Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER-PROJET

- *Nom de naissance* ▸ Reinbacher
- *Nom d'usage* ▸ Reinbacher
- *Prénom* ▸ Hedwig
- *Adresse* ▸ 9 passage feuillat, 69003 Lyon

Titre professionnel visé

Développeur-se logiciel - Niveau III

Sommaire:

1. Le concept de “Neksoo”

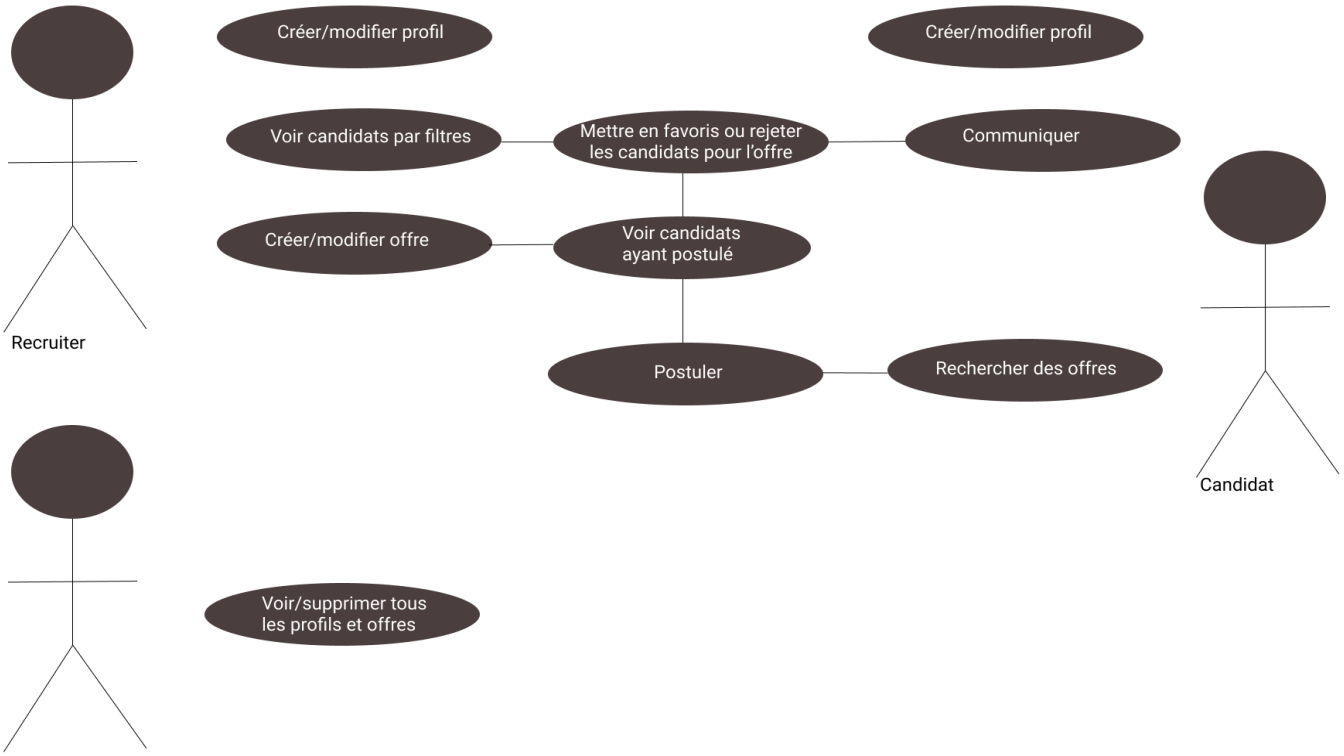
1.1 Concept

Le terme “Neksoo” comme terme joue sur la consonance du mot “connection” (anglais) et représente l’idée de créer un lien entre un recruteur et un candidat.

L’application a comme objectif de faciliter cette création de lien en mettant en avant des candidats sous forme de “cards”, sur lesquelles seules les compétences sont visibles.

Si le candidat met en favoris une offre et qu’un recruteur met en favoris son profil pour cette même offre, alors ces deux personnes peuvent s’envoyer des messages, et le recruteur a accès au CV détaillé en pdf suite au matching.

1.2 User stories



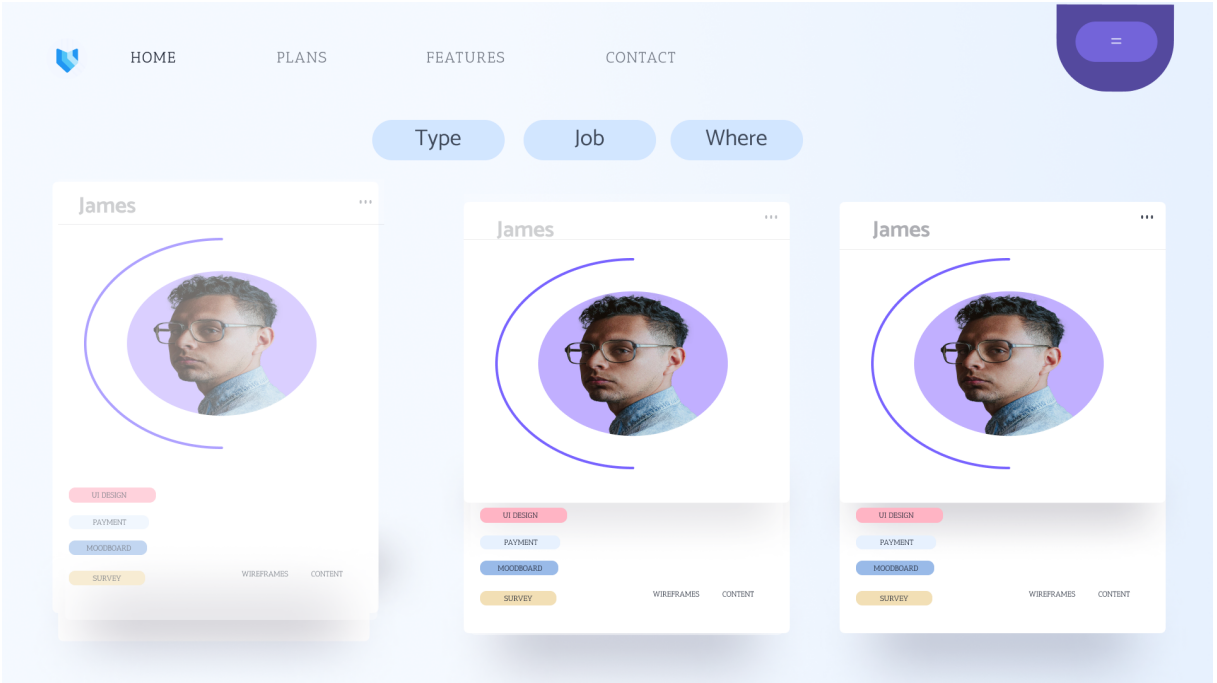
1.3 User case

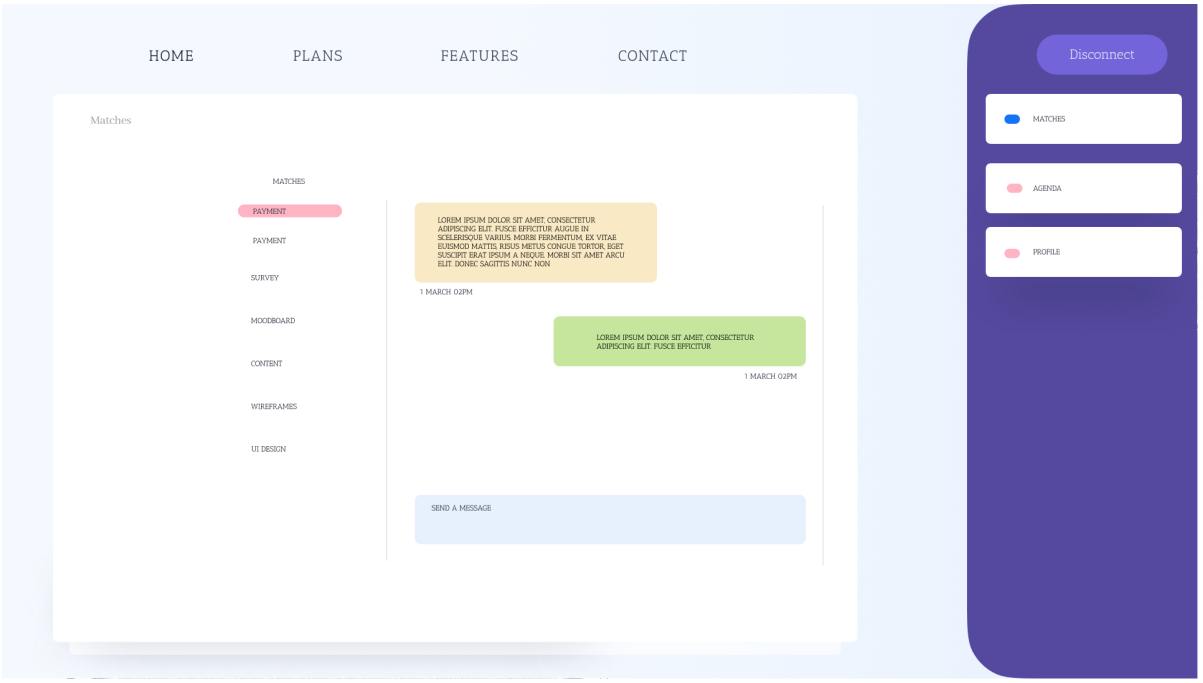
User case name	Candidates Search	
Description	Recruiter search to find candidates corresponding to names or citerias	
Actors	User recruiter	
Pre-condition	Being connected to network	
Post-Condition	Creating a job search with specific conditions	
Main scenarios	Serial No	Steps
Search by job	1	Choose job on dropdown menu
Search by cities	1-5	Global search with cities as keywords
Search by skills	2	Global search with skills as keywords

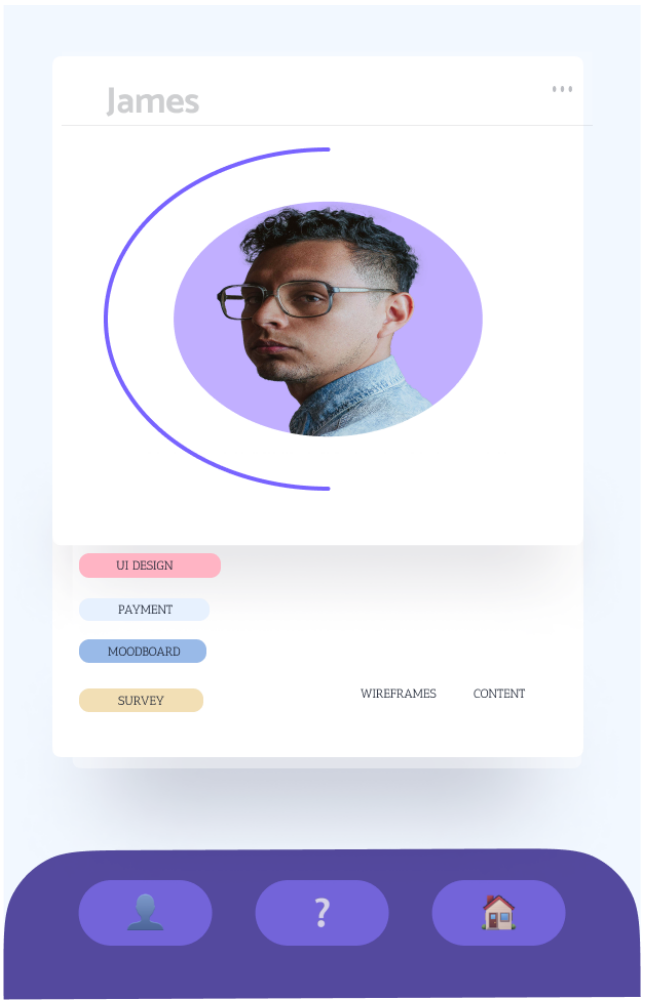
1.4 Maquettes

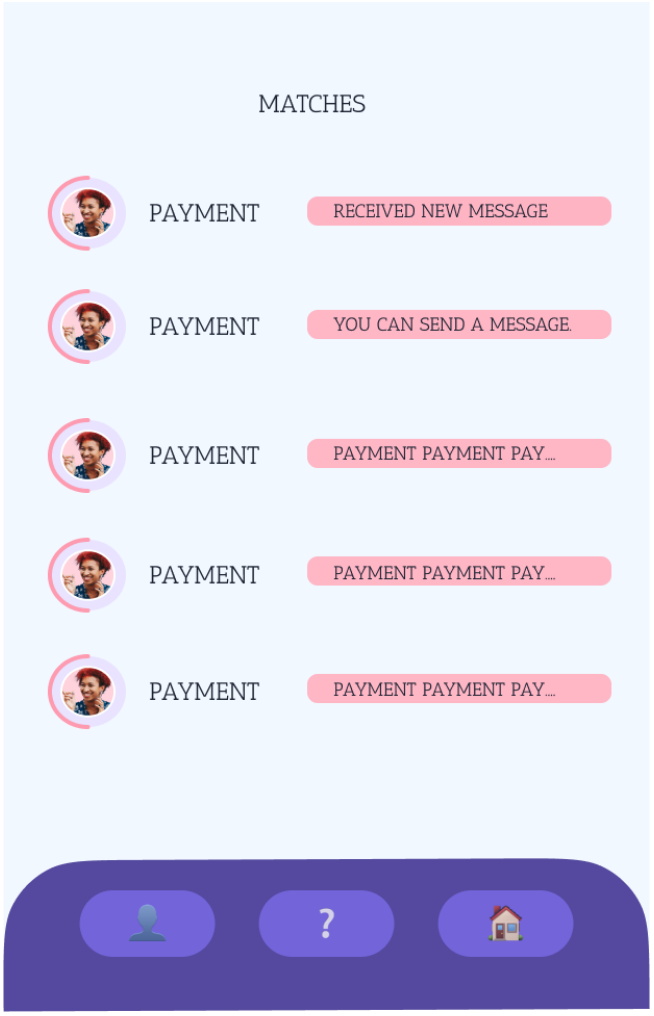
Intro: logiciel utilisé “figma”


neks
Connekt quickly.
Email
Password
Login

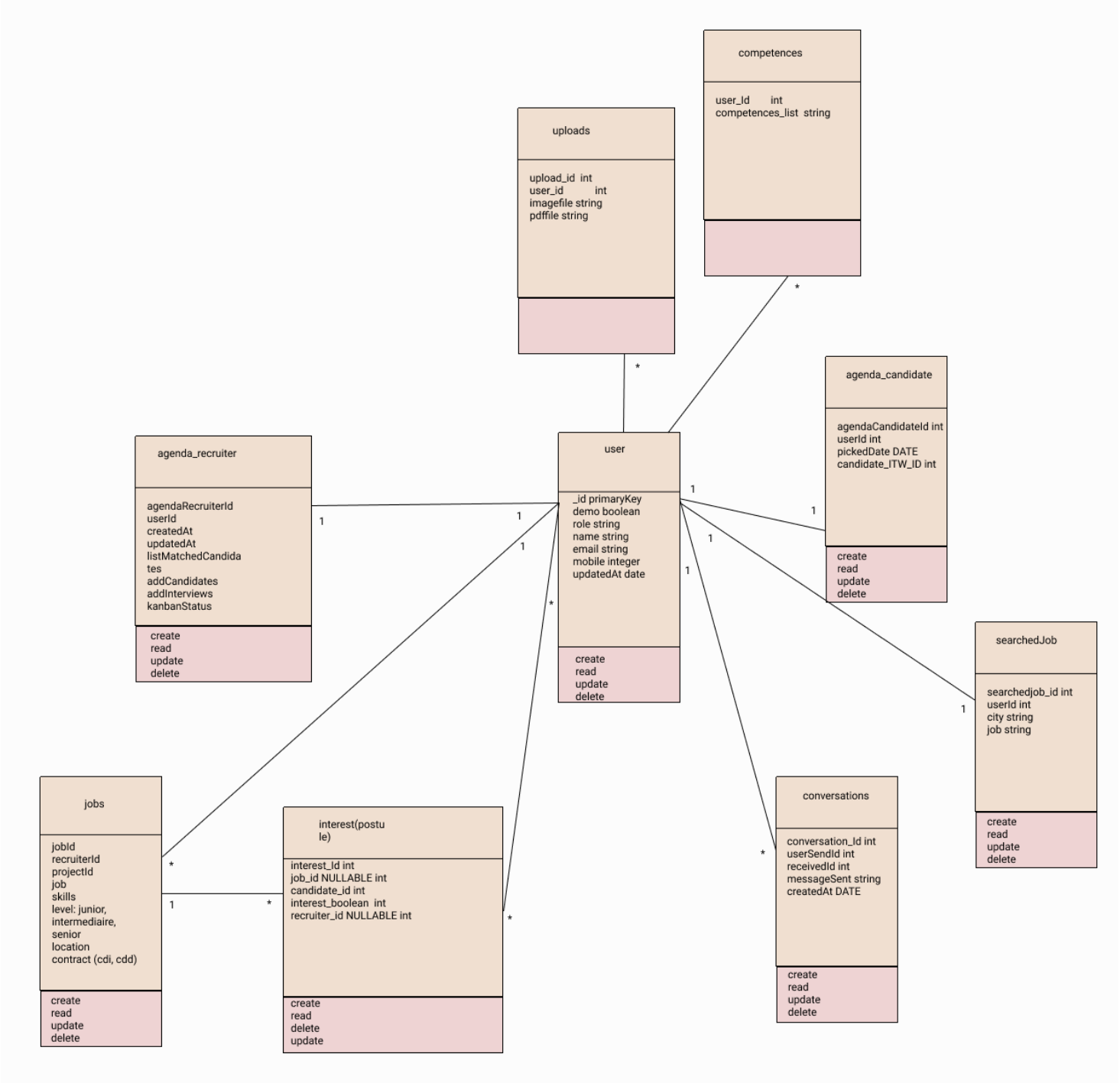

HOME
PLANS
FEATURES
CONTACT
Type
Job
Where
James
UI DESIGN
PAYMENT
MOODBOARD
SURVEY
WIREFRAMES
CONTENT


HOME
PLANS
FEATURES
CONTACT
Disconnect
Matches
MATCHES
PAYMENT
PAYMENT
SURVEY
MOODBOARD
CONTENT
WIREFRAMES
UI DESIGN
LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT. FUSCE EFFICITUR AUGUE IN SCELERISQUE VARIUS. MORBI FERMENTUM, EX VITAE EUISMOD MATTIS, RISUS METUS CONGUE TORTOR, EGET SUSCIPIT ERAT IPSUM A NEQUE. MORBI SIT AMET ARCU ELIT. DONEC SAGITTIS NUNC NON
1 MARCH 02PM
LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT. FUSCE EFFICITUR
1 MARCH 02PM
SEND A MESSAGE
MATCHES
AGENDA
PROFILE


James
UI DESIGN
PAYMENT
MOODBOARD
SURVEY
WIREFRAMES
CONTENT


MATCHES
PAYMENT
RECEIVED NEW MESSAGE
PAYMENT
YOU CAN SEND A MESSAGE.
PAYMENT
PAYMENT PAYMENT PAY....
PAYMENT
PAYMENT PAYMENT PAY....
PAYMENT
PAYMENT PAYMENT PAY....
?

1.5 UML pour le SQL



Le back-end

Intro Langages utilisés: SQL, express, sharp,

2.1 Création de la base de données SQL

```
▶ Run SQL
DROP TABLE IF EXISTS jobOffers;
▶ Run SQL
CREATE TABLE jobOffers(
  jobOffer_id INTEGER AUTO_INCREMENT PRIMARY KEY,
  recruiter_id INTEGER(100),
  CONSTRAINT FK_recruiter_id FOREIGN KEY (recruiter_id) REFERENCES user(user_id),
  available TINYINT DEFAULT(1),
  remote VARCHAR(50),
  organizationName VARCHAR(100),
  jobOffer_role VARCHAR(100),
  jobOffer_description VARCHAR(100),
  country VARCHAR(100),
  city VARCHAR(100),
  updatedAt DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

2.2 Mise en place des repositories pour accéder à la base de données

2.2.1 Jointures SQL

```
static async getCandidatesWithInterestByJob(job_id) {
    try {
        const [rows] = await connection.query<RowDataPacket[]>(`SELECT * FROM user
    INNER JOIN interest WHERE user_id = candidateWhoApplied_id  AND jobApplied_id=? AND interest IS NULL AND role="candidat"`, [job_id.jobApplied_id]);

        return rows
    }
```

Retranscription de la requête pour plus de visibilité: “INNER JOIN interest WHERE user_id = candidateWhoApplied_id AND jobApplied_id=? AND candidateWhoApplied_id=? AND interest IS NULL AND role="candidat" AND recruiterJobOffer_id IS NULL”

2.2.2 Having By SQL

2.2.3 Requêtes complexes SQL : LEFT OUTER JOIN

```
static async getJobCandidatesWithoutInterestByJob(job_id: number) {
    const [rows] = await connection.query<RowDataPacket[]>(`SELECT * FROM user LEFT OUTER JOIN interest ON user_id=candidateWhoApplied_id WHERE jobApplied_id NOT LIKE ? OR

    return rows.map(row => new User({
        user_id: row['user_id']
    }));
}
```

Retranscription de la requête SQL pour plus de visibilité :

“SELECT * FROM user LEFT OUTER JOIN interest ON user_id=candidateWhoApplied_id WHERE jobApplied_id NOT LIKE ? OR jobApplied_id IS NULL AND role="candidat"”

2.2.4 Requête mélangeant les résultats

```
static async getAllJobOffers() {
    const [rows] = await connection.query<RowDataPacket[]>(`SELECT * FROM jobOffers ORDER BY RAND()`);
```

Status: 200 OK Size: 781 Bytes Time: 8 ms

Response Headers 7 Cookies Test Results


```
{
  "success": true,
  "count": 4,
  "data": [
    {
      "jobOffer_id": 13,
      "recruiter_id": 302,
      "available": 1,
      "organizationName": "OrangeCity",
      "jobOffer_role": "Actuary",
      "jobOffer_description": "NewName",
      "updatedAt": "2022-01-25T14:43:59.000Z"
    },
    {
      "jobOffer_id": 15,
      "recruiter_id": 302,
      "available": 1,
      "organizationName": "Colr",
      "jobOffer_role": "Agricultural manager",
      "jobOffer_description": "NewName",
      "updatedAt": "2022-01-25T14:44:48.000Z"
    },
    {
      "jobOffer_id": 14,
      "recruiter_id": 302,
      "available": 1,
      "organizationName": "Lezit",
      "jobOffer_role": "Administrative assistant",
      "jobOffer_description": "NewName",
      "updatedAt": "2022-01-25T14:44:38.000Z"
    },
    {
      "jobOffer_id": 16,
      "recruiter_id": 302,
      "available": 0,
      "organizationName": "KarlCO",
      "jobOffer_role": "Youtuber",
      "jobOffer_description": "Full time",
      "updatedAt": "2019-01-25T23:00:00.000Z"
    }
  ]
}
```

Status: 200 OK Size: 781 Bytes Time: 8 ms

Response Headers 7 Cookies Test Results


```
{
  "success": true,
  "count": 4,
  "data": [
    {
      "jobOffer_id": 14,
      "recruiter_id": 302,
      "available": 1,
      "organizationName": "Lezit",
      "jobOffer_role": "Administrative assistant",
      "jobOffer_description": "NewName",
      "updatedAt": "2022-01-25T14:44:38.000Z"
    },
    {
      "jobOffer_id": 16,
      "recruiter_id": 302,
      "available": 0,
      "organizationName": "KarlCO",
      "jobOffer_role": "Youtuber",
      "jobOffer_description": "Full time",
      "updatedAt": "2019-01-25T23:00:00.000Z"
    },
    {
      "jobOffer_id": 13,
      "recruiter_id": 302,
      "available": 1,
      "organizationName": "OrangeCity",
      "jobOffer_role": "Actuary",
      "jobOffer_description": "NewName",
      "updatedAt": "2022-01-25T14:43:59.000Z"
    },
    {
      "jobOffer_id": 15,
      "recruiter_id": 302,
      "available": 1,
      "organizationName": "Colr",
      "jobOffer_role": "Agricultural manager",
      "jobOffer_description": "NewName",
      "updatedAt": "2022-01-25T14:44:48.000Z"
    }
  ]
}
```


2.2.5 Statistiques: moyenne de métiers par postes proposées

```
static async getAVGjobRoleOffer() {
    const [rows] = await connection.query<RowDataPacket[]>(`SELECT jobOffer_role,CAST(100*count(*)  /
    (SELECT count(*) from jobOffers)AS DECIMAL(4,2))   AS pourcentage  FROM jobOffers group by jobOffer_role;`);

    return rows;

}
```

Status: 200 OK Size: 266 Bytes Time: 63 ms

Response Headers 7 Cookies Test Results


```
{
  "success": true,
  "count": 4,
  "data": [
    {
      "jobOffer_role": "Actuary",
      "pourcentage": "25.00"
    },
    {
      "jobOffer_role": "Administrative assistant",
      "pourcentage": "25.00"
    },
    {
      "jobOffer_role": "Agricultural manager",
      "pourcentage": "25.00"
    },
    {
      "jobOffer_role": "Youtuber",
      "pourcentage": "25.00"
    }
  ]
}
```


2.2.3 Statistiques: Toutes les offres sur un an

```
static async getJobsOfferPerThisYear() {
    const [rows] = await connection.query<RowDataPacket[]>(`SELECT updatedAt AS date, count(*)
    FROM jobOffers
    WHERE updatedAt BETWEEN NOW()-INTERVAL 1 YEAR  AND CURRENT_DATE
    GROUP BY date`);

    return rows;

}
```

Status: 200 OK Size: 182 Bytes Time: 8 ms

Response Headers 7 Cookies Test Results


```
{
  "success": true,
  "count": 3,
  "data": [
    {
      "date": "2022-01-25T14:43:59.000Z",
      "count(*)": 1
    },
    {
      "date": "2022-01-25T14:44:38.000Z",
      "count(*)": 1
    },
    {
      "date": "2022-01-25T14:44:48.000Z",
      "count(*)": 1
    }
  ]
}
```


2.2.3 Statistiques: Toutes les offres non mises à jour depuis un an minimum

```
static async getNonUpdatedJob() {
    const [rows] = await connection.query<RowDataPacket[]>(`    SELECT *
    FROM jobOffers
    WHERE updatedAt  NOT BETWEEN NOW()-INTERVAL 1 YEAR  AND CURRENT_DATE`);

    return rows;

}
```

Status: 200 OK Size: 264 Bytes Time: 8 ms

Response Headers 7 Cookies Test Results


```
{
  "success": true,
  "count": 1,
  "data": [
    {
      "jobOffer_id": 16,
      "recruiter_id": 302,
      "available": 0,
      "remote": "Yes",
      "organizationName": "KarlCO",
      "jobOffer_role": "Youtuber",
      "jobOffer_description": "Full time",
      "country": "France",
      "city": "Lyon",
      "updatedAt": "2019-01-25T23:00:00.000Z"
    }
  ]
}
```


2.3 Controllers pour créer les routes

2.3.1 Les routes protégées

```
JobOffersController.post('/addJob', passport.authenticate('jwt', { session: false }), async (req, res) => {
    try {
        const newJob = new JobOffers(req.body);
        console.log("new", newJob);

        await jobOffers_repository.addJob(req.user['user_id'], newJob);
        res.status(201).json({
            success: true,
            data: newJob
        })

    }
    catch (error) {
        console.log(error);
        res.status(500).json(error);
    }
});
```

2.3.2 présentation du jwt et du token

```
export function generateToken(payload: any, expire = 60 * 60) {
    const token = jwt.sign(payload, process.env.JWT_SECRET, { expiresIn: expire });
    return token;


}


export function configurePassport() {
    passport.use(new Strategy({
        jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
        secretOrKey: process.env.JWT_SECRET
    }, async (payload, done) => {


        try {
            const user = await user_repository.getUser(payload.email);

            if (user) {

                return done(null, user);
            }

            return done(null, false);
        } catch (error) {
            console.log("jwt error is", error);
            return done(error, false);
        }
    }))
```

2.3.3 Joi le validateur

2.4 Les limites de express

```
configurePassport();
server.use(express.urlencoded({limit: '2000mb', extended: false,  parameterLimit:50000,type:'application/x-www-form-urlencoded' }));
server.use(express.text({ limit: '2000mb' }));
```

2.5 Enregistrer des éléments sous forme base64

2.5.1 Les images et sharp

```
export async function uploadImage(base64: string) {
    /*   const body = await got(base64).buffer();
      console.log("body", body);
   */
    const baseImage = randomUUID() + '.jpeg';
    const uri = base64.split(';base64,').pop()
    const buffer = Buffer.from(uri, 'base64url');
    const img = sharp(buffer)


    await Promise.all([
        // img.toFormat('png'),
        img.toFile(__dirname + '/../../public/uploads/' + baseImage),
        img.resize(200, 200).toFile(__dirname + '/../../public/uploads/thumbnails/' + baseImage)])
        .then(res => { console.log("Done ===============>!", res); })
        .catch(err => {
            console.error("Error sharp promise", err);
        });
    console.log("base ilage", baseImage);


    return baseImage;
}
```

2.5.2 Décoder pdf traduit en base64 avec node

```
export async function uploadPdf(base64: string) {

    const uriPDF = base64.split(';base64,').pop()
    const buffer = Buffer.from(uriPDF, 'base64url');
    const namePdf = randomUUID() + '.pdf';
    await fs.writeFileSync(__dirname + '/../../public/pdfs/' + namePdf, buffer)

    return namePdf;

}
```

Conclusion pour le back-end: