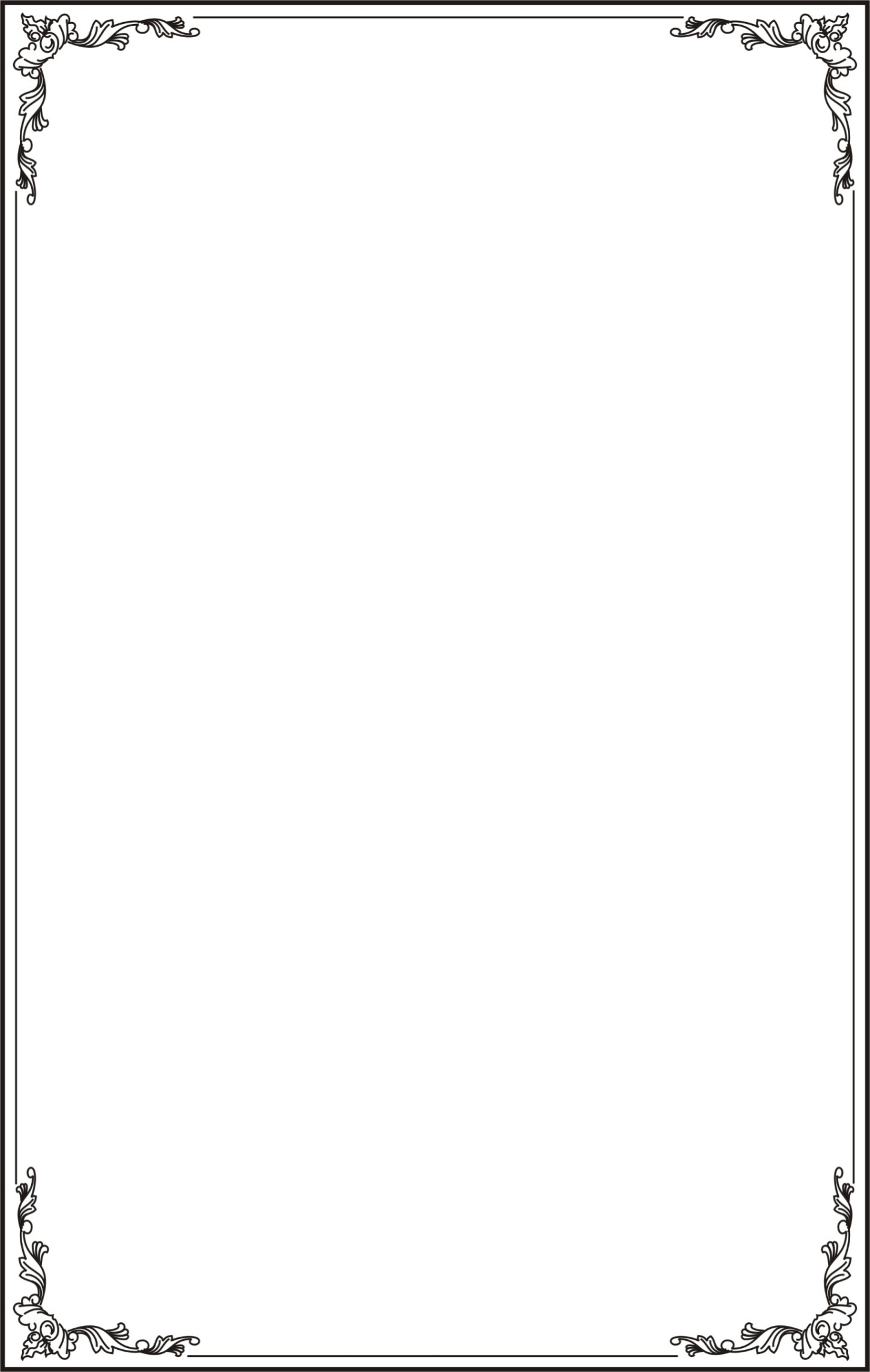
**CÔNG TY CỔ PHẦN**

**VIỄN THÔNG FPT**

****

**BÀI THU HOẠCH**

**ĐÀO TẠO TÂN BINH R&D TUẦN 7 & 8**

|  |  |
| --- | --- |
| Người hướng dẫn: | **MR. TRƯƠNG TẤN SANG** |
| Người thực hiện: | **TRẦN KHÁNH GIA UY** |
| Đơn vị công tác: | **Trung tâm giám sát và đảm bảo dịch vụ (SCC)** |
| Vị trí: | **Open Source Web Software Developer** |
| Bộ phận: | **SCC** |

***Thành phố Hồ Chí Minh, tháng 05 năm 2021***

# MỤC LỤC

[MỤC LỤC 2](#_Toc76932749)

[CHƯƠNG 1: XÂY DỰNG APP THEO UMIJS VÀ DVAJS 3](#_Toc76932750)

[1.1 Ý tưởng 3](#_Toc76932751)

[1.2 Cách thực hiện 3](#_Toc76932752)

[1.3 Source code 7](#_Toc76932753)

[1.4 Demo 7](#_Toc76932754)

[CHƯƠNG 2: XÂY DỰNG APP VỚI AUTH0 VÀ REFRESH TOKEN 8](#_Toc76932755)

[2.1 Ý tưởng 8](#_Toc76932756)

[2.2 Cách thực hiện 8](#_Toc76932757)

[2.2.1 Refresh token 8](#_Toc76932758)

[2.2.2 Auth0 11](#_Toc76932759)

[2.3 Source code 13](#_Toc76932760)

[2.4 Demo 13](#_Toc76932761)

[2.5 Hướng đi tiếp theo 14](#_Toc76932762)

# CHƯƠNG 1: XÂY DỰNG APP THEO UMIJS VÀ DVAJS

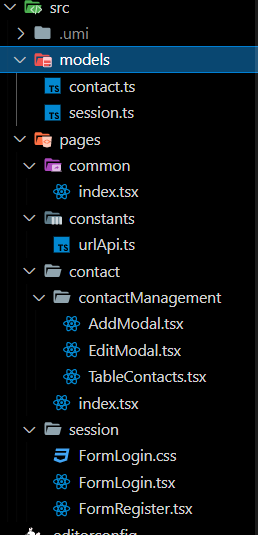
## Ý tưởng

* Em sẽ chuyển app redux lần trước theo UmiJS và quản lý redux bằng DvaJS
  1. **Cách thực hiện**
* Đầu tiên em cài đặt DvaJS và UmiJS qua lệnh terminal sau:

yarn create @umijs/umi-app

npm install dva-cli -g

* Sau đó em cấu trúc lại thư mục như sau

****

* Trong đó folder *models* chứa các file quản lý state trong redux và folder *pages* chứa các file theo feature.
* Em sẽ config route ở trong file cài đặt ***.umirc.ts*** như sau:

export default defineConfig({

  dva: {},

  nodeModulesTransform: {

    type: 'none',

  },

  routes: [

    { exact: true, path: '/', component: '@/pages/common/index' },

    { exact: true, path: '/login', component: '@/pages/session/FormLogin' },

    {

      exact: true,

      path: '/register',

      component: '@/pages/session/FormRegister',

    },

  ],

  fastRefresh: {},

});

* File ***session.ts*** ở trong folder *models* để quản lý state khi đăng kí, đăng nhập.

export default {

  namespace: 'session',

  state: {

    register: { username: '', password: '' },

    login: {

      username: '',

      password: '',

    },

    auth: window.localStorage.getItem('jwtToken'),

  },

  reducers: {

    login(state: Session, action: any) {

      return { ...state, login: action.user };

    },

    auth(state: Session, action: any) {

      return { ...state, auth: action.token };

    },

    register(state: Session, action: any) {

      return { ...state, register: action.user };

    },

  },

  effects: {

    \*handleLogin(action: any, { call, put }: { call: any; put: any }) {

      const user = action.user;

      yield axios.get(userData).then((response) => {

        for (const userInt of response.data) {

          if (user.username === userInt.username) {

            bcrypt

              .compare(user.password, userInt.password)

              .then((valid: any) => {

                if (valid) {

                  const token = userInt.token;

                  window.localStorage.setItem('jwtToken', token);

                }

              });

            break;

          }

        }

      });

      yield put({

        type: 'login',

        user: user,

      });

      yield call(delay, 500);

      yield put({

        type: 'auth',

        token: window.localStorage.getItem('jwtToken'),

      });

    },

    \*handleRegister(action: any, { put }: { put: any }) {

      const user = action.user;

      const salt = bcrypt.genSaltSync(10);

      const hassPassword = bcrypt.hashSync(user.password, salt);

      const token = jwt.sign(

        {

          username: user.username,

          password: hassPassword,

        },

        'SECRET\_TOKEN',

      );

      const data = {

        username: user.username,

        password: hassPassword,

        token: token,

      };

      yield axios

        .post(userData, data)

        .then((response) => {

          console.log(response);

        })

        .catch((err) => {

          console.log(err);

        });

      yield put({

        type: 'register',

        user: user,

      });

    },

  },

};

* Trong đó *state* để khởi tạo state ban đầu, *reducers* để lắng nghe các action mỗi khi được gọi và trả về state tương ứng, *effects* để xử lý các logic liên quan.
* Khi đó, trong file ***FormLogin.tsx*** em sẽ cài đặt để gọi action như sau :

const dispatch = useDispatch();

  const onFinish = (values: User) => {

    dispatch({ type: 'session/handleLogin', user: values });

    setTimeout(() => {

      let path = '/';

      history.push(path);

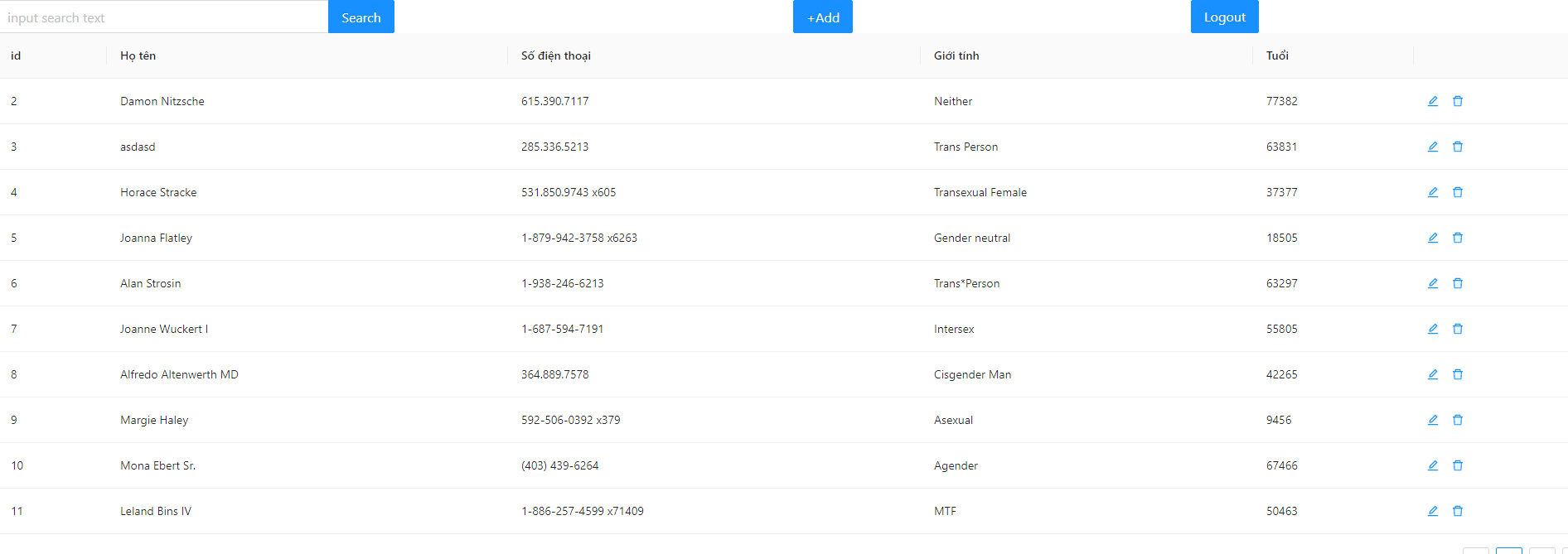
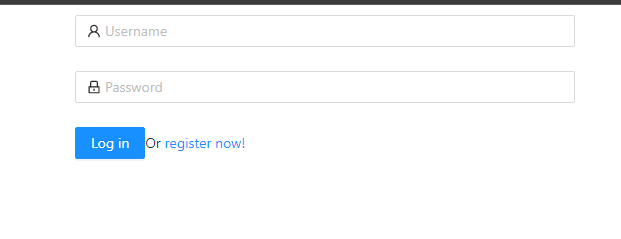
    }, 2000);

  };

* Tương tự với các feature còn lại
  1. **Source code**
* Tất cả source code nằm trong link github sau:

[internSCC/React/umijs-dva at main · uytran36/internSCC (github.com)](https://github.com/uytran36/internSCC/tree/main/React/umijs-dva)

* 1. **Demo**

****

# CHƯƠNG 2: XÂY DỰNG APP VỚI AUTH0 VÀ REFRESH TOKEN

* 1. **Ý tưởng**
  2. **Cách thực hiện**
* Em sẽ clone lại project sử dụng UmiJS và DvaJS và thêm Auth0 và refresh token trong project này
  + 1. **Refresh token**
* Trong file ***sesstion.ts***

\*handleLogin(action: any, { call, put }: { call: any; put: any }) {

const user = action.user;

  yield axios.get(userData).then((response) => {

   for (const userInt of response.data) {

     if (user.username === userInt.username) {

       bcrypt

        .compare(user.password, userInt.password)

        .then((valid: any) => {

         if (valid) {

           const token = jwt.sign(

             {

             username: user.username,

             password: userInt.password,

             },

             'SECRET\_TOKEN',

             { expiresIn: '1h' },

         );

            const refreshToken = jwt.sign(

             {

               username: user.username,

                password: userInt.password,

              },

              'SECRET\_TOKEN',

              { expiresIn: '100d' },

            );

            window.localStorage.setItem('jwtToken', token);

            window.localStorage.setItem('refreshJwtToken', refreshToken);

          }      }

        });

        break;

      }

    }

  });

  yield put({

   type: 'login',

    user: user,

  });

  yield call(delay, 500);

  yield put({

   type: 'auth',

    token: window.localStorage.getItem('jwtToken'),

  });

},

* Như vậy em đã cấp phát token khi đăng nhập và nó có hiệu lực trong 1 giờ. Ngoài ra, em cấp phát thêm refresh token để cấp phát lại token khi nó hết hiệu lực.
* Trong file chứa component App:

function App(props: any) {

  const token: any = window.localStorage.getItem('jwtToken');

  let dec: any;

  if (token !== null) {

    jwt.verify(token, 'SECRET\_TOKEN', (err: any, decoded: any) => {

      console.log(err);

      dec = decoded;

      console.log(decoded);

    });

    if (dec === undefined) {

      const refreshToken: any = window.localStorage.getItem('refreshJwtToken');

      jwt.verify(refreshToken, 'SECRET\_TOKEN', (err: any, decoded: any) => {

        console.log(err);

        dec = decoded;

      });

      const newToken = jwt.sign(

        {

          username: dec.username,

          password: dec.password,

        },

        'SECRET\_TOKEN',

        { expiresIn: '1h' },

      );

      window.localStorage.setItem('jwtToken', newToken);

    }

  }

  const { isAuthenticated, isLoading, getAccessTokenSilently, user } =

    useAuth0();

  if (isLoading) {

    return <Spin></Spin>;

  }

  getAccessTokenSilently().then((token) => {

    window.localStorage.setItem('token', token);

  });

  console.log('is authenticated');

  console.log(isAuthenticated);

  return (

    <div className="App">

      {(token !== null ) || isAuthenticated === true ? (

        <Contact />

      ) : (

        <Redirect to="/login" />

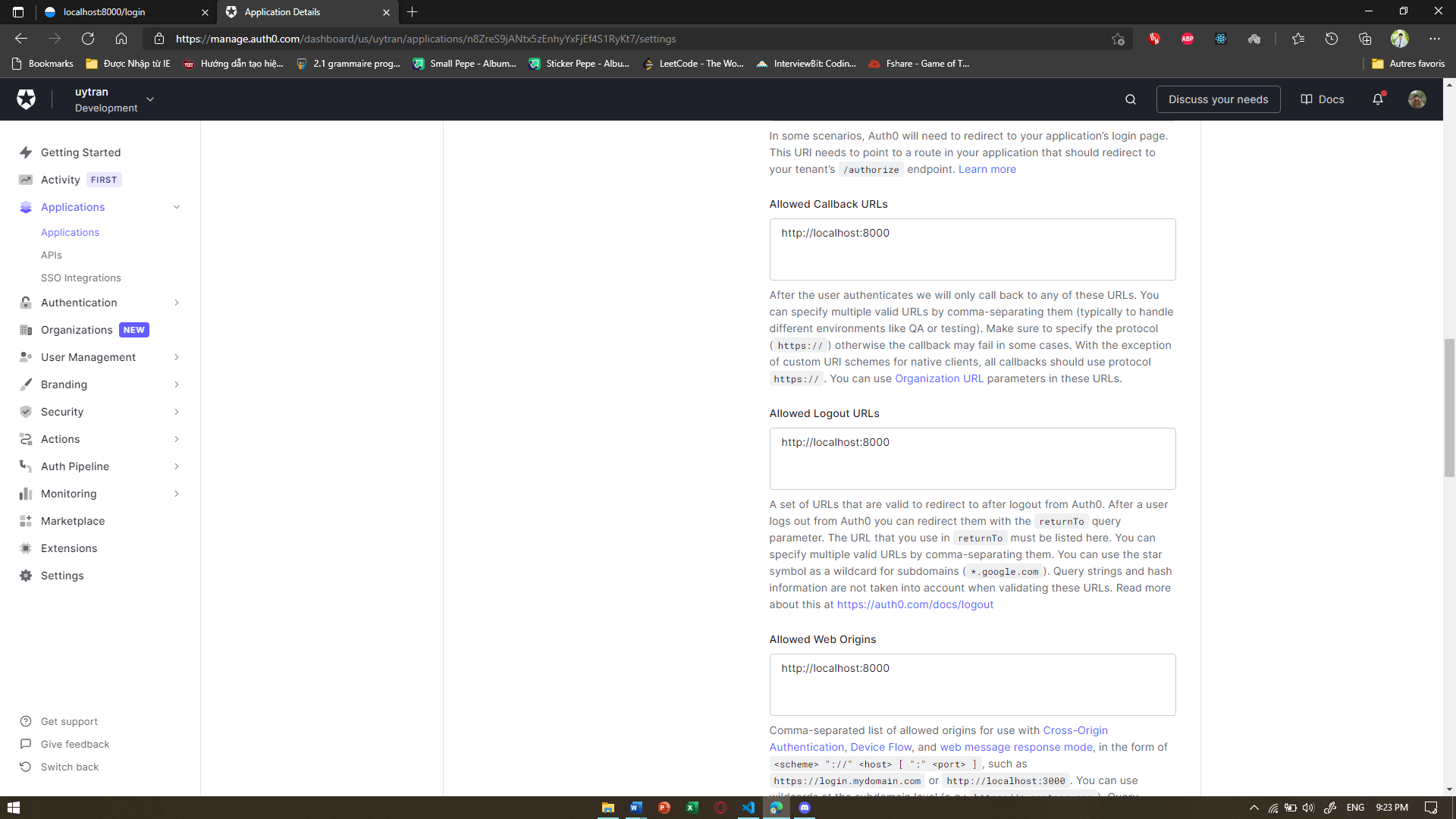
      )}

    </div>

  );

}

* Em sẽ decode jwt và nếu như nó bị hết hạn thì sẽ sử dụng refresh token để tạo token mới.
  + 1. **Auth0**
* Đầu tiên em tạo app trong Auth0 và thêm địa chỉ <http://localhost:8000> như sau:



* Để sử dụng được Auth0, em sẽ thêm folder wrapper như sau:



* Trong file ***Auth0.tsx***

import { Auth0Provider } from '@auth0/auth0-react';

export default (props: any) => {

  return (

    <Auth0Provider

      domain="uytran.us.auth0.com"

      clientId="n8ZreS9jANtx5zEnhyYxFjEf4S1RyKt7"

      redirectUri="http://localhost:8000"

      useRefreshTokens={true}

    >

      {props.children}

    </Auth0Provider>

  );

};

* Domain và clientId được lấy từ cài đặt app trong Auth0
* Sau đó em config wrapper như sau:

export default defineConfig({

  dva: {},

  nodeModulesTransform: {

    type: 'none',

  },

  routes: [

    {

      exact: true,

      path: '/',

      component: '@/pages/common/index',

      wrappers: ['@/wrappers/Auth0'],

    },

    {

      exact: true,

      path: '/login',

      component: '@/pages/session/FormLogin',

      wrappers: ['@/wrappers/Auth0'],

    },

    {

      exact: true,

      path: '/register',

      component: '@/pages/session/FormRegister',

      wrappers: ['@/wrappers/Auth0'],

    },

  ],

  fastRefresh: {},

});

* Em thêm button đăng nhập vào Auth0 trong app của mình:

const { loginWithRedirect} = useAuth0();

<button onClick={() => loginWithRedirect()}>Login with Auth0</button>

* Tương tự với button logout:

const { logout } = useAuth0();

  //when click logout button

  const onClickLogout = () => {

    window.localStorage.removeItem('jwtToken');

    window.localStorage.removeItem('refreshJwtToken');

    window.localStorage.removeItem('token');

    logout();

  };

<Button

type="primary"

  size="large"

  onClick={props.onClickLogout}

>

Logout

</Button>

* Khi login thành công bằng Auth0, em sẽ lưu token của Auth0 vào local storage

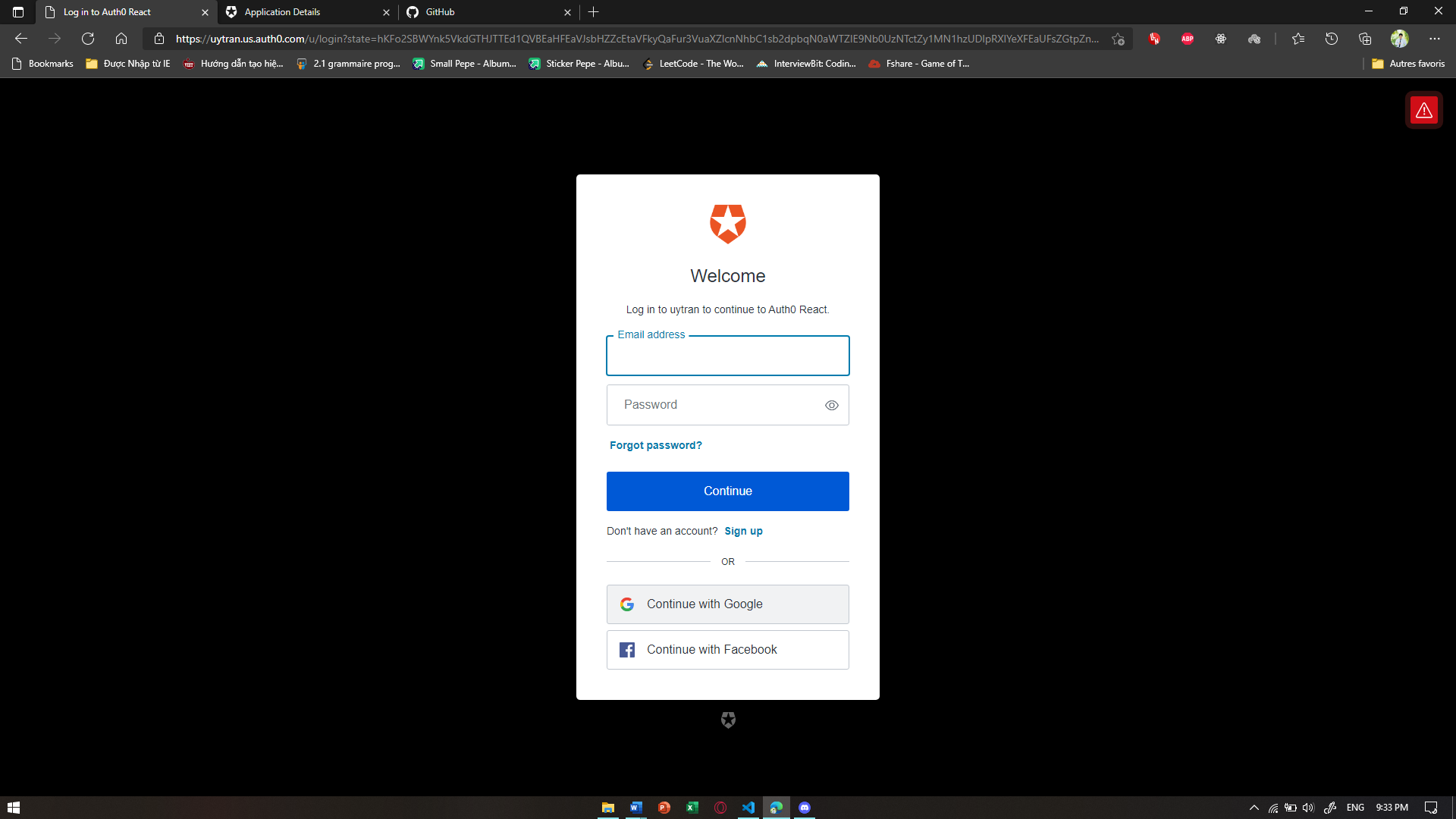
getAccessTokenSilently().then((token) => {

window.localStorage.setItem('token', token);

});

* Như vậy là em có thể sử dụng được Auth0 trong app của mình.
  1. **Source code**
* Tất cả source code nằm trong link github sau:

[internSCC/React/auth0-umijs at main · uytran36/internSCC (github.com)](https://github.com/uytran36/internSCC/tree/main/React/auth0-umijs)

* 1. **Demo**
  2. **Hướng đi tiếp theo**
* Tìm hiểu về Ant design pro và thêm một trang mới vào project Ant design pro, sử dụng bizcharts để xây dựng biểu đồ và tạo table giống như hình sau :

