

# **DOCUMENTATION DE NOTRE APPLICATION DE GESTION DES UTILISATEURS AVEC LE FRAMEWORK ANGULAR WEB GROUPE HAKASA ADEFNIPA SIMPLON**



## **Documentation technique**



## DOCUMENTATION DE NOTRE APPLICATION WEB GROUPE HAKASA

| INFORMATIONS           |   |
|------------------------|---|
| Nom du Projet          | application web gestion des utilisateurs<br>avec le framework ANGULAR       |
| Type de de<br>document | Documentation technique   |
| Date                   | 21/05/2021  |
| Version                | 1.0   |
| Mot clés               | Architecture – Fonctionnement –<br>Technologies – Diagrammes – Interactions |
| Auteurs                | MAMADOU SEMBENE<br>MAME MOR THIAM<br>BARAKA KOUMA<br>YACINE CAMARA          |

## Table de Matière

|  |          |
|--|----------|
| <b>I-Résumé du document.....</b>   | <b>4</b> |
| <b>II - Rappel sur le fonctionnement de l'application.....</b>                 | <b>4</b> |
| <b>II.1 - Description de l'application.....</b>                                | <b>4</b> |
| <b>II.2 - Décomposition du projet.....</b>                                     | <b>5</b> |
| <b>II.3 - Architecture globale.....</b>  | <b>5</b> |
| <b>II.4 - Technologies utilisées .....</b>                                     | <b>6</b> |
| <b>II.5 – Diagramme de classes.....</b>  | <b>6</b> |
| <b>II.6 - Diagramme cas d'utilisation.....</b>                                 | <b>7</b> |
| <b>III -Structure du projet.....</b>   | <b>8</b> |
| <b>IV -Comment utiliser l'application .....</b>                                | <b>9</b> |
| <b>V- Quelques code de Compréhension, du Projet</b>                            |          |
| <b>VI- lien du Projet</b>  |          |
| <b><i>1.Se connecter</i></b>   |          |
| <b><i>2.Inscription</i></b>  |          |
| <b><i>3.Gestion des Rôles(administrateur,formateur, étudiant, finance)</i></b> |          |

## Résumé du document

Ce document est la documentation technique de notre application web qui va gérer la gestion des utilisateurs avec l'utilisation d'un framework ANGULAR.

Il est divisé en trois parties :

- Le fonctionnement de l'application ;
- Le structure du projet ;
- L'utilisation de l'application.

## Rappel sur le fonctionnement de l'application

### Description de l'application

Notre projet a pour but de mettre en place une application web qui assure la gestion des utilisateurs avec l'utilisation du framework angular. L'application gère la connexion, la création de compte et la gestion des rôles(formateur, administrateur, finance, etudiant).

- ❑ **La connexion:** l'utilisateur peut accéder à son espace personnel avec son login et mot de passe ou avec son compte gmail.
- ❑ **La création de compte:** Pour la création, il y aura un formulaire où on va renseigner les informations de l'utilisateur: nom, prénom, pseudo,

adresse, téléphone, email et checkbox (pour l'attribution de rôles) le mot de passe est autogénéré.

Après ouverture du compte par l'administrateur, la personne concernée reçoit un mail avec ses accès (email et/ou pseudo et mot de passe).

PS: le mot de passe est généré automatiquement à l'inscription et crypté lors du stockage.

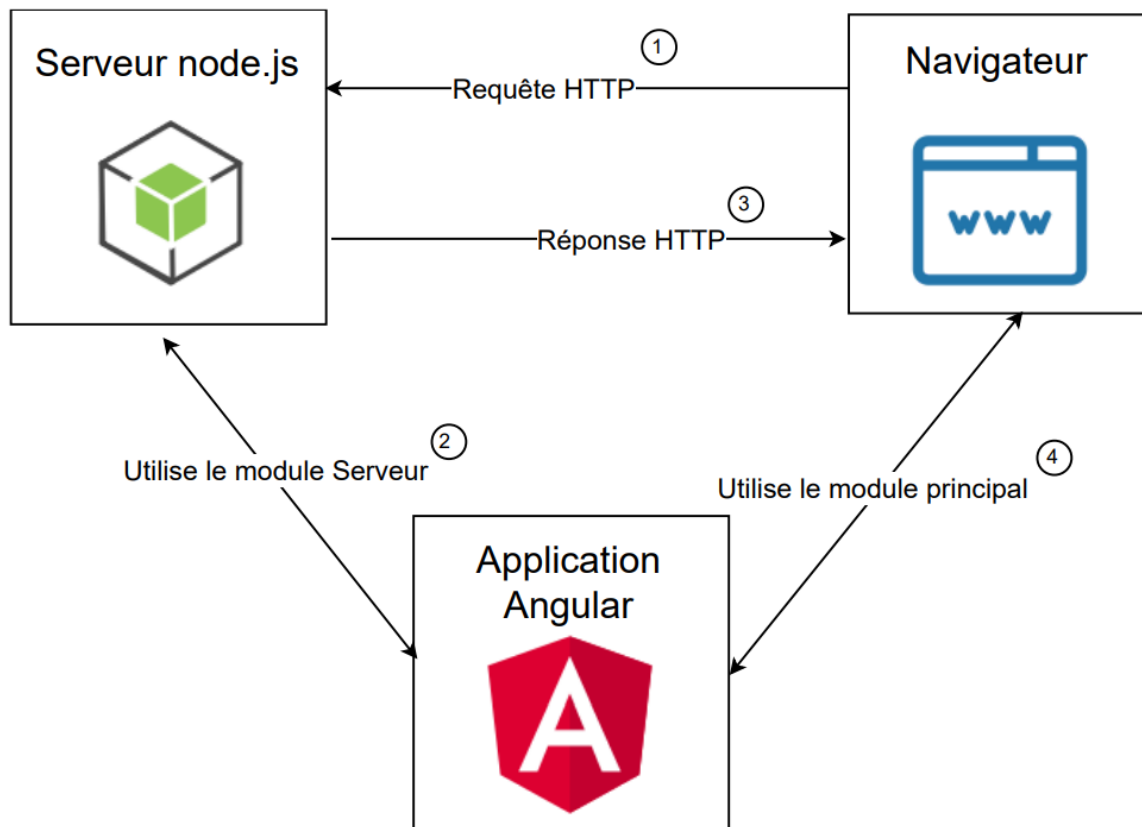
- ❑ **Administration de la plateforme:** Les administrateurs peuvent modifier les rôles (retirer, ajouter des rôles) de n'importe quel utilisateur. La gestion de rôles permet la limitation de chaque utilisateur en lui redirigeant juste dans sa partie qui lui concerne dans l'application.

## Décomposition du projet

Notre projet se décompose en différentes parties :

- ❑ **Gestion de la connexion**
- ❑ **Création du compte**
- ❑ **Gestion des rôles**

## Architecture globale



- Initialement, cela commence par Angularjs où toutes les demandes émanant du côté client sont traitées.

- Une fois la phase initiale terminée, la demande entre dans Node qui contrôle le côté serveur.
- La demande entre dans Express.js où elle demande l'accès à la base de données.
- Mysql collecte toutes les données une fois la demande acceptée et les transmet à Express.js
- Express relaie la réponse à Node qui l'envoie à Angular.

## Technologies utilisées

Pour les technologies que nous avons utilisées pour développer ce projet:

**Jira** comme outils de gestion de projet

**Github, Slack** comme outils de travail collaboratif

**Figma** pour le maquettage de l'application web.

**UML avec LUCHIDCHART** pour la modélisation de l'application web

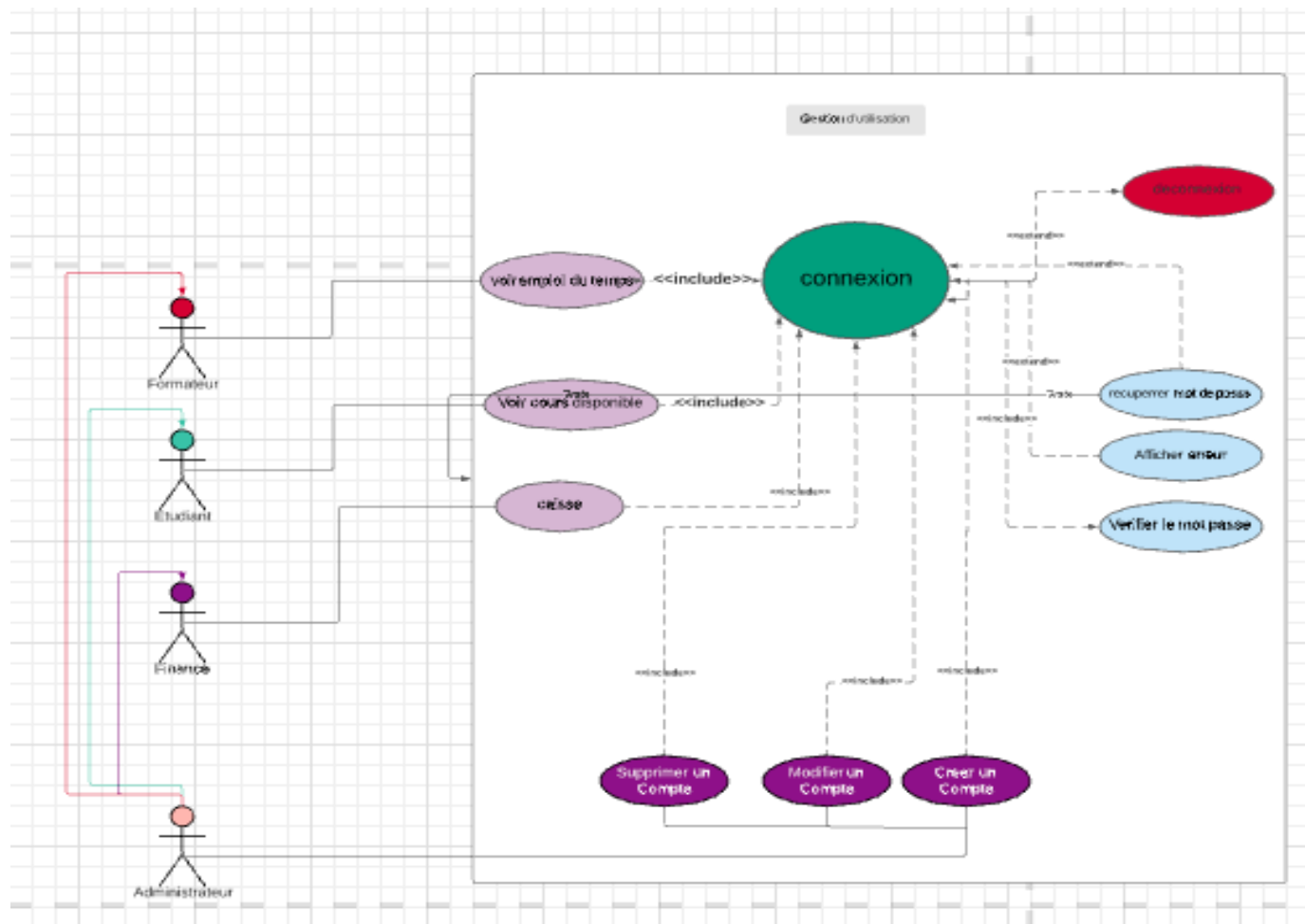
**Angular** comme framework pour le développement frontend de l'application web.

**Nodejs & Express** pour gérer la partie backend

**Mysql** comme système de base de données qui assure le stockage et la restitution en cas de besoin des informations.

## Diagramme cas d'utilisation:

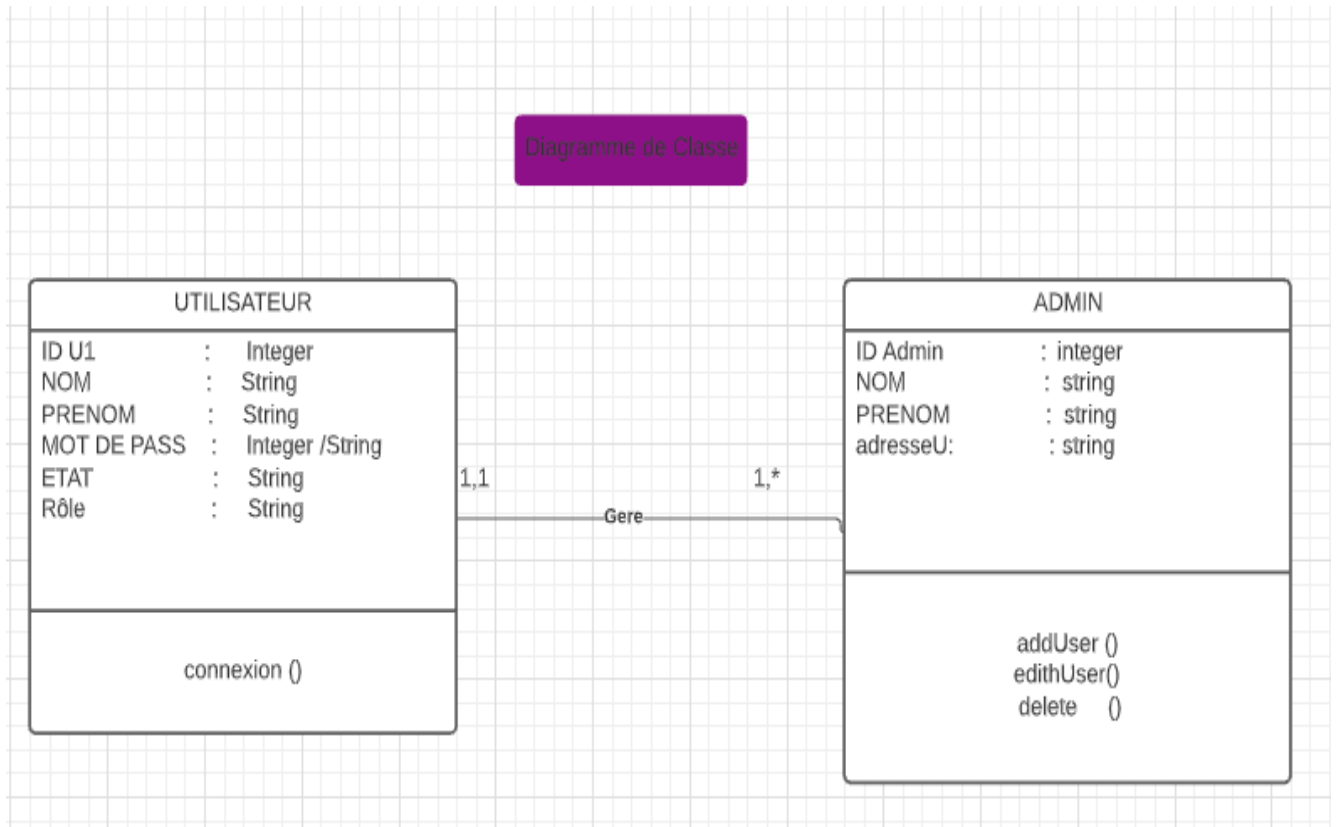
Ce diagramme présente l'ensemble des acteurs du projet ainsi que les fonctionnalités de l'application.





## Diagramme de classe

Le diagramme de classes est un schéma utilisé pour représenter les interfaces des systèmes ainsi que leurs relations.



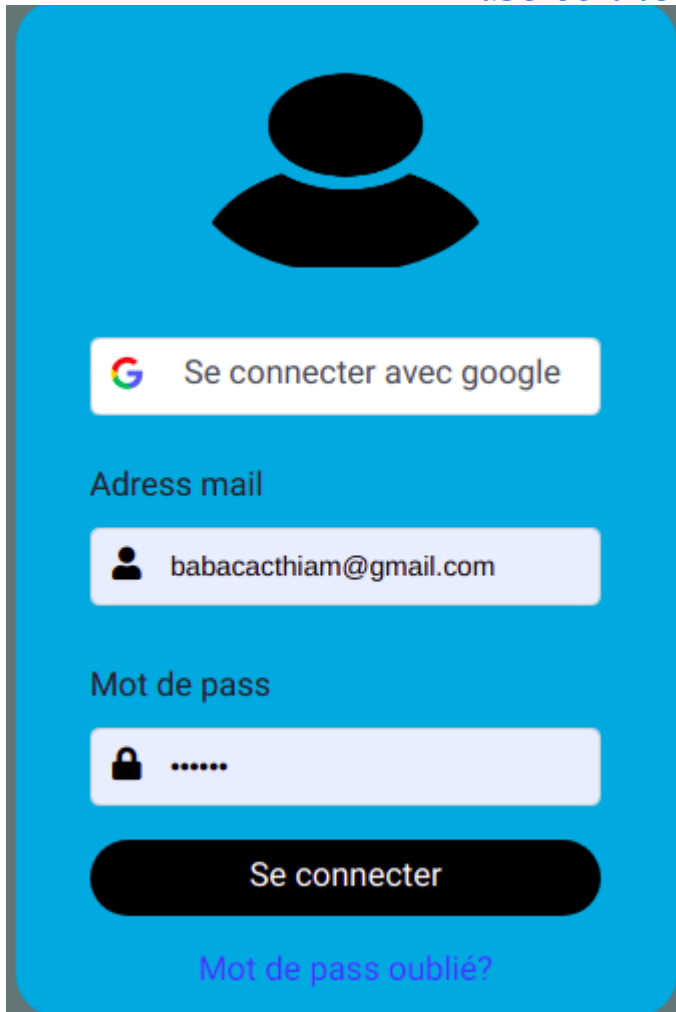
### **III -Structure du projet**

La structure du Projet se présente sur

## IV -Comment utiliser l'application .

Avec notre application web, on gère la gestion des utilisateurs. L'application est composée de trois (3) parties: La Connexion (), La Création de Compte (Inscription) et La Gestion des Rôles (formateur, administrateur, finance etudiant).

### *1.Se connecter*

The image shows a login form on a blue background. At the top is a black silhouette of a person's head and shoulders. Below it is a white button with the Google 'G' logo and the text 'Se connecter avec google'. Underneath is the label 'Adress mail' followed by a white input field containing the email 'babacacthiam@gmail.com'. Below that is the label 'Mot de pass' followed by a white input field with a lock icon and six dots. At the bottom is a large black button with the text 'Se connecter' in white. Below the button is a link in purple text that says 'Mot de pass oublié?'.


*Tout utilisateur ayant un compte dans l'application peut accéder à son espace personnel :*


- avec son compte google*
- et/ou avec son email ou pseudo et mot de passe*


*La récupération de mot de passe, l'utilisateur a la possibilité de récupérer son mot de passe une fois oublié.*







Et si l'utilisateur n'a pas de compte il peut cliquer sur inscription ou création de compte .

## 2.Inscription

 **Espace Administrateur**

 Finance

 Etudiant

| Nom     | Prenom  | Role      | Email                 | Action  |
|---------|---------|-----------|-----------------------|---|
| khouma  | baraka  | admin     | barakakouma@gmail.com |   |
| Sembene | Mamadou | formateur | mouhammadou@gmail.com |   |
| Sembene | Mamadou | admin     | mamadoucmbn@gmail.com |   |

**FORMULAIRE D'INSCRIPTION**

Remplissez soigneusement le formulaire d'inscription.

**Prénom**

**Nom**

**email**

**Tel**

**Role**

Soumettre

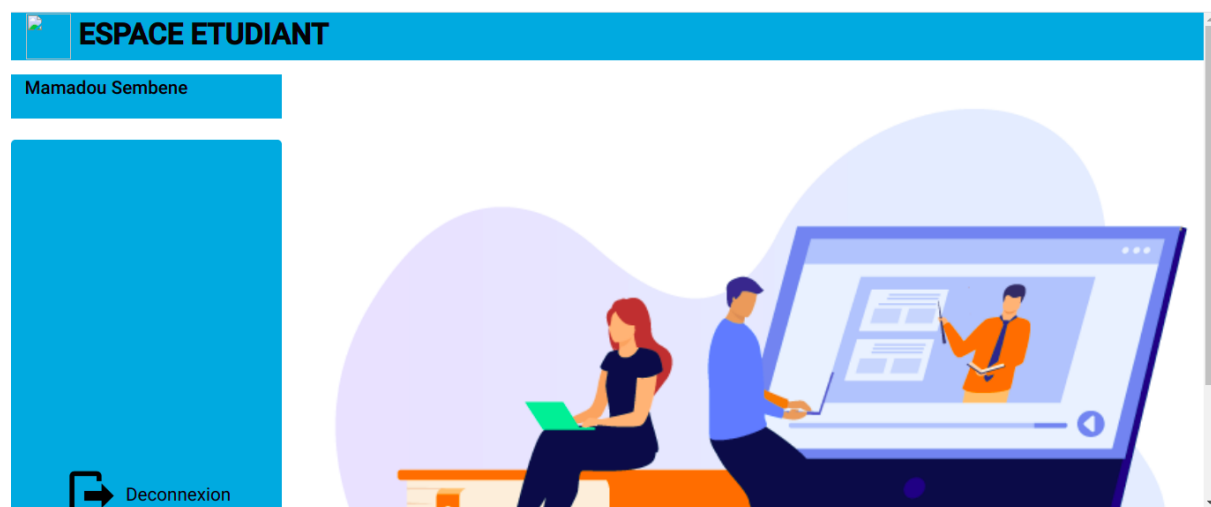
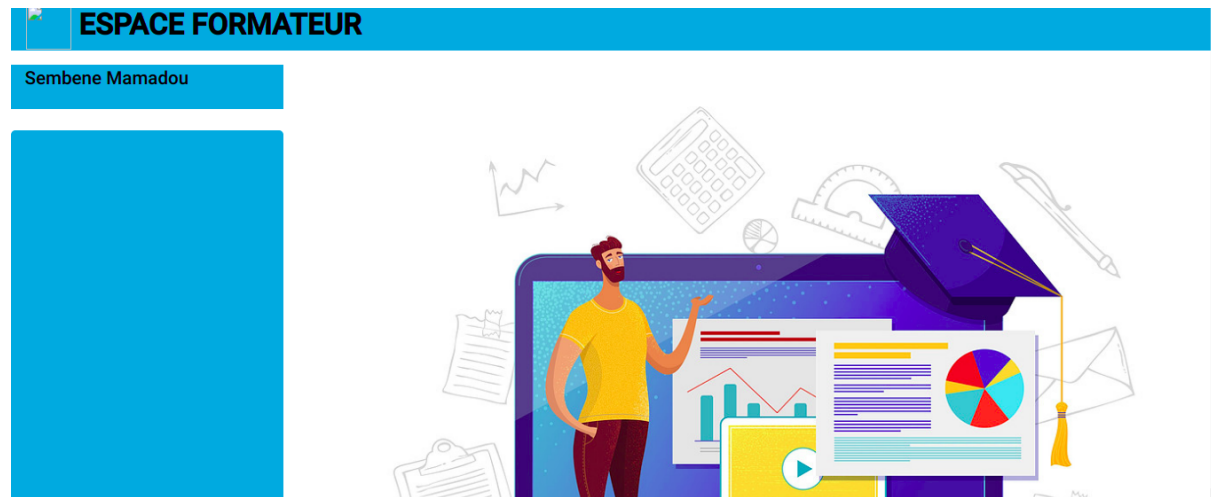
*Et seule l'utilisateur admin peut créer un compte. Mais l'utilisateur doit renseigner ces informations :*

- *Nom:*
- *prénom:*
- *pseudo:*
- *Adresse:*
- *Téléphone:*
- *email :*

*et checkbox(pour l'attribution de rôles) le mot de passe de ce compte est autogénéré.*

### **3.Gestion des Rôles(administrateur,formateur, étudiant, finance)**





## V- Quelques code de Compréhension, du Projet

Ce code indique le gestion d'administration du site,

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<title>Document</title>

</head>

<body>

<div>

    <div class="row">

        <div class=" col-md-12 text-*-left"
style="background-color:#22B0DD">

            <h2 id="policeEtud">  <a
href="#"

                id="ahrefEtud"> Espace Administrateur </a></h2>

        </div>

    </div>

</div>

<div class="row">

    <div class="col-md-3">

        <div class="row mt-3 container-sm
text-*-center"style="background-color:#22B0DD">

            <div id="top">{{prenomUser}} {{nomUser}} </div>

        </div>

        <div class="row btn btn-lg mt-4 container-sm
barreDec"style="background-color:#22B0DD">

```

```

        <br><br><br>

        <a [routerLink]="['/finance',id.value]" id="deconnexion"><button
id="button">

        <p>Finance </p>

    </button></a>

    <br><br><br><br>

    <a [routerLink]="['/etudiant',id.value]"
id="deconnexion"><button id="button">

        <p> Etudiant </p>

    </button></a><br><br><br><br>

    <a [routerLink]="['/formateur',id.value]"
id="deconnexion"><button id="button">

        <p> Formateur </p>

    </button></a><br><br>

    <br><br><br>

    <a href="#" id="deconnexion"><button id="button">

        <p> Gestion </p>

    </button></a>

```



```

        <button class="btn btn-info dropdown-toggle"
type="button" id="dropdownMenuButton" data-toggle="dropdown"
aria-haspopup="true" aria-expanded="false">

        <div class="dropdown-menu"
aria-labelledby="dropdownMenuButton">

            <a class="dropdown-item"
[routerLink]="['/modification',id.value]" href="#"> Modifier un
utilisateur</a>

            <a class="dropdown-item"
[routerLink]="['/inscription',id.value]" href="#">Ajouter un
utilisateur</a>

        </div>

    </button >


    <br><br><br><br><br><br>

    <a [routerLink]="['/login']" class="Deconnexion">

    <p> Deconnexion
</p>

    </a>

    <br><br><br><br>

</div>

</div>

<br><br><br>

```

```

<div class="col-md-9" style=" margin-top: 85px;">

    <div class="container">

<table class="table mt-4 table-striped">

    <thead>

        <tr>

            <th scope="col">Nom</th>

            <th scope="col">Prenom</th>

            <th scope="col">Role</th>

            <th scope="col">Email</th>

            <th scope="col-2">Action</th>

            <th></th>

        </tr>

    </thead>

    <tbody>

        <tr class="text" *ngFor="let user of allUser">

            <td>{{user.Nom}}</td>

            <td>{{user.Prenom}}</td>

            <td>{{user.role}}</td>

            <td>{{user.email}}</td>

```

```

        <td><i title="Modifier cet utilisateur" class="fas
fa-edit fa-lg text-success"
[routerLink]="['/modification/',user.id]"></i></td>

        <td><i title="Supprimer cet utilisateur"
(click)="Supprimer(user)" class="far fa-trash-alt
text-danger"></i></td>

    </tr>

</tbody>

</table>

</div>

</div>

</div>

<div>

    <p id="info">

        Si vous voulez plus d'information rendez vous sur ce lien:<a

            href="https://www.blacmi.toutachatsn.com/realisations/">
Cliquez-ici</a>

    </p>

</div>

<div class="jumbotron text-center" id="header">

    <div>

        <a href="#">

            <p id="copyright"> hakassa@copyright</p>

```

```

    </a>

  </div>

</div>

</div>

</body>

</html>

```

## *user Controle*

```

// TODO vérifier la pseudo longueur, regex de messagerie, mot de passe ect
models.User.findOne({
  attributes: ['email'],
  where: {email:email }
})
.then(function(userFound){
  if (!userFound){
    if (!EMAIL_REGEX.test(email)){
      return res.status(400).json({'error':'email non valide'})
    }
  }
  bcrypt.hash(password,5,function(err , bryptedPassword){
    // nouvel utilisateur
    var newUser = models.User.create({
      Nom:Nom,
      Prenom:Prenom,
      email:email,
      // tel:tel,
      role:role,
      password: bryptedPassword,
      isAdmin:0
    }).then(function(newUser){
      return res.status(201).json({
        | newUser
      })
    }).catch(function(err){
      return res.status(500).json({'error': 'impossible ajouter un utilisateur'})
    })
  })
  else{
    return res.status(409).json({'error':'utilisateur existe deja'});
  }
}).catch(function(err){
  return res.status(500).json({'error':'impossible de vérifier utilisateur'})
})
},
//Connexion

```

**vérifier l' email.**

```

// TODO verify mail regex & password length
models.User.findOne({
  where: {email: email}
})
.then(function(userFound){
  if(userFound){
    bcrypt.compare(password, userFound.password, function(errBycrypt, resBycrypt){
      if(resBycrypt){
        models.User.findAll().then(function(tasks){
          console.log(tasks);
          let userInfo = {
            'userId': userFound.id,
            'role': userFound.role,
            'prenom': userFound.Prenom,
            'nom': userFound.Nom,
            'token': jwtUtils.generateTokenForUser(userFound),
          }
          return res.status(200).json({userInfo, tasks});
        })
      } else {
        return res.status(403).json({ "error": "invalid password" })
      }
    })
  } else {
    return res.status(404).json({ 'error': 'user not exist in DB' })
  }
})
.catch(function(err){
  return res.status(500).json({ 'error': 'unable to verify user' })
})
},

```

```

// supprimer utilisateur
supp: function(req, res) {
  id = req.body.delId.id
  console.log(id)
  models.User.destroy({
    where: {id}
  }).then(() => {
    res.status(200).json({
      message: "suppression effective"
    })
  })
},
// modification profil
modifier: (req, res) => {
  data = req.body
  console.log(data)
  Prenom = data.Prenom
  Nom = data.Nom
  email = data.email
  role = data.role
  models.User.update(
    {
      Prenom: Prenom,
      Nom: Nom,
      role: role,
    },
    { where: { email } }
  )
  res.status(200).json({
    message: "modification effective"
  })
},
oublier: (req, res) => {
  var email
  email = req.body.email

  console.log(email)
  function entierAlea(min, max) {
    return Math.floor(Math.random() * (max - min + 1)) + min;
  }
  var password = 'user' + entierAlea(1, 1000);
  sendMail(email, password)
  console.log(password)
  bcrypt.genSalt(10, function(err, salt) {
    bcrypt.hash(password, salt, (err, hash) => {
      // create record
    })
  })
}

```

## Envoi de mail

```
44 //Envoie du mot de passe par mail
45 sendMail = (email, password)=>{
46     const transporter = nodemailer.createTransport({
47         service:"gmail",
48         auth:{
49             user: "ghostland95@gmail.com",
50             pass: "ALxamdoulilAH"
51         }
52     });
53     const options = {
54         from:"manemorthiampo@gmail.com",
55         to:email,
56         subject: email,
57         text:`pass:${password}`,
58         password:"Seneg@160"
59     }
60     transporter.sendMail(options, function(err,info){
61         if(err){
62             console.log(e-rr);
63             return;
64         }
65         console.log( "Sent:" + info.response);
66     })
67 }
```

## VI- lien du Projet

<https://github.com/thiampo/Apphakasa.git>