# SOUTENANCE DE STAGE

## FW16

Juin 2024




QUENTIN BRANDY

SOUTENANCE DE STAGE EN ENTREPRISE

# PROJET DE STAGE

## Création d’un site d’offre d’emploi


*fig4 : logo JobOdyssey*

# Comment faire un chat réactif et fonctionnel avec les fonctionnalité nécessaire ?

# PLAN

- Recherche et benchmark
- Implémentation du chat
- Implémentations  des messages
- Implémentation d’images et fichiers
- Mise à jour en temps réel

# RECHERCHE ET BENCHMARK

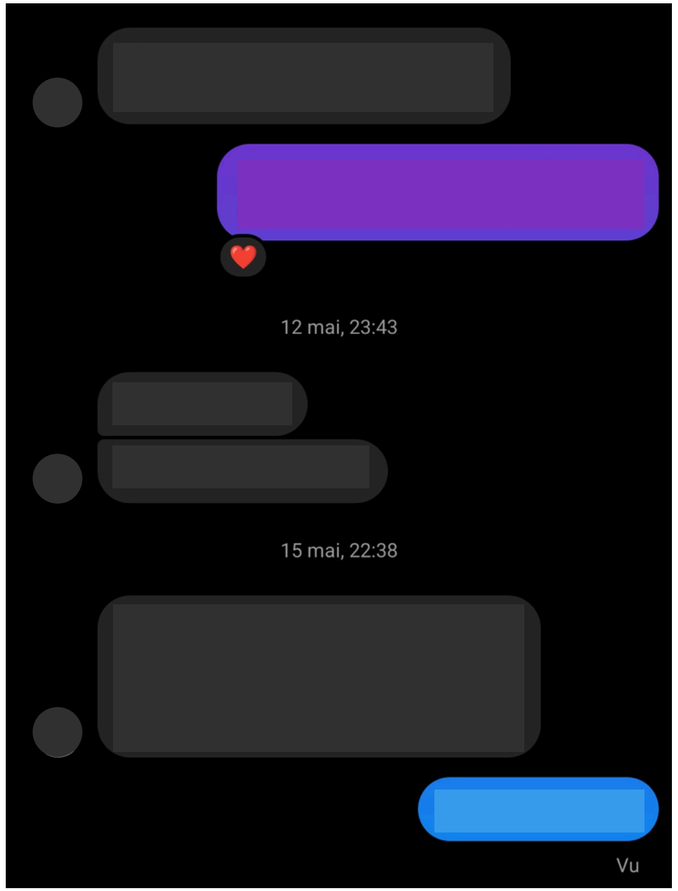


*fig5 : illustration du chat d'Instagram*


*fig6 : illustration du chat de discord*


*fig7 : illustration du chat de twitter*

# RECHERCHE ET BENCHMARK

## Design du chat


*fig8 : Wireframe figma du chat*

## Design des messages

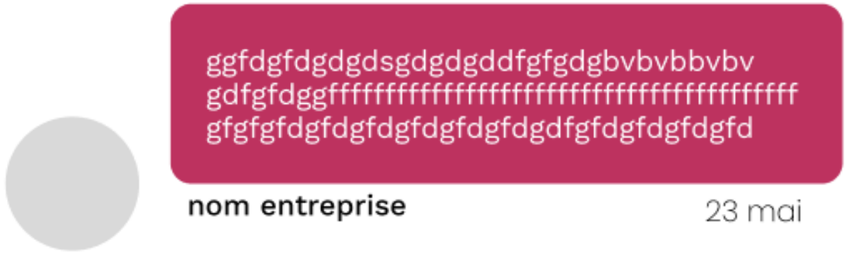


*fig9 : Design du message reçu*

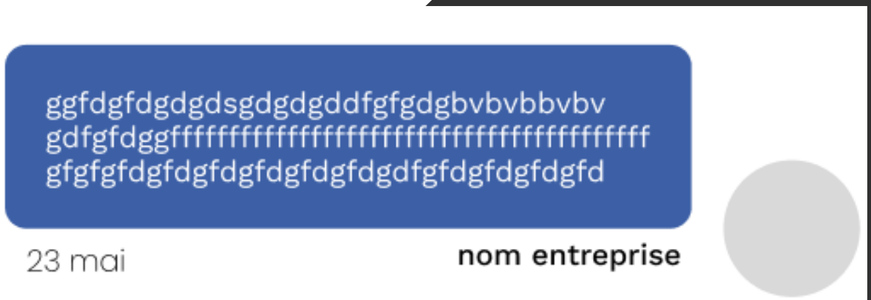


*fig10 : Design du message envoyé*

# IMPLEMENTATION DU CHAT

```
<li className="flex items-end gap-4 justify-start">
        <div className="flex gap-4 pb-3 sm:pb-0 items-center">
            <img className="max-w-10 md:max-w-14 border-2 rounded-full border-black p-1" src="/profile.svg" alt="" />
        </div>
        <div className="sm:pb-2">
            <p className="bg-button rounded-lg p-4 font-medium text-sm md:text-base text-white text-wrap max-w-52 sm:max-w-96 md:max-w-[35rem]">Lorem ipsum dolor sit amet consectetur adipisicing elit. Quas tenetur dolor voluptas doloribus adipisci veritatis velit cupiditate, quos dicta iusto voluptatum nostrum illo pariatur numquam repellendus eaque id atque in.</p>
            <div className="flex flex-col sm:flex-row w-full justify-between px-2">
            <p className="font-medium">Nom de l'entrperise</p>
            <p className=" font-light">23 mai</p>
            </div>
        </div>
    </li>
```

*fig11 : html du message*

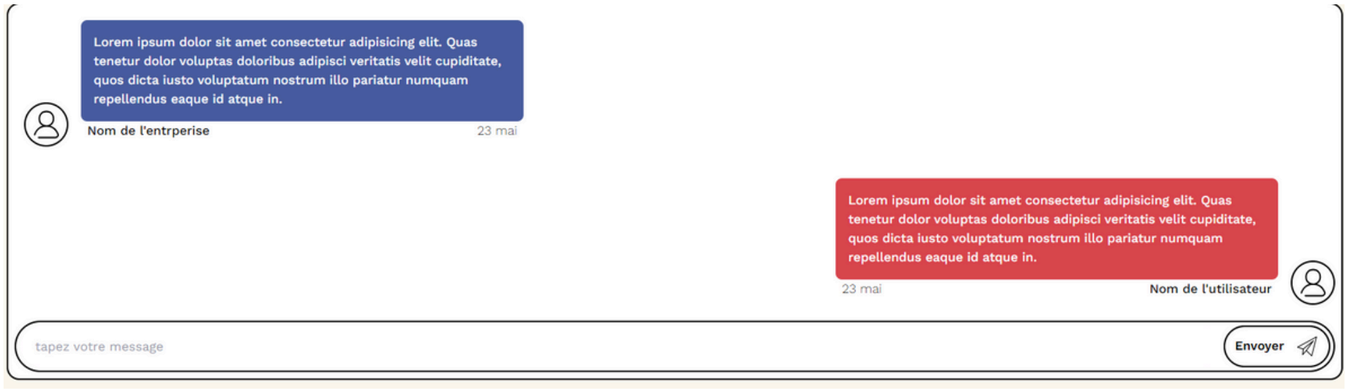


*fig12 : rendu du chat*

# IMPLEMENTATION DES MESSAGES

**URL du chat**

**/offre/candidature/11/4/chat**

```
export async function loader({params}) {
    let offreid = params.offreid;
    let userid = params.userid;
    let connecteduser = await Getuser();
    if(connecteduser.user.role === "Role_User") {
      let user = connecteduser.user;
      let connected = user.role;
        let entreprise = await GetEntreprisebyOffer(offreid);
        entreprise = entreprise.company;
        console.log(entreprise);
        let chatmessage = await GetChat(offreid , userid , entreprise.id);
        return {user , entreprise , chatmessage , offreid , connected };
    }
else if(connecteduser.user.role === "Role_Company") {
  let entreprise = connecteduser.user;
  let connected = entreprise.role;
   let entrepriseid = entreprise.id;
    let user = await GetUserbyId(userid);
    let chatmessage = await GetChat(offreid , userid , entrepriseid);
    return {user , entreprise , chatmessage , offreid , connected};
    }
}
```

*fig13 : loader des données du chat*

# IMPLEMENTATION DES MESSAGES

```
    message.sender === data.connected && data.connected === "Role_User" ? (
    <li key={index} className="flex items-end gap-4 justify-end">
    <div className="flex flex-col w-full items-end pb-2">
        <p className="bg-rouge max-w-52 sm:max-w-96 md:max-w-[35rem] rounded-lg p-4 font-medium text-white text-wrap text-end
        break-words">{message.message}</p>
        <div className="flex w-full flex-wrap h-full px-2">
        <p className="font-medium w-full text-end">{data.user.nom} {data.user.prenom}</p>
        <p className=" font-light w-full text-end ">{TransformDate(message.createdAt)}</p>
        </div>
    </div>
    <div className="flex gap-4 items-center">
    <img className="max-w-10 md:max-w-14 border-2 rounded-full border-black p-1" src={`/uploads/${data.user.photoprofile}`}alt=""
    >
    </div>
</li>
```

*fig14 : html de l'affichage du type de message*

```
: message.sender === data.connected && data.connected === "Role_Company" ?
```

*fig15 : condition du type de message*

# IMPLEMENTATION DES MESSAGES

```
const Addmessage = async () => {
  let message = document.querySelector("#inputmessage").value;
  if (message === "") {
    return;
  }
  let Newmessage = {
    message: message,
    sender: data.connected,
  };
  let ChatData = new FormData();
  ChatData.append("message", message);
  ChatData.append("userId", data.user.id);
  ChatData.append("companyId", data.entreprise.id);
  ChatData.append("offreId", data.offreid);
  ChatData.append("sender", data.connected);
```

*fig16 : fonction d'ajout de message*

```
document.querySelector('#inputmessage').value = "";
let response = await AddChatMessage(ChatData);
if(response.message === "message envoyé") {
    if(chatmessage.message === "pas de messages") {
        setChatmessage([])
        socketRef.current.emit('SendMessage', "new message");
        Newmessage.createdAt = new Date();
        setLocalMessage([... LocalMessage, Newmessage]);
    }
```

*fig17 : fonction d'ajout de message suite*

# IMPLEMENTATION DES IMAGES ET FICHIERS

```
<ul className="flex gap-5 pb-2 flex-wrap">
{filepreview.length > 0  ? (
    filepreview.map((file, index) => (
        VerifyFileExt(file.name) === "image" ? (
    <li>
        <img id={file.url} data-filename={file.name}  className="bg-white p-1 rounded-md w-9 cursor-pointer absolute" onClick=
        {DelImage} src="/burgerclose.svg" alt="" />
    <img  key={index} className=" aspect-auto max-w-40 max-h-40 rounded-md" src={file.url} alt="" />
    </li>
        ) : VerifyFileExt(file.name) === "document" ? (
            <li className="">
                <img id={file.url} data-filename={file.name}  className="bg-white p-1 bg-opacity-10 rounded-md w-10
                cursor-pointer absolute" onClick={DelImage} src="/burgerclose.svg" alt="" />
            <div key={index} className="flex cursor-pointer p-4 rounded-lg gap-2 items-center justify-center bg-rouge
            hover:brightness-90">
    <p className="font-semibold text-white">{file.name}</p>
     </div>
            </li>
        ) : null
))) : null}
    </ul>
    {ErrorMaxFile ? ( <p className="flex font-semibold w-full justify-center py-1 items-center text-red-500">Vous avez atteint
    la limite de fichier par message</p>) : null}
    {ErrorFile ? ( <p className="flex font-semibold w-full justify-center py-1 items-center text-red-500">Le fichier est trop
    lourd</p>) : null}
```

*fig18 : html de l'ajout des fichier et images*

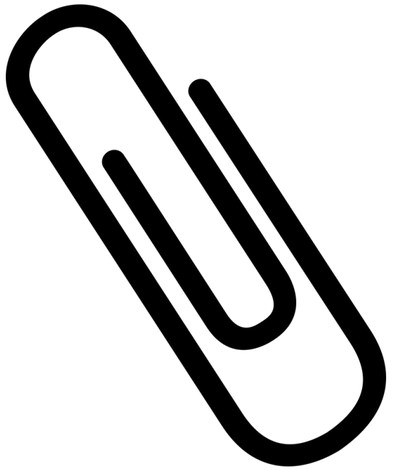

*fig19 :  trombones*

# IMPLEMENTATION DES IMAGES ET FICHIERS

```
const AddImage = (event) => {
    const file = event.target.files[0];
    if (file) {
        if (file.size > 5 * 1024 * 1024) {
            setErrorFile(true);
            setTimeout(() => {
                setErrorFile(false);
            }, 3000);
        } else {
            if(files.length > 3) {
                setErrorFile(true);
                setTimeout(() => {
                    setErrorMaxFile(false);
                }, 3000);
                return;
            }
            files.push(file);
            console.log(files);
            let urlfile = URL.createObjectURL(file);
            console.log(urlfile);
            setFilepreview([...filepreview, { url: urlfile, name: file.name }]);
        }
    };
};
```

*fig20 : fonction d'ajout de fichier*

```
const DelImage = (event) => {
    let urlfile = event.target.id;
    let filename = event.target.dataset.filename;
    console.log(filename);
    setFilepreview(filepreview.filter(file => file.url !== urlfile));
    setFiles(files.filter(file => file.name !== filename));
    console.log(files);
    console.log(filepreview);
};
```

*fig21 : fonction de suppression de fichier*

```
const VerifyFileExt = (file) => {
    let ext = file.split('.').pop();
    if (ext === "jpg" || ext === "jpeg" || ext === "png") {
        return "image";
    } else if(ext === "pdf" || ext === "doc" || ext === "docx") {
        return "document";
    }
}
```

*fig22 : fonction de vérification d'extention*

# IMPLEMENTATION DES IMAGES ET FICHIERS


*fig24 : résultat d'affichage de message*

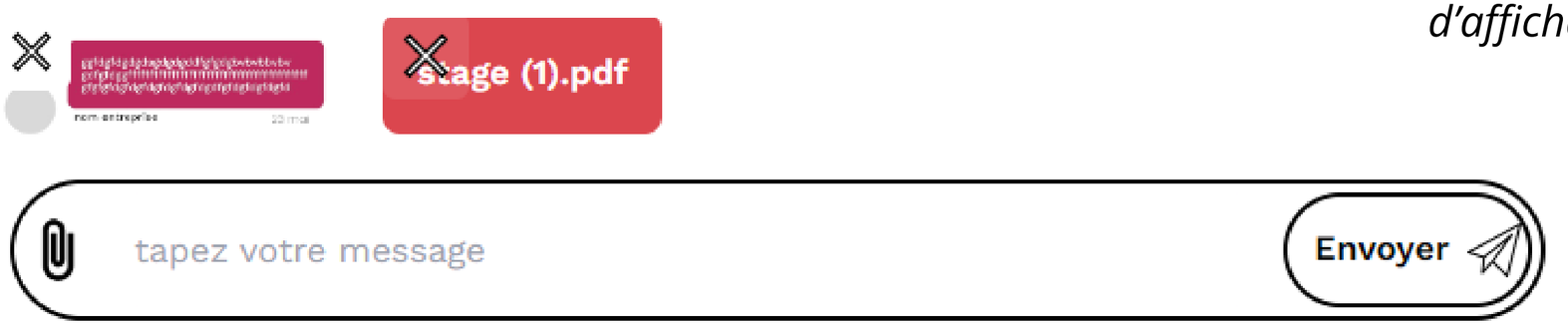


*fig23 : résultat de prévisualisation de fichier*

# MISE A JOUR EN TEMPS REEL


*fig25 : logo socket.io*

**Web socket**

**Node.JS**

**React**

# MISE A JOUR EN TEMPS REEL

## Comment ça marche ?

```
const socket = io('http://localhost:3000', {
    auth: {
      token: token,
    },
    query: {
      "userId": data.user.id,
      "companyId": data.entreprise.id,
      "offreId": data.offreid,
    }
  });
```

*fig26 : constante de connexion front de socket.io*

**Front end**

```
const { Server } = require('socket.io');
const { createServer } = require('node:http');
const server = createServer(app);

const io = new Server(server, {
  cors: {
      origin: ['http://localhost:5173', 'http://localhost:80', 'https://localhost:443'
      , 'http://localhost:3000' , 'http://localhost:3306' ,
      'https://jobodyssey.quentinbrandy.fr' , 'http://jobodyssey.quentinbrandy.fr'],
      methods: ["GET", "POST"]
  }
});
```

```
server.listen(PORT, () => console.log(`Listening on port: ${PORT}`));
```

```
io.on('connection', async (socket) => {
  try {
    let user = await log.authenticateJWTSocket(socket.handshake.auth.token);
    console.log(user);

    const roomName = `${socket.handshake.query.offreId}_${socket.handshake.query.companyId}_${socket.handshake.query.userId}`;
    console.log(roomName);
    socket.join(roomName);
```

*fig27 : Ensemble de code de connexion back de socket.io*

**Back end**

# MISE A JOUR EN TEMPS REEL

```
socket.on('SendMessage', async (newmessage)  => {

  if(newmessage) {
    let allmessages = await chat.getChatSocket( socket.handshake.query.offreId ,
      socket.handshake.query.userId ,  socket.handshake.query.companyId  );

    socket.to(roomName).emit('GetMessage', allmessages);
  }
 else{
  socket.to(roomName).emit('GetMessage', "problème dans l'update");
 }
});
```

*fig28 : Fonction socket.io de nouveau message*

```
socketRef.current.on('GetMessage', (message) => {
        if(message === "problème dans l'update") {
            return;
        }
        else if (chatmessage.message === "pas de messages") {
            setChatmessage("")
            setLocalMessage("");
            setChatmessage(message);
            scroll.scrollToBottom({
                containerId: 'messageList'
            }, {smooth: false , duration: 0});
        }
        setLocalMessage("");
      setChatmessage(message);
    });
```

*fig29 : Fonction socket.io de récupération message*

**Back end**

**Front end**

# BILAN PERSONNEL ET PROFESSIONNEL

# MERCI POUR VOTRE ATTENTION