

TP : Réalisation d'une application Front-end avec React en communiquant avec une API avec Rest et JWT

Architecture des composants d'entreprise

## Table des matières

1. Objectif du TP.....	2
2. Prérequis .....	2
3. Développement de l'application .....	2
1. Créer un nouveau projet.....	2
2. Installer les dépendances bootstrap, axios, react-router-dom, react-validation, validator .....	3
3. Configurer le Router dans index.js .....	3
4. Créer le service AuthService .....	4
5. Créer le service axiosConfig.js .....	5
6. Créer le service CustomersService .....	6
7. Créer le composant NavBar .....	6
8. Créer les composant Home .....	11
9. Ajouter une image dans votre application .....	11
10. Créer le composant Login .....	11
11. Créer le composant Register .....	13
12. Créer le composant Profile .....	17
13. Créer le composant CustomerList .....	18
14. Créer le composant CustomerComponent .....	19
15. Créer le composant BankAccount .....	22
16. Créer le composant WirerTransfert .....	23
17. Ajouter les routes dans Apps.js .....	23
18. Ajouter les styles dans Apps.css .....	24
4. Tester votre application.....	25
1. UC n°1 : se connecter avec le compte agentguichet.....	26
2. UC n°2 : se connecter avec le compte agentguichet2.....	27
3. UC n°3 : se connecter avec le compte client.....	28
4. UC n°3 : se connecter avec le compte admin.....	29

## 1. Objectif du TP

Réaliser une application Front-end avec React et communiquer avec une API Rest sécurisée avec JWT.

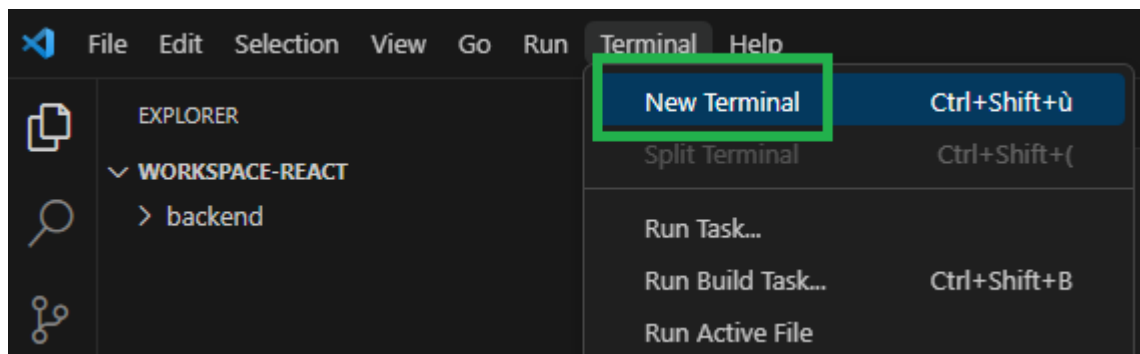
## 2. Prérequis

- Visual Studio Code (VSD);
- NODE JS ;
- Une connexion Internet pour permettre à NPM de télécharger les librairies javascript.
- Le serveur de la couche backend (<https://github.com/abbouformations/bank-service-multi-connecteur-jwt.git>) doit être démarré pour pouvoir tester par la suite votre application.

## 3. Développement de l'application

### 1. Créer un nouveau projet

- Dans VSD, ouvrir un terminal :



- Lancer la commande suivante :

```
PS C:\workspace\emsi\2022-2023\tp10reactjwt> npx create-react-app tp10reactjwt
```

- Se positionner ensuite dans le dossier *tp10reactjwt* et lancer la commande **npm start** comme montré dans l'écran suivant :

```
PS C:\workspace\emsi\2022-2023\tp10reactjwt> cd .\tp10reactjwt\
PS C:\workspace\emsi\2022-2023\tp10reactjwt\tp10reactjwt> npm start

> tp10reactjwt@0.1.0 start
> react-scripts start
```

- Vérifier que l'application a été bien démarrée.

## 2. Installer les dépendances bootstrap, axios, react-router-dom, react-validation, validator

- Lancer la commande suivante :

```
PS C:\workspace\emsi\2022-2023\tp10reactjwt\tp10reactjwt> npm i bootstrap axios react-router-dom react-validation validator
```

## 3. Configurer le Router dans index.js

- Au niveau du fichier **index.js**, entourer la balise **<App>** par la balise **<BrowserRouter>** comme expliqué ci-dessous en surbrillance :

```
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";
import reportWebVitals from "./reportWebVitals";
import { BrowserRouter } from "react-router-dom";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

- Au niveau du fichier **App.js**, importer les feuilles de style de **bootstrap** comme expliqué ci-dessous en surbrillance :

```
import logo from "./logo.svg";
import "./App.css";
import "bootstrap/dist/css/bootstrap.min.css";

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
      </header>
    </div>
  );
}
```

```

        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;

```

#### 4. Créer le service AuthService

- Créer le dossier **services** dans **src**.
- Créer ensuite le fichier **auth.service.js** dans le dossier services et développer les fonctions suivantes :

```

import axios from "axios";

const API_URL = "http://localhost:8080/auth/";

const register = (username, password, email) => {
  return axios.post(API_URL + "signup", {
    username,
    password,
    email,
  });
};

const login = (username, password) => {
  return axios
    .post(API_URL + "signin", {
      username,
      password,
    })
    .then((response) => {
      if (response.data) {
        localStorage.setItem("user", JSON.stringify(response.data));
      }

      return response.data;
    });
};

```

```
const logout = () => {
  localStorage.removeItem("user");
};

const getCurrentUser = () => {
  return JSON.parse(localStorage.getItem("user"));
};

const AuthService = {
  register,
  login,
  logout,
  getCurrentUser,
};

export default AuthService;
```

## 5. Créer le service axiosConfig.js

- Dans le dossier **services**, créer le fichier **axiosConfig.js** suivant :

```
import axios from "axios";
import AuthService from "../auth.service";

const api = axios.create({
  baseURL: "http://localhost:8080/api/rest/customer",
});

api.interceptors.request.use(
  (config) => {
    const user = AuthService.getCurrentUser();

    if (user && user.jwtToken) {
      config.headers.Authorization = `Bearer ${user.jwtToken}`;
    }

    return config;
  },
  (error) => {
    Promise.reject(error);
  }
);

export default api;
```

## 6. Créer le service CustomersService

- Dans le dossier **services**, créer le fichier **customers.service.js** suivant :

```
import api from "../axiosConfig";

const getCustomers = () => {
  return api.get("/agent_guichet/all");
};

const createCustomer = (identityRef, firstname, lastname, username) => {
  return api.post("/agent_guichet/create", {
    identityRef,
    firstname,
    lastname,
    username,
  });
};

const updateCustomer = (identityRef, firstname, lastname, username) => {
  return api.put("/agent_guichet/update/" + identityRef, {
    firstname,
    lastname,
    username,
  });
};

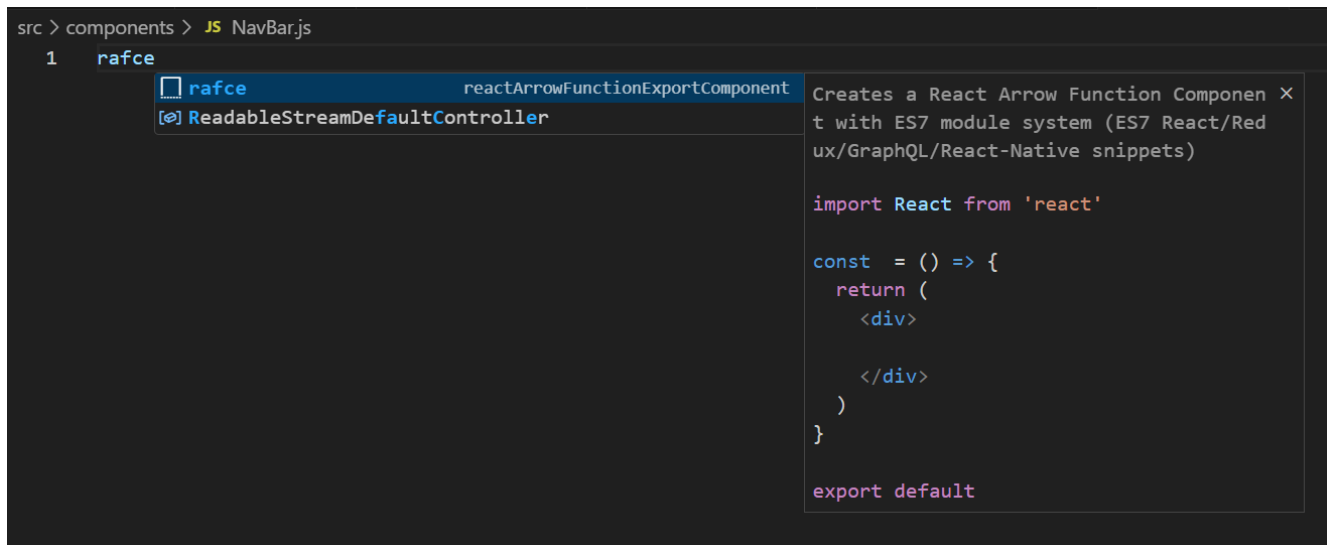
const deleteCustomer = (identityRef) => {
  return api.delete("/agent_guichet/delete/" + identityRef);
};

const CustomersService = {
  getCustomers,
  createCustomer,
  updateCustomer,
  deleteCustomer,
};

export default CustomersService;
```

## 7. Créer le composant NavBar

- Dans src, créer le dossier components ensuite créer le fichier NavBar.js
- Utiliser l'extension **ES7** de VSC pour générer les lignes de codes automatiquement comme expliqué ci-dessous :



- Vérifier que le squelette du code de la fonction **NavBar** a été bien généré :



- Modifier la fonction NavNar comme expliqué ci-dessous :

```
import React, { useState, useEffect } from "react";
import { Link } from "react-router-dom";
import AuthService from "../services/auth.service";

const NavBar = () => {
  const [currentUser, setCurrentUser] = useState(undefined);
  const [showClientBoard, setShowClientBoard] = useState(false);
  const [showAgentGuichetBoard, setShowAgentGuichetBoard] = useState(false);
  const [showAgentGuichetGetBoard, setShowAgentGuichetGetBoard] =
    useState(false);
  useEffect(() => {
    const user = AuthService.getCurrentUser();
```



```

    if (user) {
      setCurrentUser(user);
      setShowClientBoard(user.roles.includes("ROLE_CLIENT"));
      setShowAgentGuichetBoard(user.roles.includes("ROLE_AGENT_GUICHET"));
      setShowAgentGuichetGetBoard(
        user.roles.includes("ROLE_AGENT_GUICHET_GET")
      );
    }
  }, []);

const logOut = () => {
  AuthService.logout();
  setShowClientBoard(false);
  setShowAgentGuichetBoard(false);
  setShowAgentGuichetGetBoard(false);
  setCurrentUser(undefined);
};

return (
  <nav className="navbar navbar-expand navbar-dark bg-dark">
    <Link to={"/"} className="navbar-brand">
      Formation React 2023
    </Link>
    <div className="navbar-nav mr-auto">
      <li className="nav-item">
        <Link to={"/home"} className="nav-link">
          Home
        </Link>
      </li>

      {showAgentGuichetBoard | showAgentGuichetGetBoard && (
        <>
          <li className="nav-item">
            <Link to={"/manage_customers"} className="nav-link">
              Customers Management
            </Link>
          </li>

          <li className="nav-item">
            <Link to={"/manage_bankaccounts"} className="nav-link">
              Bank Accounts Management
            </Link>
          </li>
        </>
      )}

      {showClientBoard && (
        <>
          <li className="nav-item">

```

```

        <Link to={"/consult_account"} className="nav-link">
          My Bank Account
        </Link>
      </li>
      <li className="nav-item">
        <Link to={"/add_wirer_transfer"} className="nav-link">
          Wired Transfer
        </Link>
      </li>
    </>
  )}

  {currentUser && (
    <li className="nav-item">
      <Link to={"/profile"} className="nav-link">
        Profile
      </Link>
    </li>
  )}
</div>

{currentUser ? (
  <div className="navbar-nav ml-auto">
    <li className="nav-item">
      <Link to={"/profile"} className="nav-link">
        {currentUser.username}
      </Link>
    </li>
    <li className="nav-item">
      <a href="/login" className="nav-link" onClick={logout}>
        Logout
      </a>
    </li>
  </div>
) : (
  <div className="navbar-nav ml-auto">
    <li className="nav-item">
      <Link to={"/login"} className="nav-link">
        Login
      </Link>
    </li>

    <li className="nav-item">
      <Link to={"/register"} className="nav-link">
        Sign Up
      </Link>
    </li>
  </div>

```

```

    })
  </nav>
);
};

export default NavBar;

```

- Ajouter ensuite le composant NavBar dans le composant principal à savoir App.js :

```

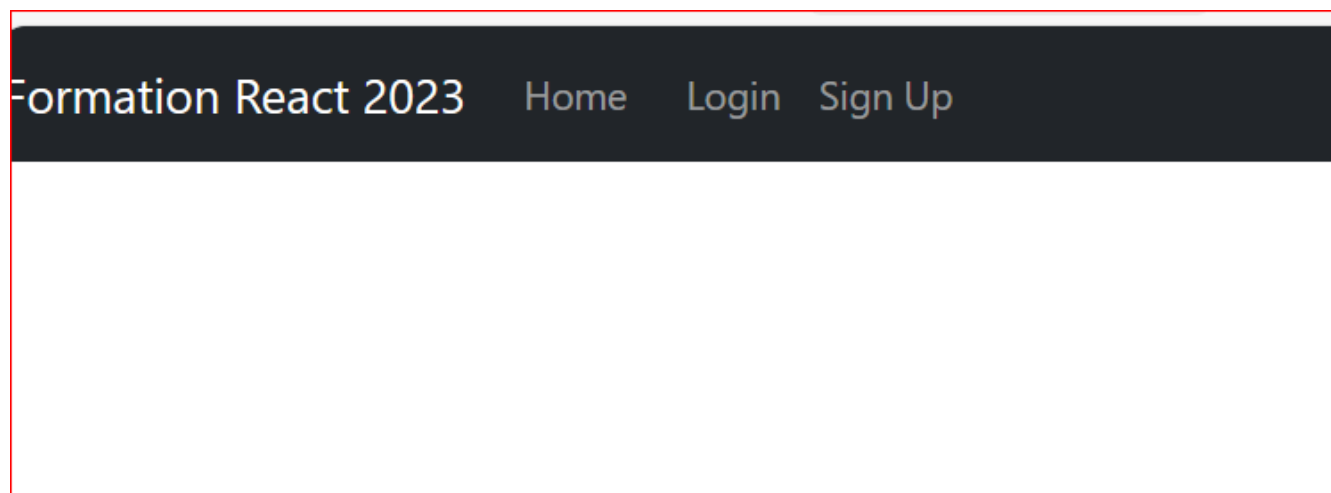
import logo from "../logo.svg";
import "../App.css";
import "bootstrap/dist/css/bootstrap.min.css";
import NavBar from "../components/NavBar";

function App() {
  return (
    <div>
      <NavBar />
    </div>
  );
}

export default App;

```

- Vérifier que le composant NavBar a été bien affiché :



## 8. Créer les composant Home

- Dans le dossier composants créer le fichier **Home.js** :

```
import React from "react";

const Home = () => {
  return <div>To be completed</div>;
};

export default Home;
```

## 9. Ajouter une image dans votre application

- Créer le dossier **images** dans **src**, ensuite copier une image dans ce dernier (le nom de l'image doit être login.png). Dans cet atelier, nous avons utilisé l'image se trouvant au lien suivant :

[https://apps.odoo.com/web/image/loempia.module/120456/icon\\_image?unique=b4d3ea9](https://apps.odoo.com/web/image/loempia.module/120456/icon_image?unique=b4d3ea9)

## 10. Créer le composant Login

- Dans le dossier composants créer le fichier **Login.js** :

```
import React, { useState, useRef } from "react";
import { useNavigate } from "react-router-dom";
import Form from "react-validation/build/form";
import Input from "react-validation/build/input";
import CheckButton from "react-validation/build/button";

import AuthService from "../services/auth.service";
import login from "../images/login.png";

const required = (value) => {
  if (!value) {
    return (
      <div className="alert alert-danger" role="alert">
        This field is required!
      </div>
    );
  }
};

const Login = () => {
  let navigate = useNavigate();

  const form = useRef();
  const checkBtn = useRef();

  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [loading, setLoading] = useState(false);
  const [message, setMessage] = useState("");

  const onChangeUsername = (e) => {
```

```

    const username = e.target.value;
    setUsername(username);
  };

  const onChangePassword = (e) => {
    const password = e.target.value;
    setPassword(password);
  };

  const handleLogin = (e) => {
    e.preventDefault();

    setMessage("");
    setLoading(true);

    form.current.validateAll();

    if (checkBtn.current.context._errors.length === 0) {
      AuthService.login(username, password).then(
        () => {
          navigate("/home");
          window.location.reload();
        },
        (error) => {
          const resMessage =
            (error.response &&
              error.response.data &&
              error.response.data.message) ||
            error.message ||
            error.toString();

          setLoading(false);
          setMessage(resMessage);
        }
      );
    } else {
      setLoading(false);
    }
  };

  return (
    <div className="col-md-12">
      <div className="card card-container">
        <img
          src={login}
          alt="profile-img"
          className="profile-img-card"
        />

        <Form onSubmit={handleLogin} ref={form}>
          <div className="form-group">
            <label htmlFor="username">Username</label>
            <Input
              type="text"
              className="form-control"
              name="username"
            />
          </div>
        </Form>
      </div>
    </div>
  );

```

```

        value={username}
        onChange={onChangeUsername}
        validations={[required]}
      />
    </div>

    <div className="form-group">
      <label htmlFor="password">Password</label>
      <Input
        type="password"
        className="form-control"
        name="password"
        value={password}
        onChange={onChangePassword}
        validations={[required]}
      />
    </div>

    <div className="form-group">
      <button className="btn btn-primary btn-block" disabled={loading}>
        {loading && (
          <span className="spinner-border spinner-border-sm"></span>
        )}
        <span>Login</span>
      </button>
    </div>

    {message && (
      <div className="form-group">
        <div className="alert alert-danger" role="alert">
          {message}
        </div>
      </div>
    )}
    <CheckButton style={{ display: "none" }} ref={checkBtn} />
  </Form>
</div>
</div>
);
};

export default Login;

```

## 11. Créer le composant Register

- Dans le dossier composants créer le fichier **Register.js**

```

import React, { useState, useRef } from "react";
import Form from "react-validation/build/form";
import Input from "react-validation/build/input";
import CheckButton from "react-validation/build/button";
import { isEmail } from "validator";

```

```

import AuthService from "../services/auth.service";
import login from "../images/login.png";

const required = (value) => {
  if (!value) {
    return (
      <div className="alert alert-danger" role="alert">
        This field is required!
      </div>
    );
  }
};

const validEmail = (value) => {
  if (!isEmail(value)) {
    return (
      <div className="alert alert-danger" role="alert">
        This is not a valid email.
      </div>
    );
  }
};

const vusername = (value) => {
  if (value.length < 3 || value.length > 20) {
    return (
      <div className="alert alert-danger" role="alert">
        The username must be between 3 and 20 characters.
      </div>
    );
  }
};

const vpassword = (value) => {
  if (value.length < 6 || value.length > 40) {
    return (
      <div className="alert alert-danger" role="alert">
        The password must be between 6 and 40 characters.
      </div>
    );
  }
};

const Register = () => {
  const form = useRef();
  const checkBtn = useRef();

  const [username, setUsername] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [successful, setSuccessful] = useState(false);
  const [message, setMessage] = useState("");

  const onChangeUsername = (e) => {
    const username = e.target.value;
  }
};

```

```

    setUsername(username);
  };

  const onChangeEmail = (e) => {
    const email = e.target.value;
    setEmail(email);
  };

  const onChangePassword = (e) => {
    const password = e.target.value;
    setPassword(password);
  };

  const handleRegister = (e) => {
    e.preventDefault();

    setMessage("");
    setSuccessful(false);

    form.current.validateAll();

    if (checkBtn.current.context._errors.length === 0) {
      AuthService.register(username, password, email).then(
        (response) => {
          setMessage(response.data);
          setSuccessful(true);
        },
        (error) => {
          const resMessage =
            (error.response &&
              error.response.data &&
              error.response.data.message) ||
            error.message ||
            error.toString();

          setMessage(resMessage);
          setSuccessful(false);
        }
      );
    }
  };

  return (
    <div className="col-md-12">
      <div className="card card-container">
        <img
          src={login}
          alt="profile-img"
          className="profile-img-card"
        />

        <Form onSubmit={handleRegister} ref={form}>
          {!successful && (
            <div>
              <div className="form-group">
                <label htmlFor="username">Username</label>

```



```

        <Input
          type="text"
          className="form-control"
          name="username"
          value={username}
          onChange={onChangeUsername}
          validations={[required, vusername]}
        />
      </div>

      <div className="form-group">
        <label htmlFor="email">Email</label>
        <Input
          type="text"
          className="form-control"
          name="email"
          value={email}
          onChange={onChangeEmail}
          validations={[required, validEmail]}
        />
      </div>

      <div className="form-group">
        <label htmlFor="password">Password</label>
        <Input
          type="password"
          className="form-control"
          name="password"
          value={password}
          onChange={onChangePassword}
          validations={[required, vpassword]}
        />
      </div>

      <div className="form-group">
        <button className="btn btn-primary btn-block">Sign Up</button>
      </div>
    </div>
  )}

  {message && (
    <div className="form-group">
      <div
        className={
          successful ? "alert alert-success" : "alert alert-danger"
        }
        role="alert"
      >
        {message}
      </div>
    </div>
  )}
  <CheckButton style={{ display: "none" }} ref={checkBtn} />
</Form>
</div>
</div>

```

```
);
};

export default Register;
```

## 12. Créer le composant Profile

- Dans le dossier composants créer le fichier **Profile.js** :

```
import React, { useState, useEffect } from "react";
import AuthService from "../services/auth.service";

const Profile = () => {
  const [jwtToken, setJwtToken] = useState("");
  const [username, setUsername] = useState("");
  const [roles, setRoles] = useState("");

  useEffect(() => {
    const user = AuthService.getCurrentUser();
    if (user) {
      setJwtToken(user.jwtToken);
      setUsername(user.username);
      setRoles(user.roles);
    }
  }, []);

  return (
    <div className="container">
      <header className="jumbotron">
        <h3>Username : {username}</h3>
        <br />
        <h3>Token : {jwtToken}</h3>
        <br />
        <h3>Roles list : </h3>

        <table class="table">
          <thead class="thead-dark">
            <tr>
              <th scope="col">Role name</th>
            </tr>
          </thead>
          <tbody>
            {roles instanceof Array &&
              roles.map((role, index) => {
                return (
                  <tr>
                    <td>{role}</td>
                  </tr>
                );
              })}
          </tbody>
        </table>
      </header>
    </div>
  );
}
```

```

        </table>
      </header>
    </div>
  );
};

export default Profile;

```

### 13. Créer le composant CustomerList

- Dans le dossier composants créer le fichier **CustomerList.js** :

```

import React from "react";
import { useState, useEffect } from "react";
import AuthService from "../services/auth.service";

const CustomerList = ({ customers, editCustomer, deleteCustomer }) => {
  const [showAgentGuichetBoard, setShowAgentGuichetBoard] = useState(false);

  useEffect(() => {
    const user = AuthService.getCurrentUser();

    if (user) {
      setShowAgentGuichetBoard(user.roles.includes("ROLE_AGENT_GUICHET"));
    }
  }, []);

  return (
    <table className="table table-hover mt-3" align="center">
      <thead className="thead-light">
        <tr>
          <th scope="col">N°</th>
          <th scope="col">Firstname</th>
          <th scope="col">Lastname</th>
          <th scope="col">Identity Ref</th>
          <th scope="col">Username</th>
          {showAgentGuichetBoard && <th scope="col">Option</th>}
        </tr>
      </thead>
      {customers.map((customer, index) => {
        return (
          <tbody key={customer.id}>
            <tr>
              <td>{customer.id}</td>
              <td>{customer.firstname}</td>
              <td>{customer.lastname}</td>
              <td>{customer.identityRef}</td>
              <td>{customer.username}</td>
              {showAgentGuichetBoard && (
                <td>
                  <button
                    type="button"
                    className="btn btn-warning"
                    onClick={() => editCustomer(customer)}

```

```

                Edit
            </button>
            <button
                type="button"
                className="btn btn-danger mx-2"
                onClick={() => deleteCustomer(customer.identityRef)}
            >
                Delete
            </button>
        </td>
    )}
</tr>
</tbody>
);
}}
</table>
);
};

export default CustomerList;

```

#### 14. Créer le composant CustomerComponent

- Dans le dossier components créer le fichier **CustomerComponent.js** :

```

import React, { useState, useEffect } from "react";
import CustomerList from "../CustomerList";
import CustomersService from "../services/customers.service";
import AuthService from "../services/auth.service";

const CustomerComponent = () => {
    /* state definition */
    const [id, setId] = useState("");
    const [identityRef, setIdentityRef] = useState("");
    const [firstname, setFirstname] = useState("");
    const [lastname, setLastname] = useState("");
    const [username, setUsername] = useState("");
    const [messageInfo, setMessageInfo] = useState("");
    const [messageError, setMessageError] = useState("");
    const [customers, setCustomers] = useState([]);

    const [showAgentGuichetBoard, setShowAgentGuichetBoard] = useState(false);

    async function save(event) {
        event.preventDefault();
        if (id) {
            await CustomersService.updateCustomer(
                identityRef,
                firstname,
                lastname,
                username
            ).then((result) => setMessageInfo("Customer updated with success"));
        } else {
            await CustomersService.createCustomer(

```

```

        identityRef,
        firstname,
        lastname,
        username
    ).then((result) => setMessageInfo("Customer added with success"));
}
// reset state
setId("");
setFirstname("");
setLastname("");
setIdentityRef("");
setUsername("");
loadCustomers();
setMessageError("");
}

async function editCustomer(customers) {
    setFirstname(customers.firstname);
    setLastname(customers.lastname);
    setIdentityRef(customers.identityRef);
    setUsername(customers.username);
    setId(customers.id);
    setMessageError("");
    setMessageInfo("");
}

async function deleteCustomer(id) {
    setMessageError("");
    setMessageInfo("");
    await CustomersService.deleteCustomer(id)
        .then((result) => {
            setMessageInfo(result.data);
        })
        .catch((e) => {
            console.log(e);
            setMessageError(e.response.data.message);
        });
    loadCustomers();
}

/* end handlers */

/* jsx */

useEffect(() => {
    const user = AuthService.getCurrentUser();

    if (user) {
        setShowAgentGuichetBoard(user.roles.includes("ROLE_AGENT_GUICHET"));
    }

    if (user) (async () => await loadCustomers())();
}, []);

async function loadCustomers() {
    await CustomersService.getCustomers()

```

```

        .then((result) => {
            setCustomers(result.data);
        })
        .catch((e) => {
            console.log(e);
            //setMessageError(e.message);
            setMessageError(e.response.data.details);
        });
    }

    return (
        <div className="container mt-4">
            <div className="container">
                {messageError && (
                    <div className="alert alert-danger" role="alert">
                        {messageError}
                    </div>
                )}
                {messageInfo && (
                    <div className="alert alert-success" role="alert">
                        {messageInfo}
                    </div>
                )}
            </div>

            {showAgentGuichetBoard && (
                <form>
                    <div className="form-group my-2">
                        <input
                            hidden
                            type="text"
                            className="form-control"
                            value={id}
                            onChange={(e) => setId(e.target.value)}
                        />
                        <label>Lastname</label>
                        <input
                            type="text"
                            className="form-control"
                            value={lastname}
                            onChange={(e) => setLastname(e.target.value)}
                        />
                    </div>

                    <div className="form-group mb-2">
                        <label>Firstname</label>
                        <input
                            type="text"
                            className="form-control"
                            value={firstname}
                            onChange={(e) => setFirstname(e.target.value)}
                        />
                    </div>

                    <div className="row">

```

```

        <div className="col-4">
          <label>Identity Ref</label>
          <input
            type="text"
            className="form-control"
            value={identityRef}
            onChange={(e) => setIdentityRef(e.target.value)}
          />
        </div>
      </div>

      <div className="row">
        <div className="col-4">
          <label>Username</label>
          <input
            type="text"
            className="form-control"
            value={username}
            placeholder="Customers"
            onChange={(e) => setUsername(e.target.value)}
          />
        </div>
      </div>

      <div>
        <button className="btn btn-primary m-4" onClick={save}>
          Save
        </button>
      </div>
    </form>
  )}
  <CustomerList
    customers={customers}
    editCustomer={editCustomer}
    deleteCustomer={deleteCustomer}
  />
</div>
);
};

export default CustomerComponent;

```

## 15. Créer le composant BankAccount

- Dans le dossier components créer le fichier **BankAccount.js** :

```

import React from "react";

const BankAccount = () => {
  return <div>To be completed</div>;
};

export default BankAccount;

```

## 16. Créer le composant WirerTransfert

- Dans le dossier composants créer le fichier **WirerTransfert.js** :

```
import React from "react";

const WirerTransfert = () => {
  return <div>To be completed</div>;
};

export default WirerTransfert;
```

## 17. Ajouter les routes dans Apps.js

- Dans le dossier composants, créer le fichier **routes.application.js** :

```
import React from "react";
import Home from "../Home";
import Login from "../Login";
import Register from "../Register";
import Profile from "../Profile";
import CustomerComponent from "../CustomerComponent";
import BankAccount from "../BankAccount";
import WirerTransfert from "../WirerTransfert";
import { Route, Routes } from "react-router-dom";

const RoutesApplications = () => {
  return (
    <div className="container mt-3">
      <Routes>
        <Route path="/" element={<Login />} />
        <Route path="/home" element={<Home />} />
        <Route path="/login" element={<Login />} />
        <Route path="/register" element={<Register />} />
        <Route path="/profile" element={<Profile />} />

        <Route path="/manage_customers" element={<CustomerComponent />} />
        <Route path="/manage_bankaccounts" element={<BankAccount />} />
        <Route path="/add_wirer_transfer" element={<WirerTransfert />} />
        <Route path="/consult_account" element={<BankAccount />} />
      </Routes>
    </div>
  );
};

export default RoutesApplications;
```



- Ajouter le composant **RoutesApplication** dans **Apps.js** comme expliqué ci-dessous :

```
import './App.css';
import 'bootstrap/dist/css/bootstrap.min.css';
import NavBar from './components/NavBar';
import RoutesApplications from './components/routes.applications';

function App() {
  return (
    <div>
      <NavBar />
      <RoutesApplications />
    </div>
  );
}

export default App;
```

## 18. Ajouter les styles dans Apps.css

- Modifier la feuille de style du composant principal **Apps.css** comme expliqué ci-dessous :

```
label {
  display: block;
  margin-top: 10px;
}

.card-container.card {
  max-width: 350px !important;
  padding: 40px 40px;
}

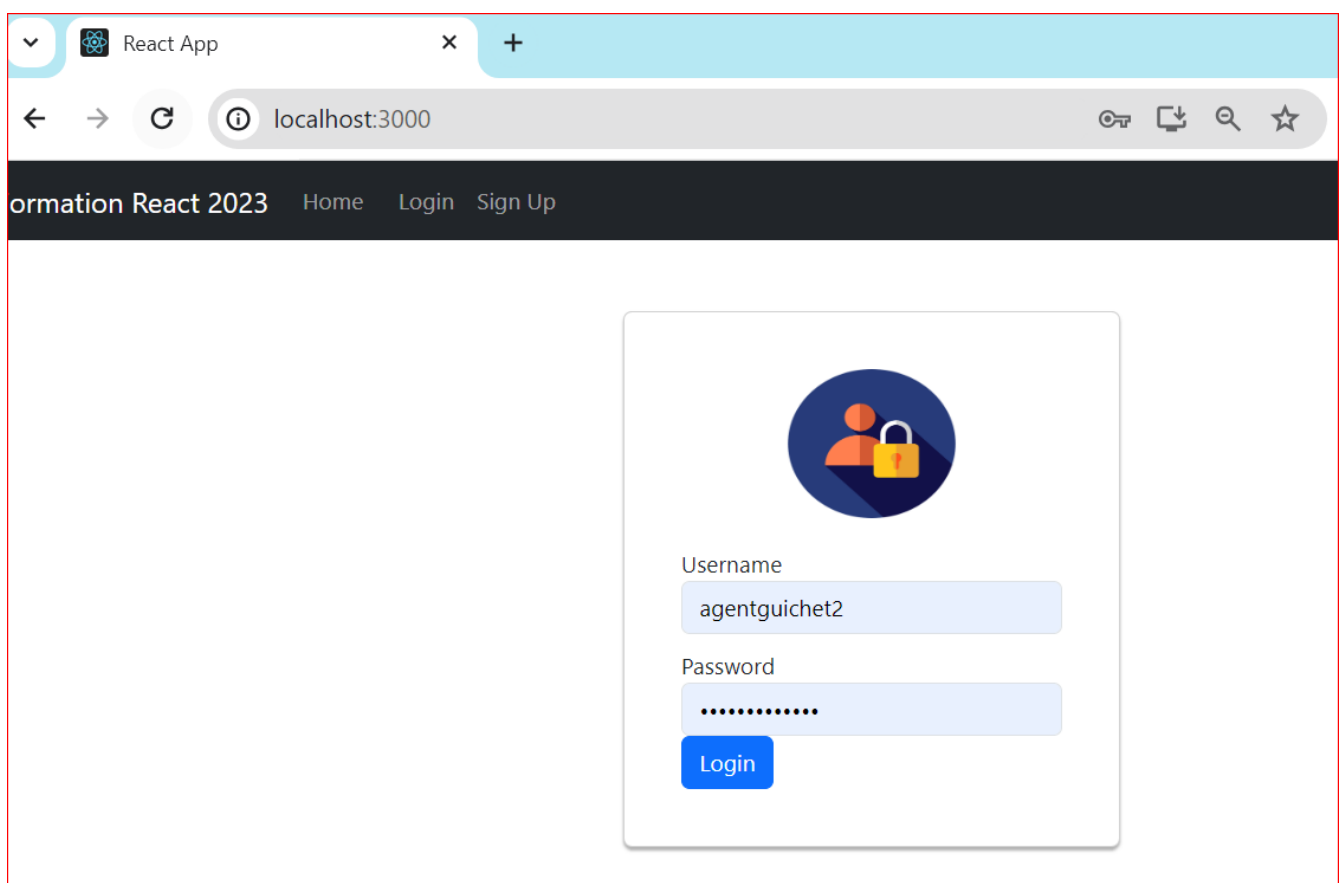
.card {
  background-color: #f7f7f7;
  padding: 20px 25px 30px;
  margin: 0 auto 25px;
  margin-top: 50px;
  -moz-border-radius: 2px;
  -webkit-border-radius: 2px;
  border-radius: 2px;
  -moz-box-shadow: 0px 2px 2px rgba(0, 0, 0, 0.3);
  -webkit-box-shadow: 0px 2px 2px rgba(0, 0, 0, 0.3);
  box-shadow: 0px 2px 2px rgba(0, 0, 0, 0.3);
}

.profile-img-card {
```

```
width: 225px;
height: 105px;
margin: 0 auto 10px;
display: block;
-moz-border-radius: 50%;
-webkit-border-radius: 50%;
border-radius: 50%;
}
```

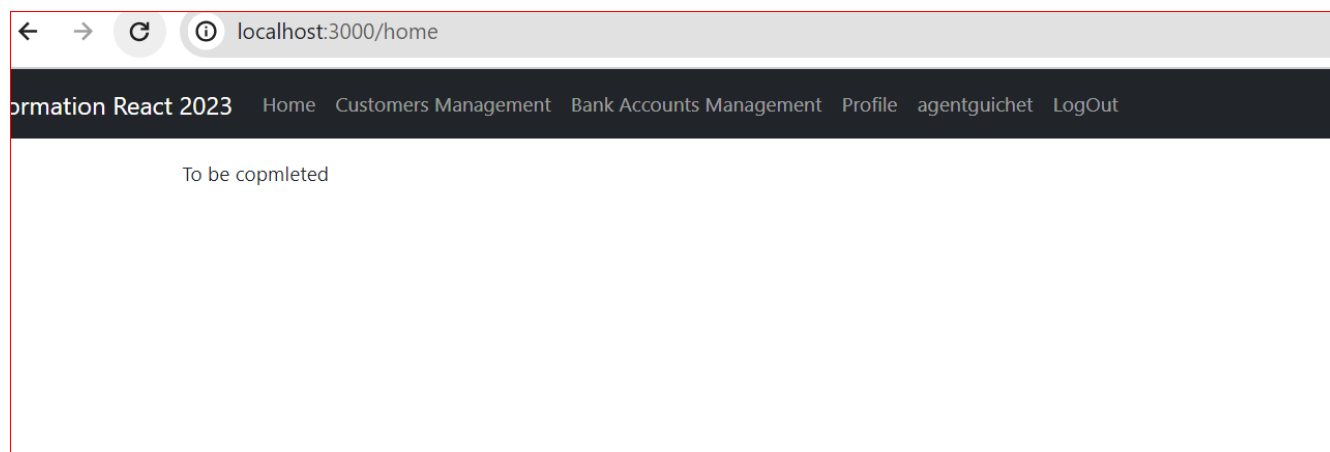
#### 4. Tester votre application

- Commencer par démarrer la couche backend en lançant la commande : ***java -jar backend.jar***. Ce JAR est celui du build de l'atelier développé dans le TP n°9. Le code source de ce dernier est disponible sur GITHUB moyennant le lien suivant : <https://github.com/abbouformations/bank-service-multi-connecteur-jwt.git>  
Vous pouvez cloner le projet avec la commande : **git clone LE\_LIEN\_GIT** et ensuite au niveau d'IntelliJ vous pouvez lancer la commande **clean install** pour générer le JAR.
- Une fois le serveur backend est lancé, lancer l'URL <http://localhost:3000> :



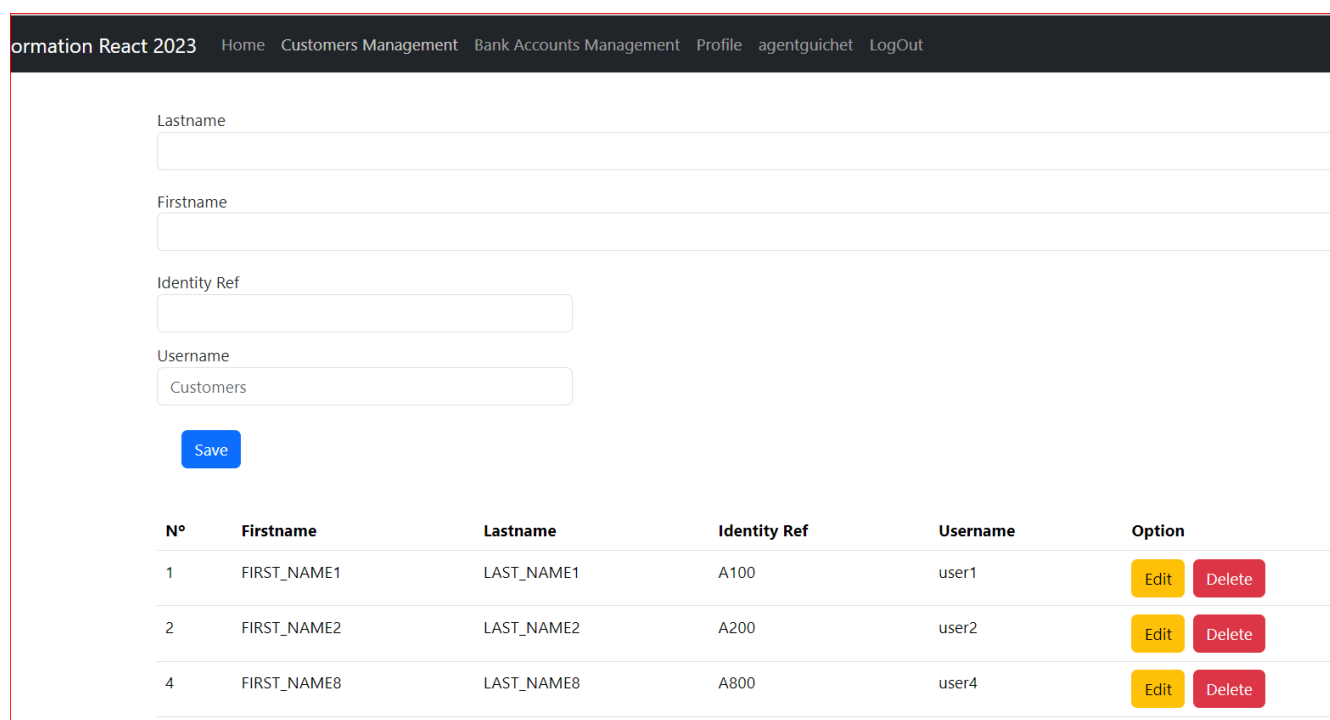
## 1. UC n°1 : se connecter avec le compte **agentguichet**

- Au niveau de la couche backend, l'utilisateur agentguichet dispose de rôle ROLE\_AGENT\_GUICHET.
- Entrer le compte suivant : username=[agentguichet, password= agentguichet] :



The screenshot shows a web browser at localhost:3000/home. The navigation bar includes 'ormation React 2023', 'Home', 'Customers Management', 'Bank Accounts Management', 'Profile', 'agentguichet', and 'LogOut'. The main content area displays the text 'To be copmleted'.

- Cliquer sur le menu « Customers Management » pour la gestion des clients :



The screenshot shows the 'Customers Management' page. It features a form with fields for 'Lastname', 'Firstname', 'Identity Ref', and 'Username' (containing 'Customers'). A blue 'Save' button is below the form. Below the form is a table with columns: N°, Firstname, Lastname, Identity Ref, Username, and Option.

N°	Firstname	Lastname	Identity Ref	Username	Option
1	FIRST_NAME1	LAST_NAME1	A100	user1	<button>Edit</button> <button>Delete</button>
2	FIRST_NAME2	LAST_NAME2	A200	user2	<button>Edit</button> <button>Delete</button>
4	FIRST_NAME8	LAST_NAME8	A800	user4	<button>Edit</button> <button>Delete</button>

- Vous pouvez consulter le profil de l'utilisateur en cliquant sur le menu **Profile** :

Usrename : agentguichet

Token :

eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZ2VudGd1aWNoZXQiLCJyb2xlcyI6WyJST0xFX0FH  
yK1KysA

Roles list :

Role name

ROLE\_AGENT\_GUICHET

GET\_ALL\_CUSTUMERS

GET\_CUSTOMER\_BY\_IDENTITY

CREATE\_CUSTOMER

UPDATE\_CUSTOMER

DELETE\_CUSTOMER

GET\_ALL\_BANK\_ACCOUNT

GET\_BANK\_ACCOUNT\_BY\_RIB

CREATE\_BANK\_ACCOUNT

## 2. UC n°2 : se connecter avec le compte agentguichet2

- Au niveau de la couche backend, l'utilisateur agentguichet2 dispose de rôle ROLE\_AGENT\_GUICHET\_GET. Ce dernier ne peut que consulter les clients et les comptes.
- Entrer le compte suivant : username=[agentguichet2, password= agentguichet2] :

To be copmleted

- Cliquer sur le menu Profile pour consulter le profile de l'utilisateur :

Usrename : agentguichet2

Token :

eyJhbGciOiJIUzUxMiJ9.eyJzdWliOiJhZ2VudGd1aWNoZXQyIiwicm9sZXNtVXoF5YQAT8Rolla7S8HEukeYwZofQArF6QKH\_AK8n5tU8x1ETjtc\_jkg

Roles list :

**Role name**

ROLE\_AGENT\_GUICHET\_GET

GET\_ALL\_CUSTUMERS

GET\_CUSTOMER\_BY\_IDENTITY

GET\_ALL\_BANK\_ACCOUNT

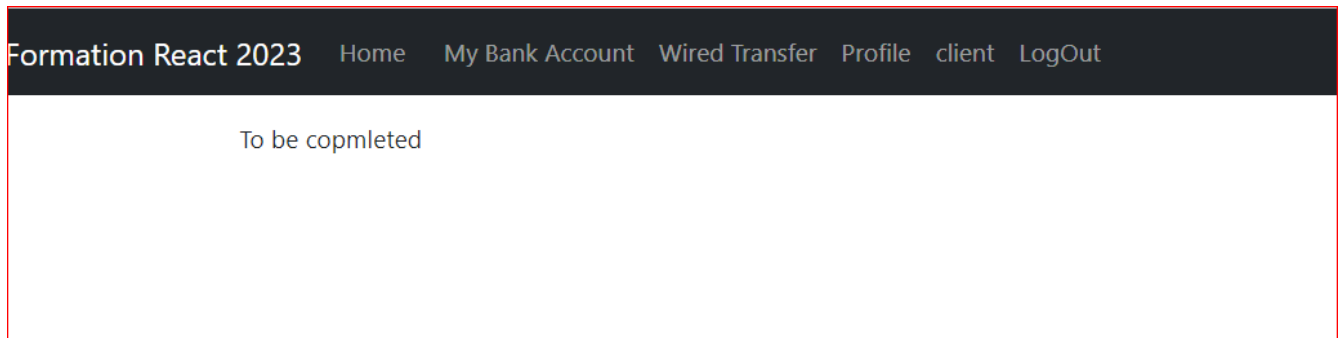
GET\_BANK\_ACCOUNT\_BY\_RIB

- Cliquer sur le menu « Customers management » et remarquer que l'utilisateur ne peut que consulter les clients :

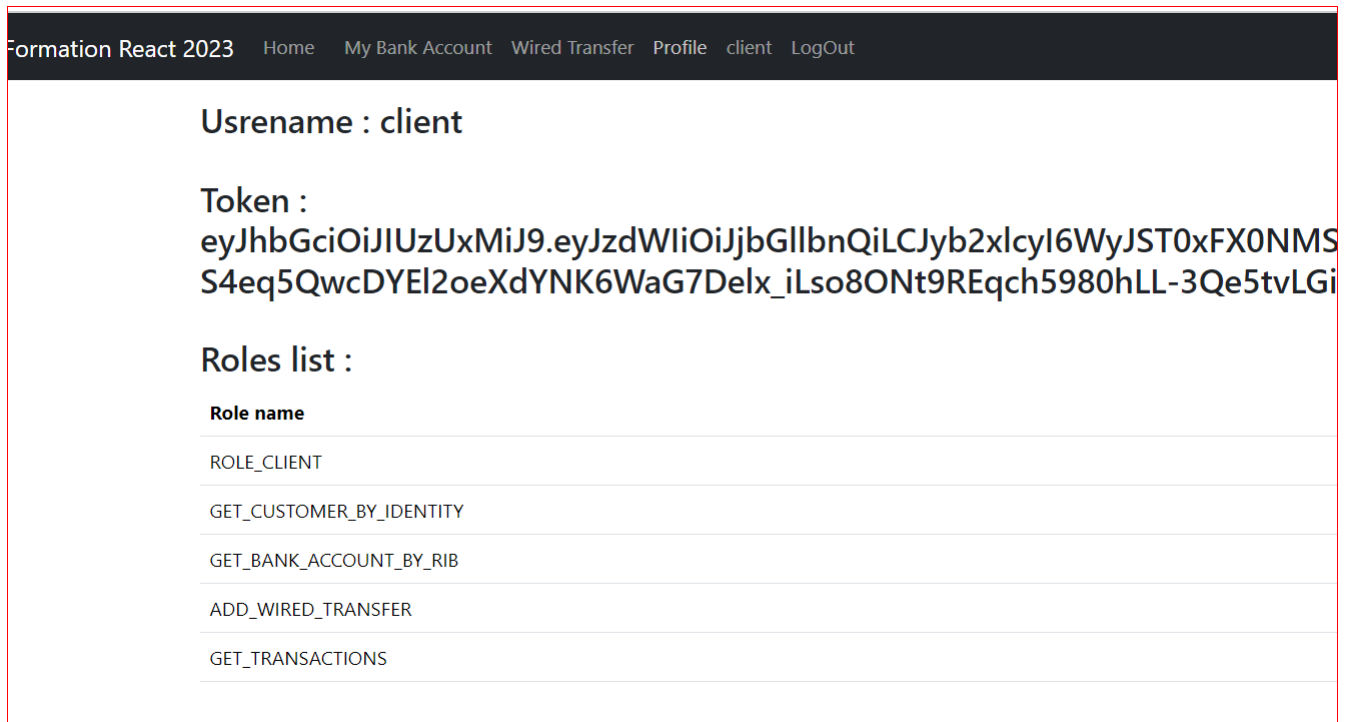
N°	Firstname	Lastname	Identity Ref	Username
1	FIRST_NAME1	LAST_NAME1	A100	user1
2	FIRST_NAME2	LAST_NAME2	A200	user2
4	FIRST_NAME8	LAST_NAME8	A800	user4
3	FIRST_NAME9	LAST_NAME9	A900	user3

### 3. UC n°3 : se connecter avec le compte **client**

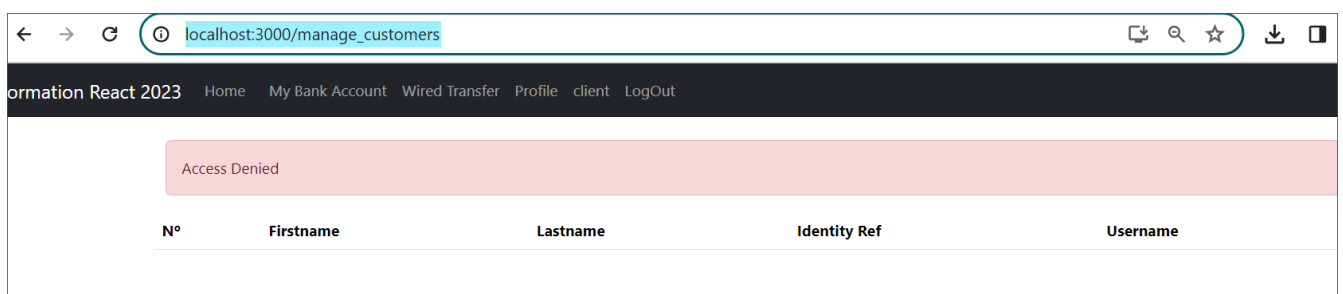
- Au niveau de la couche backend, l'utilisateur client dispose de rôle ROLE\_CLIENT. Ce dernier ne peut que consulter son compte, ses données, ses transactions et faire des virements.
- Entrer le compte suivant : username=[client, password= client] :



- Cliquer sur le menu **Profile** pour consulter son profile :



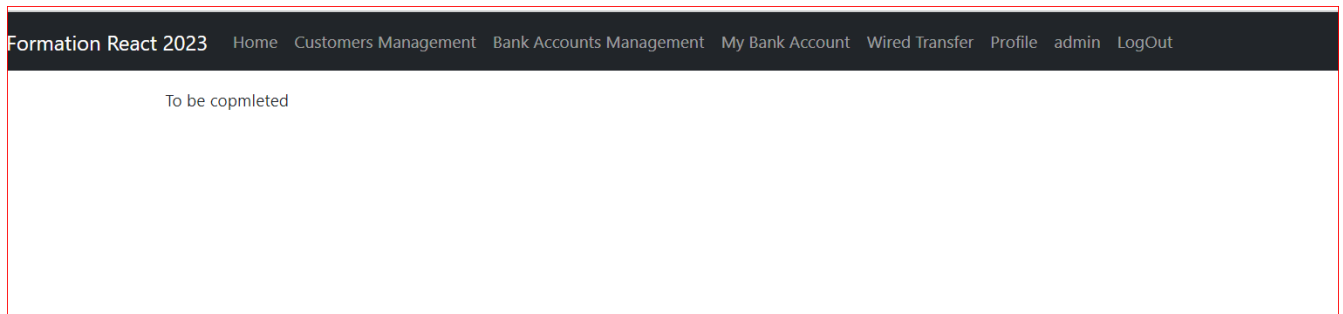
- Vous remarquez que l'application n'affiche pas les menus concernant l'agent guichet.
- Si le client essaye par exemple d'accéder au lien [http://localhost:3000/manage\\_customers](http://localhost:3000/manage_customers), l'application affiche le message suivant :



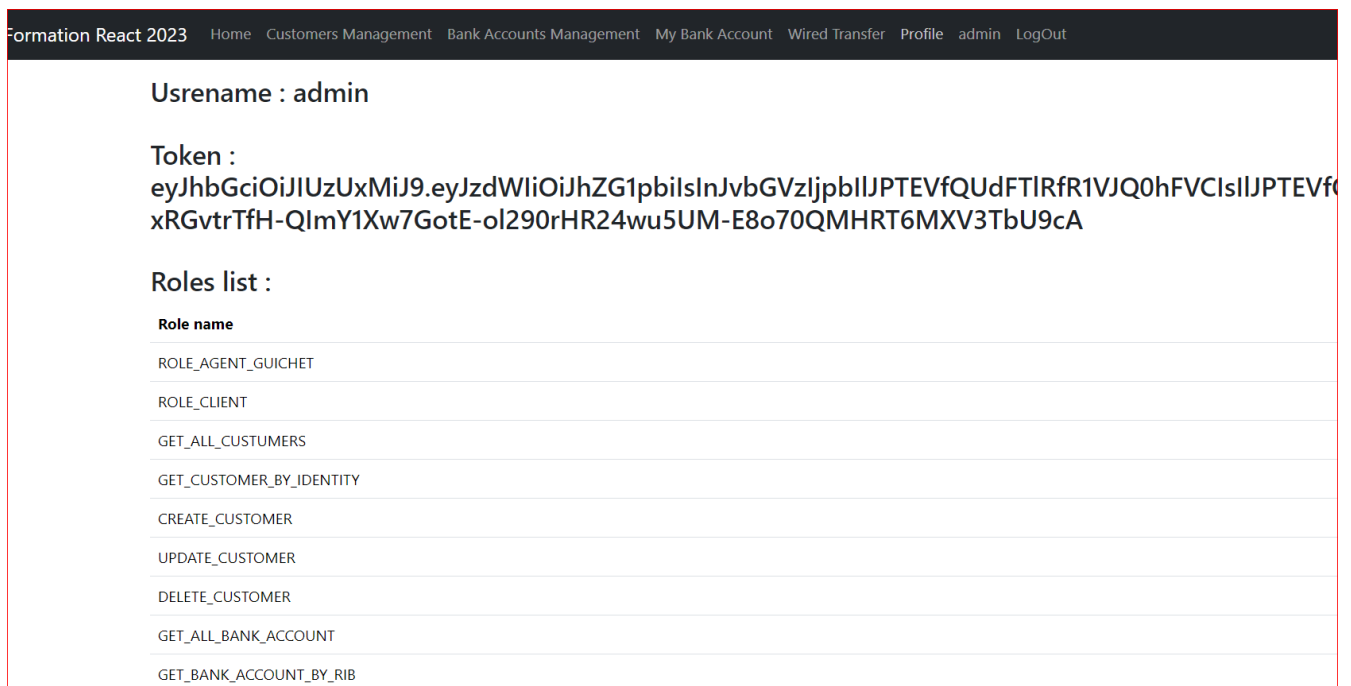
#### 4. UC n°3 : se connecter avec le compte **admin**

- Au niveau de la couche backend, l'utilisateur admin dispose des rôles suivants : ROLE\_AGENT\_GUICHET et ROLE\_CLIENT. Ce dernier dispose des toutes les autorisations.

- Entrer le compte suivant : username=[admin, password= admin] :



- Cliquer sur le menu **Profile** pour consulter son profile :



- Cliquer ensuite sur les autres menus pour tester les services.

## Conclusion

Le code source de cet atelier est disponible sur GITHUB :

<https://github.com/abbouformations/react-crud-example-axios.git>