبق وزدنا ف env.

وعندنا أيضا hasAuthorization لي كتحقق واش المستخدم لي مكونيكطي هو لي عندو الحق يقوم بواحد العملية مثلا يعدل ل profile ديالو.

الكود لي غادي تزيد هو :

```
                                                const User =
require("../models/user");
const jwt = require("jsonwebtoken");
const expressJwt = require("express-jwt");
require("dotenv").config();

const signin = (req, res) => {
  User.findOne({ email: req.body.email }, (err, user) => {
    if (err || !user) {
      return res.json({ error: "Aucune donnée trouvée" });
    }
    user.comparePassword(req.body.password, function (err, isMatch) {
      if (!isMatch) {
        return res.json({ error: "Email ou mot de passe est incorrect" });
      }
      const token = jwt.sign({ _id: user._id }, process.env.JWT_SECRET);
      res.cookie("t", token, {
        expire: new Date() + 9999,
      });
      user.hashed_password = undefined;
      user.salt = undefined;
      return res.json({
        token,
        user,
      });
    });
  });
};

const signout = (req, res) => {
  res.clearCookie("t");
  res.json({ message: "Déconnecté" });
};

const requireSignin = expressJwt({
  secret: process.env.JWT_SECRET,
  userProperty: "auth",
  algorithms: ["HS256"],
});

const hasAuthorization = (req, res, next) => {
  const authorized = req.profile && req.auth && req.profile._id ==
req.auth._id;
  if (!authorized) {
    return res.json({
      error: "Non authorisé",
    });
  }
  next();
};
```

```
module.exports = {
  signin,
  signout,
  hasAuthorization,
  requireSignin,
};
```

3- إضافة ل post controller

دائما ف dossier controllers زيد fichier جديد سميه post.js فيه غادي نزيدو des fonctions لي غادي يمكنوا

المستخدم باش يزيد tweet يسترجع tweet بل id ديالها يحذف tweet وأيضا يزيد أو يحذف تعليق.

ايضا عندنا middleware سميناها isOwner لي الدور ديالها كتحقق واش المستخدم هو مول tweet قبل ما تحذف وعندي أيضا les fonctions لي كيمكنوا من إسترجاع les tweets كاملين و les tweets الخاصين بمستخدم معين.

الكود لي غادي تزيد هو :

```
                                        const Post =
require("../models/post");

const getAllPosts = (req, res) => {
  let following = req.profile.following;
  following.push(req.profile._id);
  Post.find({ postedBy: { $in: req.profile.following } })
    .populate("comments", "text created")
    .populate("comments.postedBy", "_id name")
    .populate("postedBy", "_id name")
    .sort("-createdAt")
    .exec((err, posts) => {
      if (err) res.json({ error: err });
      res.json(posts);
    });
};

const userPosts = (req, res) => {
  Post.find({ postedBy: req.profile._id })
    .populate("comments", "text created")
    .populate("comments.postedBy", "_id name")
    .populate("postedBy", "_id name")
    .sort("-createdAt")
    .exec((err, posts) => {
      if (err) res.json({ error: err });
      res.json(posts);
    });
```

```
};

const getPostById = (req, res, next, id) => {
  Post.findById(id)
    .populate("comments", "text created")
    .populate("comments.postedBy", "_id name")
    .populate("postedBy", "_id name")
    .exec((err, post) => {
      if (err) res.json({ error: err });
      req.post = post;
      next();
    });
};

const isOwner = (req, res, next) => {
  let isMine = req.post && req.auth && req.post.postedBy._id == req.auth._id;
  if (!isMine) {
    return res.json({ error: "Non authorisé" });
  }
  next();
};

const addPost = (req, res) => {
  const { text } = req.body;
  let post = new Post({ text, postedBy: req.profile._id });
  post.save((err, data) => {
    if (err) res.json({ error: err });
    res.json(data);
  });
};

const deletePost = (req, res) => {
  let postToDelete = req.post;
  postToDelete.remove((err, deletedPost) => {
    if (err) res.json({ error: err });
    res.json(deletedPost);
  });
};

const likePost = (req, res) => {
  Post.findByIdAndUpdate(
    req.body.postId,
    { $push: { likes: req.body.userId } },
    { new: true }
  ).exec((err, result) => {
    if (err) res.json({ error: err });
    res.json(result);
  });
};

const unlikePost = (req, res) => {
```

```
  Post.findByIdAndUpdate(
    req.body.postId,
    { $pull: { likes: req.body.userId } },
    { new: true }
  ).exec((err, result) => {
    if (err) res.json({ error: err });
    res.json(result);
  });
};

const addComment = (req, res) => {
  let comment = { text: req.body.text };
  comment.postedBy = req.body.userId;
  Post.findByIdAndUpdate(
    req.body.postId,
    { $push: { comments: comment } },
    { new: true }
  ).exec((err, result) => {
    if (err) res.json({ error: err });
    res.json(result);
  });
};

const deleteComment = (req, res) => {
  let comment = req.body.comment;
  Post.findByIdAndUpdate(
    req.body.postId,
    {
      $pull: {
        comments: {
          _id: comment._id,
        },
      },
    },
    { new: true }
  ).exec((err, result) => {
    if (err) res.json({ error: err });
    res.json(result);
  });
};

module.exports = {
  getAllPosts,
  addPost,
  userPosts,
  getPostById,
  isOwner,
  deletePost,
  likePost,
  unlikePost,
  addComment,
```

```
  deleteComment,
};
```