**BÁO CÁO 10: XÂY DỰNG APP THEO UMIJS VÀ DVAJS**

1. **Ý tưởng**

* Em sẽ chuyển app redux lần trước theo UmiJS và quản lý redux bằng DvaJS

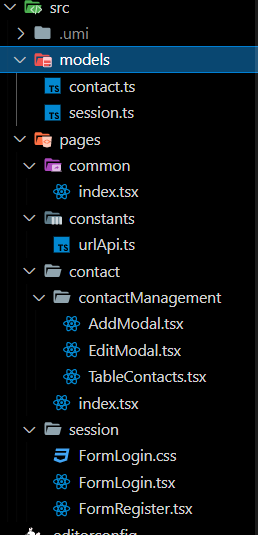
1. **Cách thực hiện**

* 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);

  };

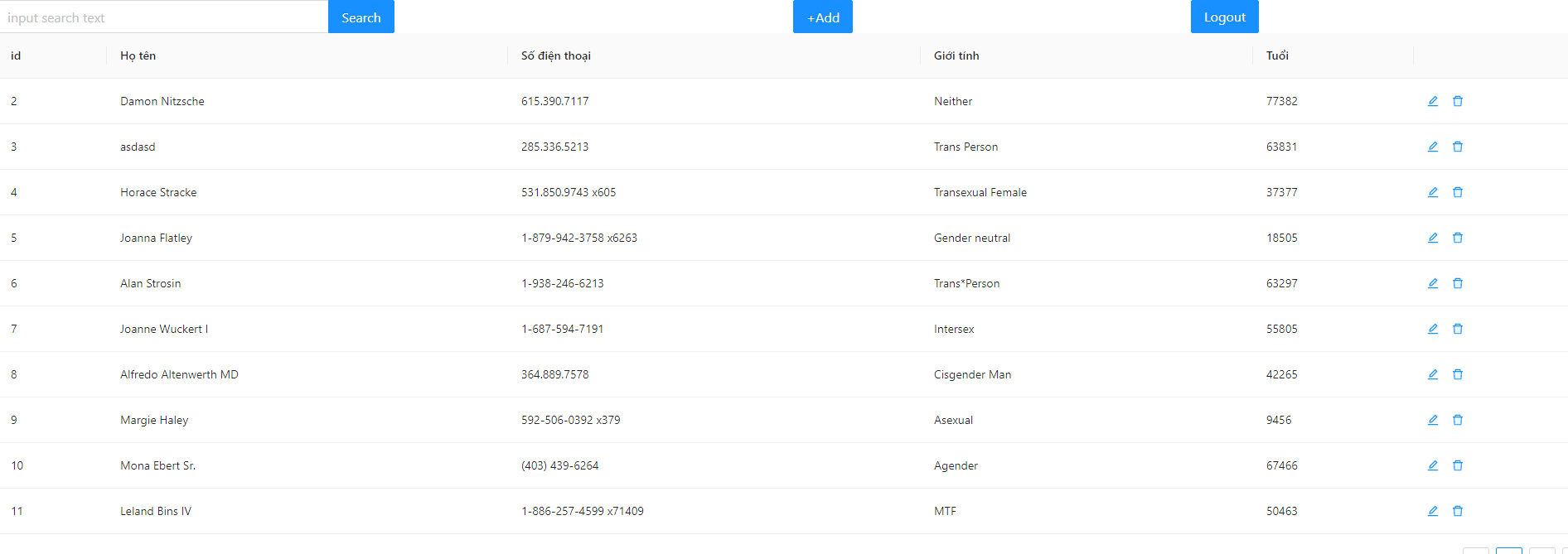
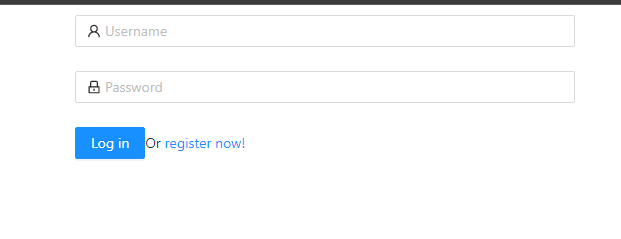
* Tương tự với các feature còn lại

1. **Source**

* 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**

****