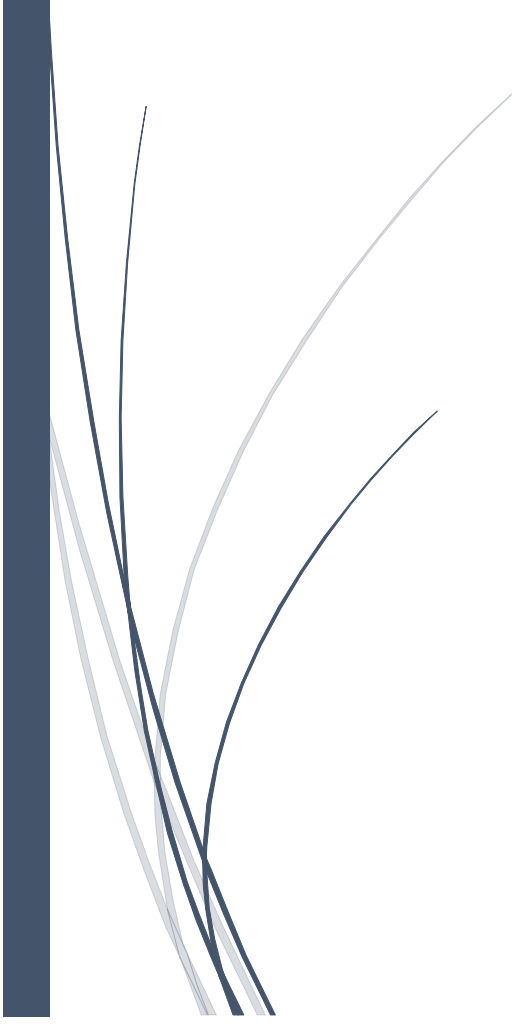
B3

PROJET NOSQL

Partage de document



EPSI

Table des matières

Contexte

Le projet se base sur MangoDB avec une api facilitant le partage de fichier via la génération de lien de téléchargement.

Voici le code CSS:

```css
:root {
    --main-bg-color: #edf5fe;
    --light-blue: #03a9f4;
    --dark-blue: #028bca;
  }

  body {
    font-family: system-ui;
    background: var(--main-bg-color);
    height: 98vh;
    overflow: hidden;
    display: flex;
    align-items: center;
    justify-content: center;
    color: #333;
  }

  .logo {
    position: absolute;
    top: 10px;
    left: 10px;
    width: 150px;
  }
section.download {
    background: #fff;
    width: 430px;
    max-width: 90%;
    border-radius: 25px;
    box-shadow: 0px 20px 20px 0px #00000017;
    padding: 2rem;
    text-align: center;
}
.download__icon {
    height: 8rem;
}
.download__meta h4 {
    margin-bottom: 0;
    line-height: 1.3;
}
.send-btn-container a {
    display: inline-block;
    font-size: 18px;
    padding: 8px 40px;
    margin-top: 15px;
    background: var(--light-blue);
    text-decoration: none;
    border: none;
    border-radius: 5px;
    color: #fff;
    cursor: pointer;
    transition: all .3s ease-in-out;
  }

  .send-btn-container a:hover {
      background: var(--dark-blue);
  }
```

La maquette du projet :

Configuration de la base

```
require('dotenv').config();
const mongoose = require('mongoose');
function connectDB() {
    // Database connection 😍
    mongoose.connect(process.env.MONGO_CONNECTION_URL, { useNewUrlParser: true, useCreateIndex:true, useUnifiedTopology: true, useFindAndModify : true });
    const connection = mongoose.connection;
    connection.once('open', () => {
        console.log('Database connected 😍😍😍😍');
    }).catch(err => {
        console.log('Connection failed 😕😕😕😕');
    });
}

// mIAY0a6u1ByJsWWZ

module.exports = connectDB;
```

Ce code JavaScript utilise Mongoose pour se connecter à une base de données MongoDB. La fonction connectDB établit la connexion en utilisant l'URL de connexion stockée dans process.env.MONGO_CONNECTION_URL. Lorsque la connexion est réussie, il affiche "**Database connected** " dans la console, sinon "**Connection failed** ". Les données sensibles sont stockées dans un fichier de configuration séparé via dotenv.

Configuration les enregistrements :

```
const connectDB = require('./config/db');
const File = require('./models/file');
const fs = require('fs');

connectDB();

// Get all records older than 24 hours
async function fetchData() {
    const files = await File.find({ createdAt : { $lt: new Date(Date.now() - 24 * 60 * 60 * 1000)} })
    if(files.length) {
        for (const file of files) {
            try {
                fs.unlinkSync(file.path);
                await file.remove();
                console.log(`successfully deleted ${file.filename}`);
            } catch(err) {
                console.log(`error while deleting file ${err} `);
            }
        }
    }
    console.log('Job done!');
}

fetchData().then(process.exit);
```

Ce script se connecte à une base de données MongoDB à l'aide de Mongoose, recherche les enregistrements de fichiers créés il y a plus de 24 heures, supprime ces fichiers du système de fichiers et les documents associés de la base de données, puis affiche les résultats dans la console.

Configuration la gestion des fichiers

```
require('dotenv').config();
const express = require('express');
const app = express();
const PORT = process.env.PORT || 3000;
const path = require('path');
const cors = require('cors');
// Cors
const corsOptions = {
  origin: process.env.ALLOWED_CLIENTS.split(',')
  // ['http://localhost:3000', 'http://localhost:5000', 'http://localhost:3300']
}

// Default configuration looks like
// {
//     "origin": "*",
//     "methods": "GET,HEAD,PUT,PATCH,POST,DELETE",
//     "preflightContinue": false,
//     "optionsSuccessStatus": 204
//   }

app.use(cors(corsOptions))
app.use(express.static('public'));

const connectDB = require('./config/db');
connectDB();

app.use(express.json());

app.set('views', path.join(__dirname, '/views'));
app.set('view engine', 'ejs');

// Routes
app.use('/api/files', require('./routes/files'));
app.use('/files', require('./routes/show'));
app.use('/files/download', require('./routes/download'));


app.listen(PORT, console.log(`Listening on port ${PORT}.`));
```

Ce code est un serveur web écrit en Node.js avec Express.js, qui gère les opérations liées à la gestion de fichiers. Ce serveur web permet aux clients d'effectuer des opérations liées à la gestion de fichiers en utilisant différentes routes. Il est sécurisé grâce à la gestion des autorisations CORS et stocke les informations relatives aux fichiers dans une base de données MongoDB. Le serveur est capable de générer des pages HTML dynamiques à l'aide du moteur de rendu EJS et répond aux requêtes HTTP reçues sur le port spécifié.

Configuration des enregistrement des métadonnées de ces fichiers dans la base de données

```
const router = require('express').Router();
const multer = require('multer');
const path = require('path');
const File = require('../models/file');
const { v4: uuidv4 } = require('uuid');

let storage = multer.diskStorage({
    destination: (req, file, cb) => cb(null, 'uploads/') ,
    filename: (req, file, cb) => {
        const uniqueName = `${Date.now()}-${Math.round(Math.random() * 1E9)}${path.extname(file.originalname)}`;
              cb(null, uniqueName)
    } ,
});

let upload = multer({ storage, limits:{ fileSize: 1000000 * 100 }, }).single('myfile'); //100mb

router.post('/', (req, res) => {
    upload(req, res, async (err) => {
      if (err) {
        return res.status(500).send({ error: err.message });
      }
        const file = new File({
            filename: req.file.filename,
            uuid: uuidv4(),
            path: req.file.path,
            size: req.file.size
        });
        const response = await file.save();
        res.json({ file: `${process.env.APP_BASE_URL}/files/${response.uuid}` });
      });
});

router.post('/send', async (req, res) => {
  const { uuid, emailTo, emailFrom, expiresIn } = req.body;
  if(!uuid || !emailTo || !emailFrom) {
      return res.status(422).send({ error: 'All fields are required except expiry.'});
  }
  // Get data from db
  try {
    const file = await File.findOne({ uuid: uuid });
    if(file.sender) {
      return res.status(422).send({ error: 'Email already sent once.'});
    }
    file.sender = emailFrom;
    file.receiver = emailTo;
    const response = await file.save();
    // send mail
    const sendMail = require('../services/mailService');
    sendMail({
      from: emailFrom,
      to: emailTo,
      subject: 'inShare file sharing',
      text: `${emailFrom} shared a file with you.`,
      html: require('../services/emailTemplate')({
                emailFrom,
                downloadLink: `${process.env.APP_BASE_URL}/files/${file.uuid}?source=email` ,
                size: parseInt(file.size/1000) + ' KB',
                expires: '24 hours'
            })
    }).then(() => {
      return res.json({success: true});
    }).catch(err => {
      return res.status(500).json({error: 'Error in email sending.'});
    });
} catch(err) {
  return res.status(500).send({ error: 'Something went wrong.'});
}

});

module.exports = router;
```

Ce module de routeur pour Express.js permet le téléchargement de fichiers, l'enregistrement des métadonnées de ces fichiers dans la base de données, et l'envoi d'un email contenant un lien de téléchargement vers le fichier. Cela permet de partager des fichiers avec d'autres utilisateurs de manière sécurisée et simple.