ВИКОРИСТАННЯ АРХІТЕКТУРНОГО ПІДХОДУ FLUX ДЛЯ РОЗРОБКИ ВЕБ-ЗАСТОСУНКІВ НА ПРИКЛАДІ БІБЛІОТЕКИ REDUX

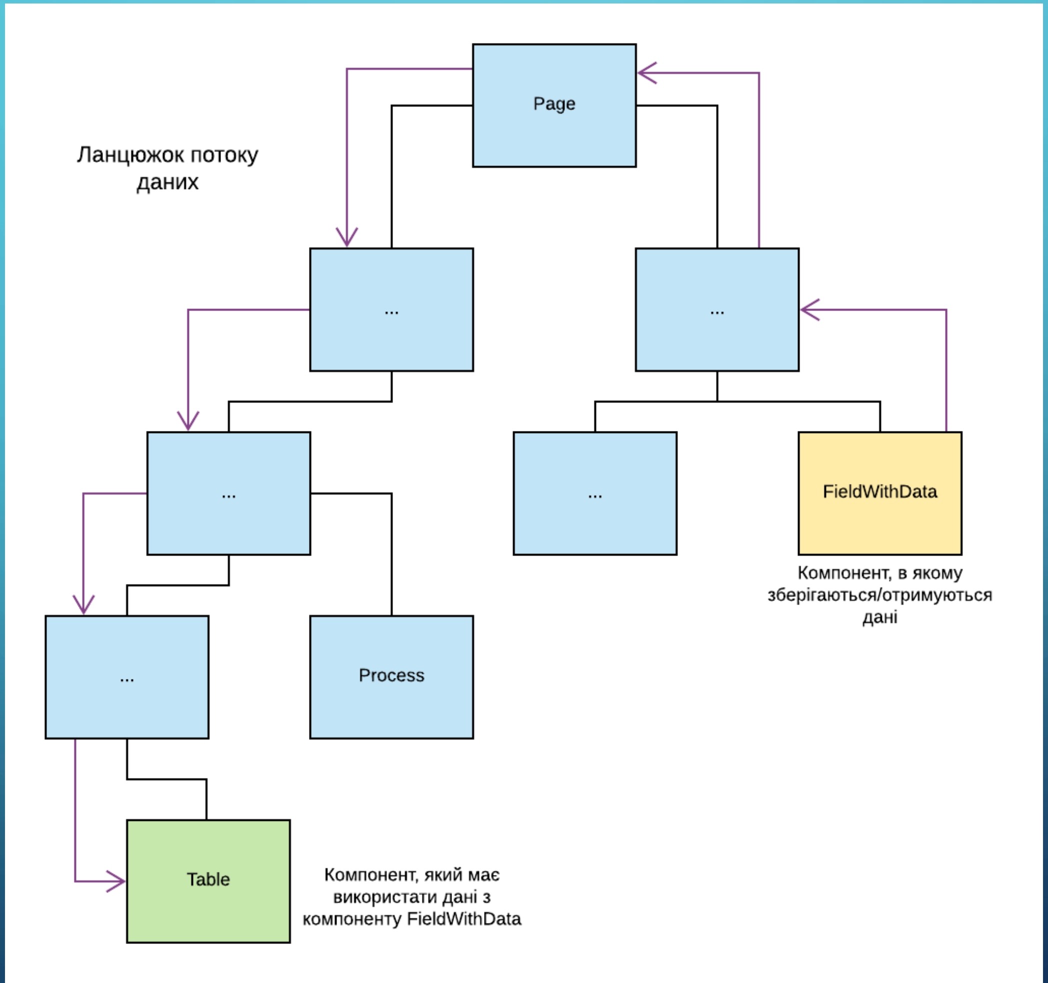
АВТОР: ЖУЛКЕВСЬКИЙ ВЛАДИСЛАВ ДМИТРОВИЧ,

ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, 3 КУРС

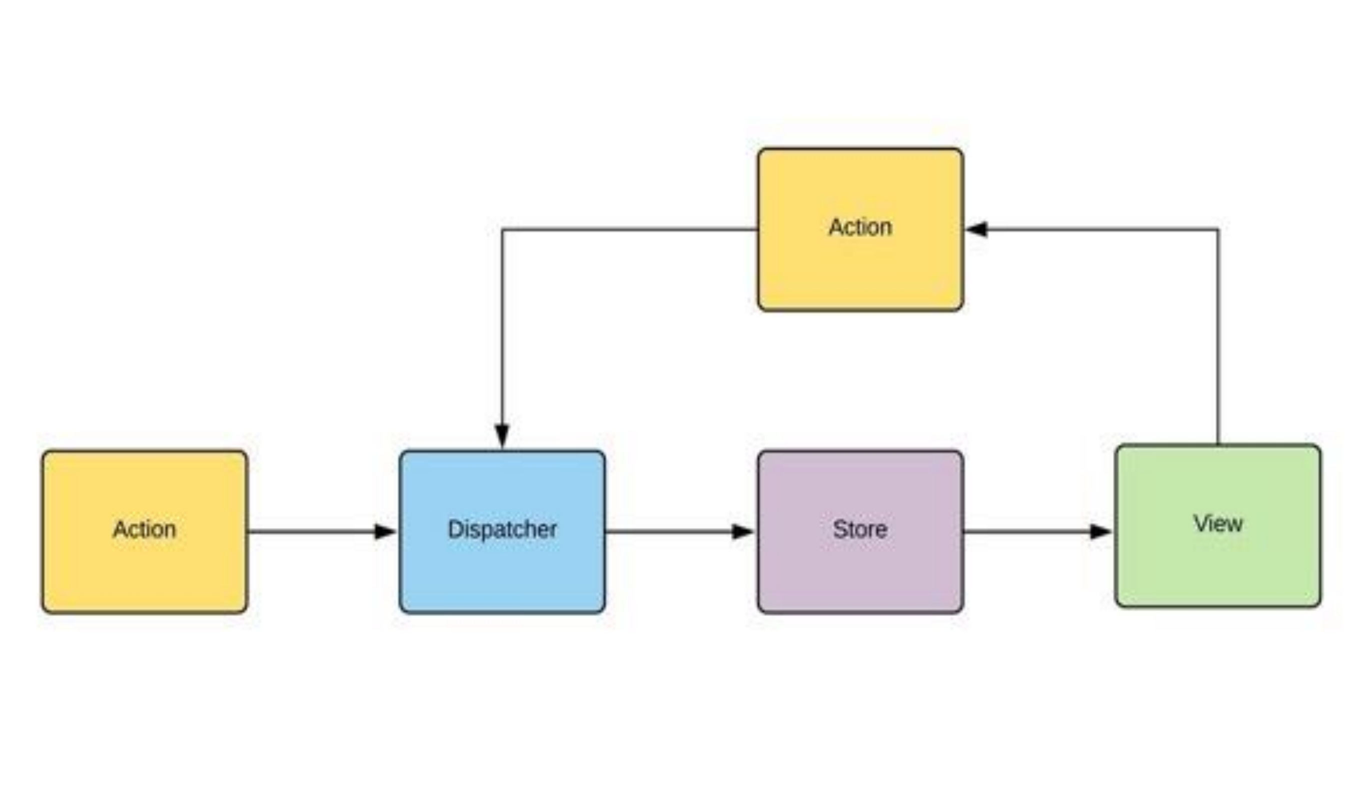
МЕТА ТА ЗАВДАННЯ КУРСОВОЇ РОБОТИ

- Висвітлення основних концепцій архітектури Flux;
- Опис проблеми у веб-застосунках та рішення, які надає підхід Flux;
- Порівняння Flux та MVC підходів;
- Аналіз бібліотек Redux та React.js;
- Створення веб-застосунку з використанням бібліотек React.js та Redux.

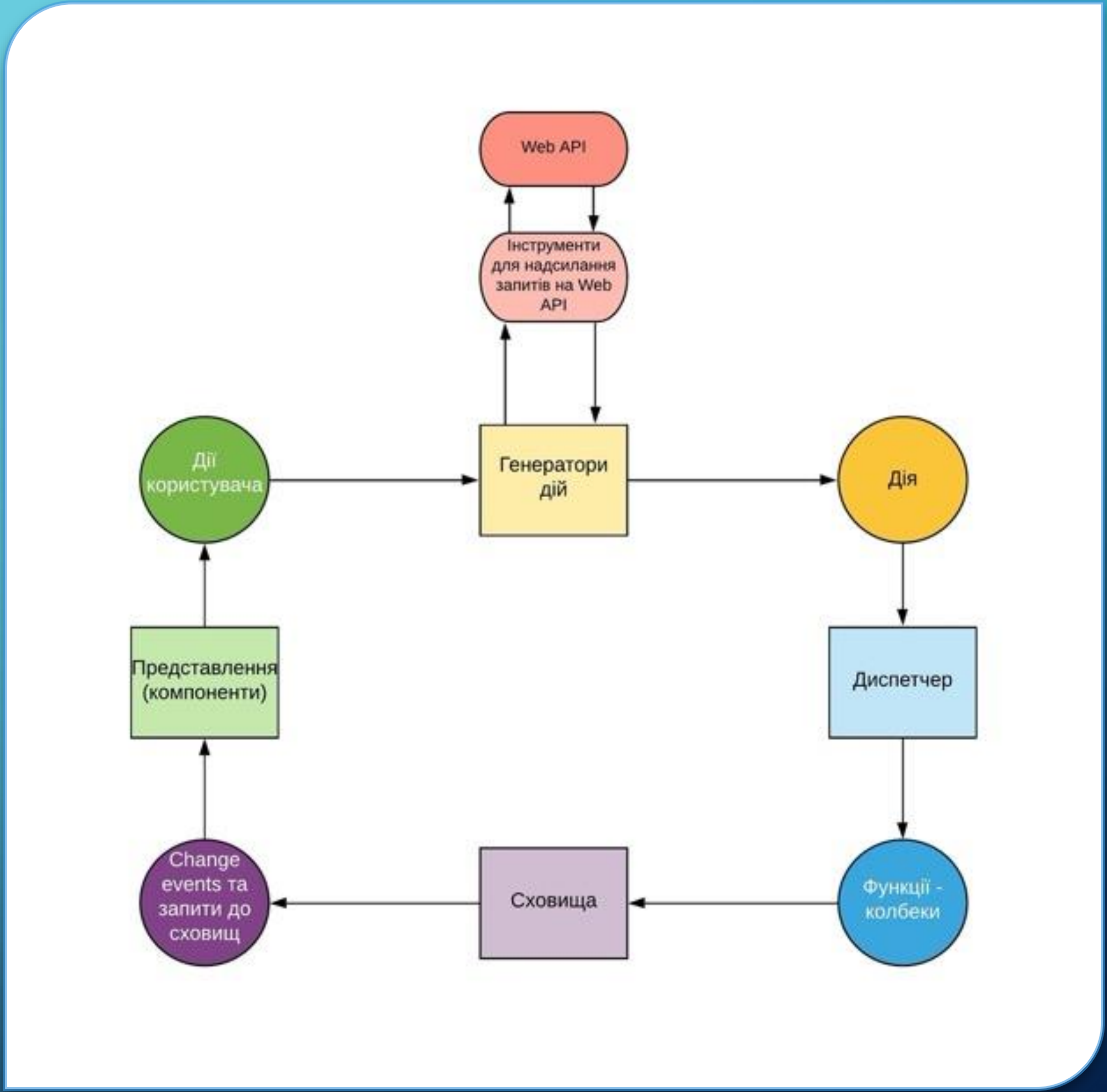
ПРОБЛЕМА МЕХАНІЗМУ ПЕРЕДАЧІ ДАНИХ У КОМПОНЕНТНО-ОРІЄНТОВАНИХ БІБЛІОТЕК ТА ФРЕЙМВОРКІВ



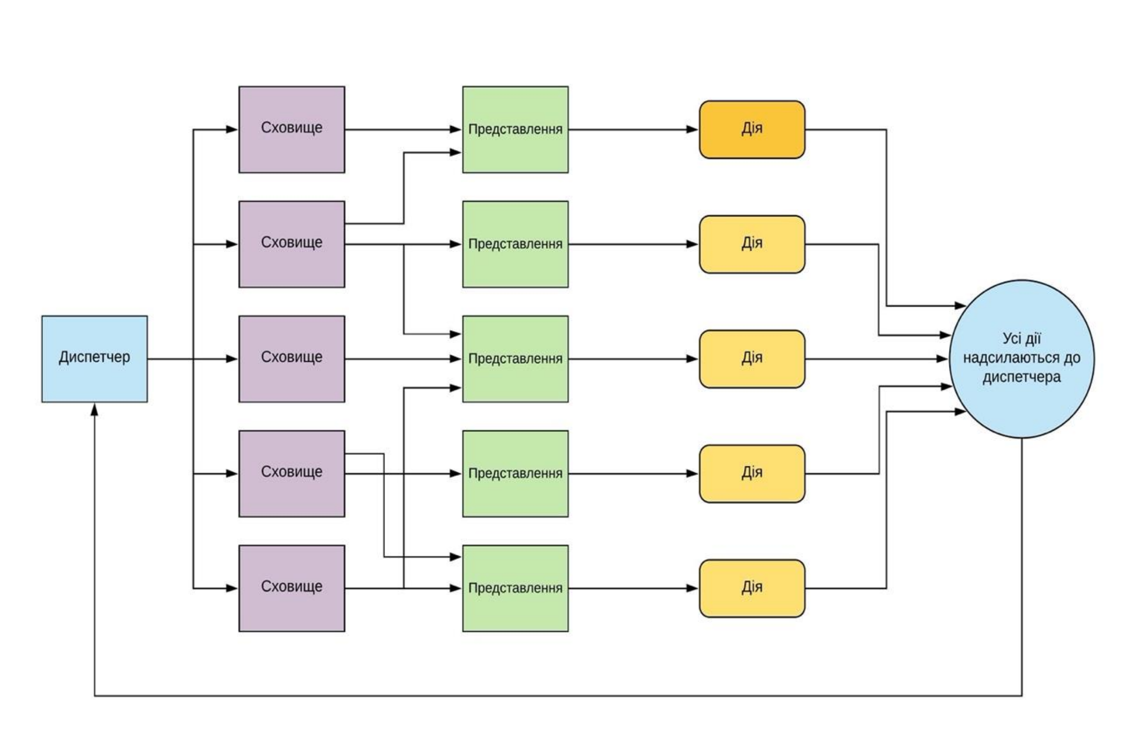
СТРУКТУРА FLUX АРХІТЕКТУРИ

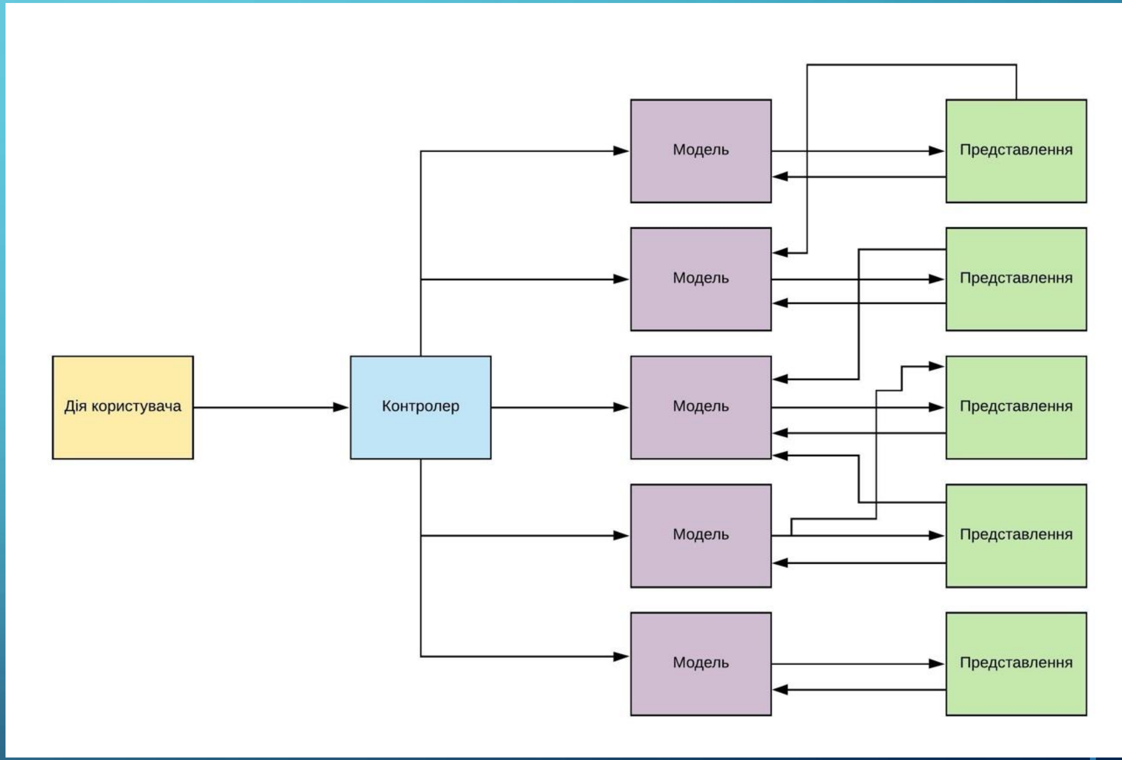


РОЗШИРЕНА МОДЕЛЬ ПОТОКУ ДАНИХ В АРХІТЕКТУРІ FLUX



FLUX VS MVC

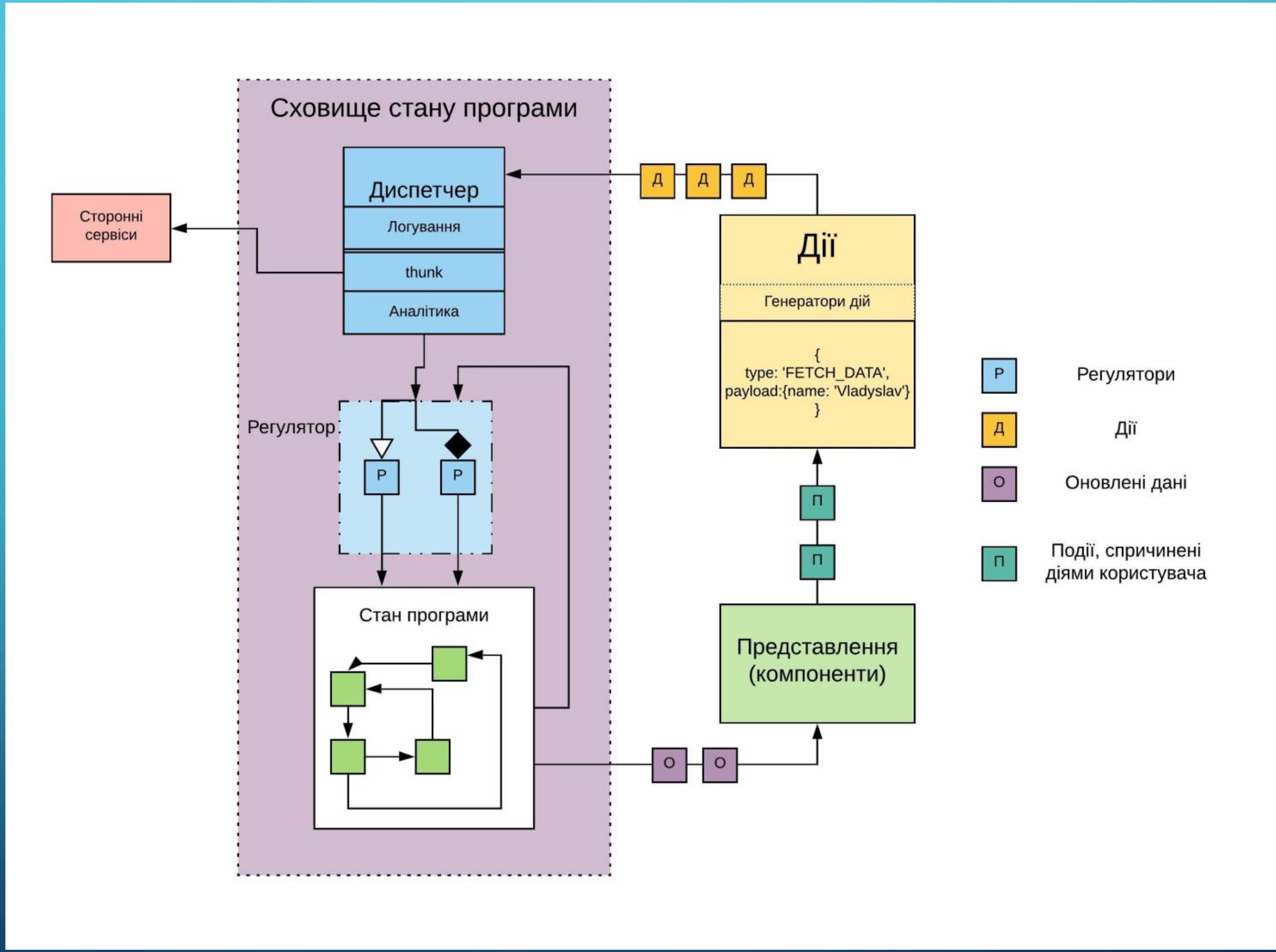

Сховище
Сховище
Сховище
Сховище
Сховище
Представлення
Представлення
Представлення
Представлення
Представлення
Дія
Дія
Дія
Дія
Дія
Диспетчер
Усі дії надсилаються до диспетчера


Дія користувача
Контролер
Модель
Модель
Модель
Модель
Модель
Представлення
Представлення
Представлення
Представлення
Представлення

ОСОБЛИВОСТІ СПЕЦИФІКАЦІЇ REDUX

- Нові компоненти – регулятори (reducers);
- Сховище – єдиний plain old javascript object(pojo);
- Дії (actions) також представлені у вигляді pojo.

ПОТІК ДАНИХ У REDUX



REACT REDUX

- Функція connect()
- Компонент provider
- useSelector()
- useDispatch()
- useStore()

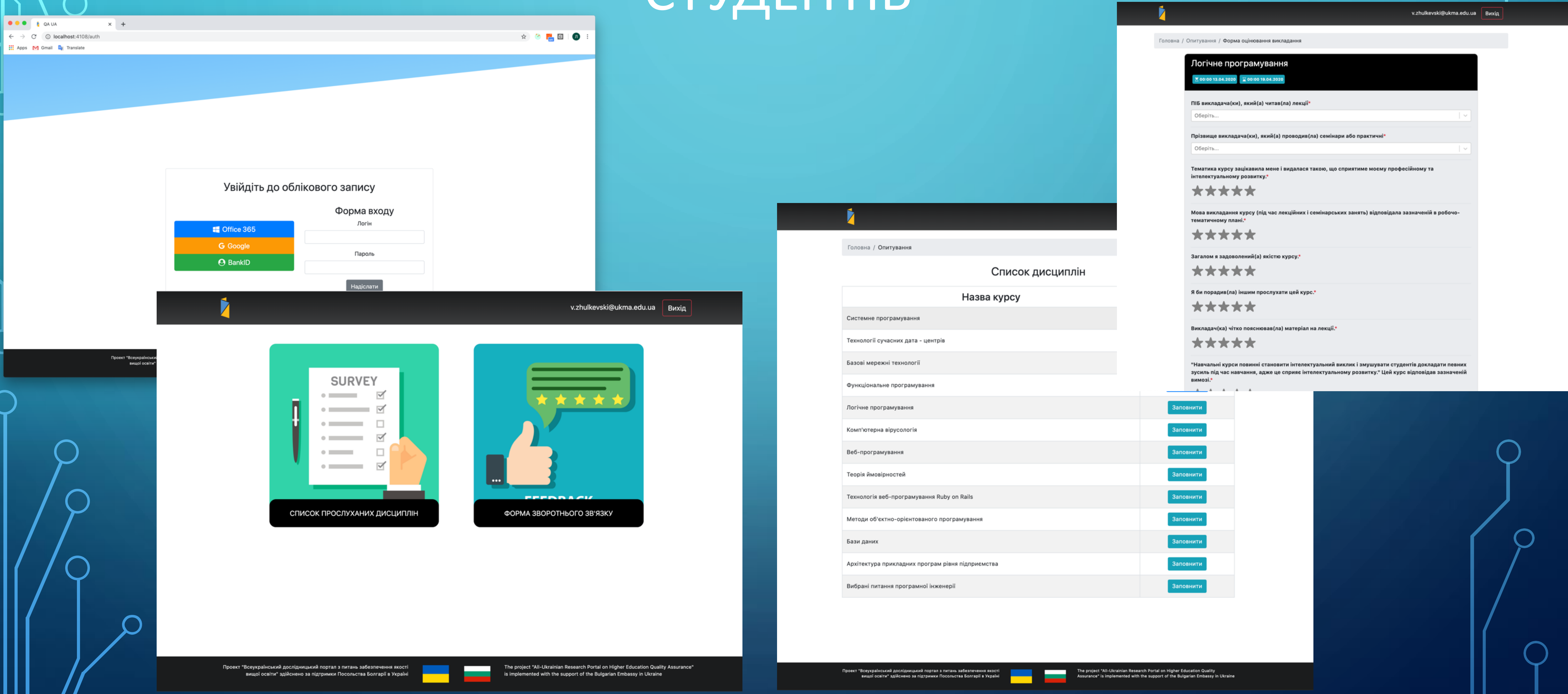
ІНІЦІАЛІЗАЦІЯ СХОВИЩА REDUX

```
import institutions from "./institution";
import surveys from "./survey";
import results from "./result"
import admin_data from "./adminData"
import sidebar from "./sidebar"
import {reducer as formReducer} from "redux-form";

const appReducer = combineReducers( reducers: {
  auth: auth,
  institutions: institutions,
  surveys: surveys,
  results: results,
  admin_data: admin_data,
  sidebar: sidebar,
  form: formReducer
});

const rootReducer =  (state, action) => {
  if (action.type === ACTION_TYPES.SIGN_OUT) {
      state = undefined;
  }
  return appReducer(state, action)
};


export default rootReducer;
```

```
import {applyMiddleware, compose, createStore} from "redux";
import thunkMiddleware from "redux-thunk";
import rootReducer from "./modules";

const composeEnhancers = window.__REDUX_DEVTOOLS_EXTENSION_COMPOSE__ || compose;

export default function store() {
  return createStore(
    rootReducer,
    composeEnhancers(applyMiddleware(thunkMiddleware))
  );
}
```

```
import React from "react";
import ReactDOM from "react-dom";
import {Provider} from "react-redux";
import store from "./redux/store";

import App from "./scenes/App/App";

ReactDOM.render(
    <Provider store={store()}>
        <App/>
    </Provider>,
    document.querySelector("#root")
);
```

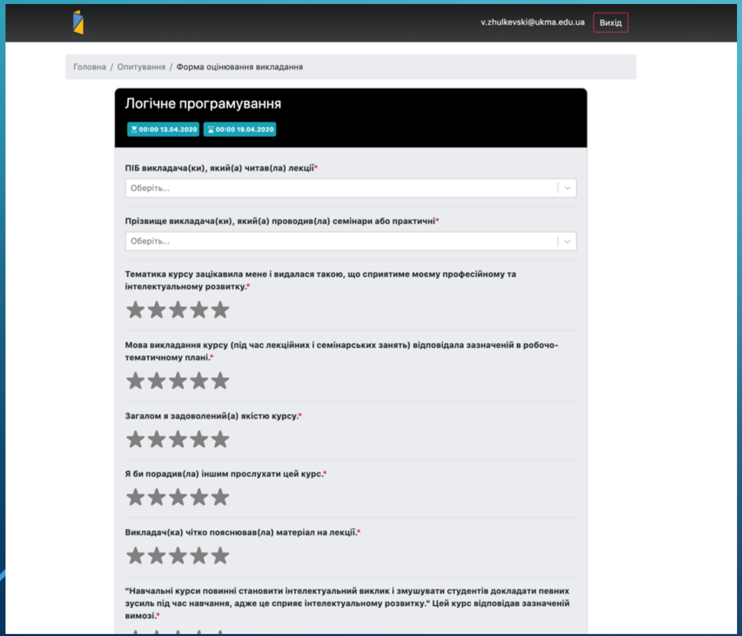
КЛІЄНТСЬКА ЧАСТИНА СЕРВІСУ ОПИТУВАННЯ СТУДЕНТІВ



ПОТІК ДАНИХ У КОМПОНЕНТІ

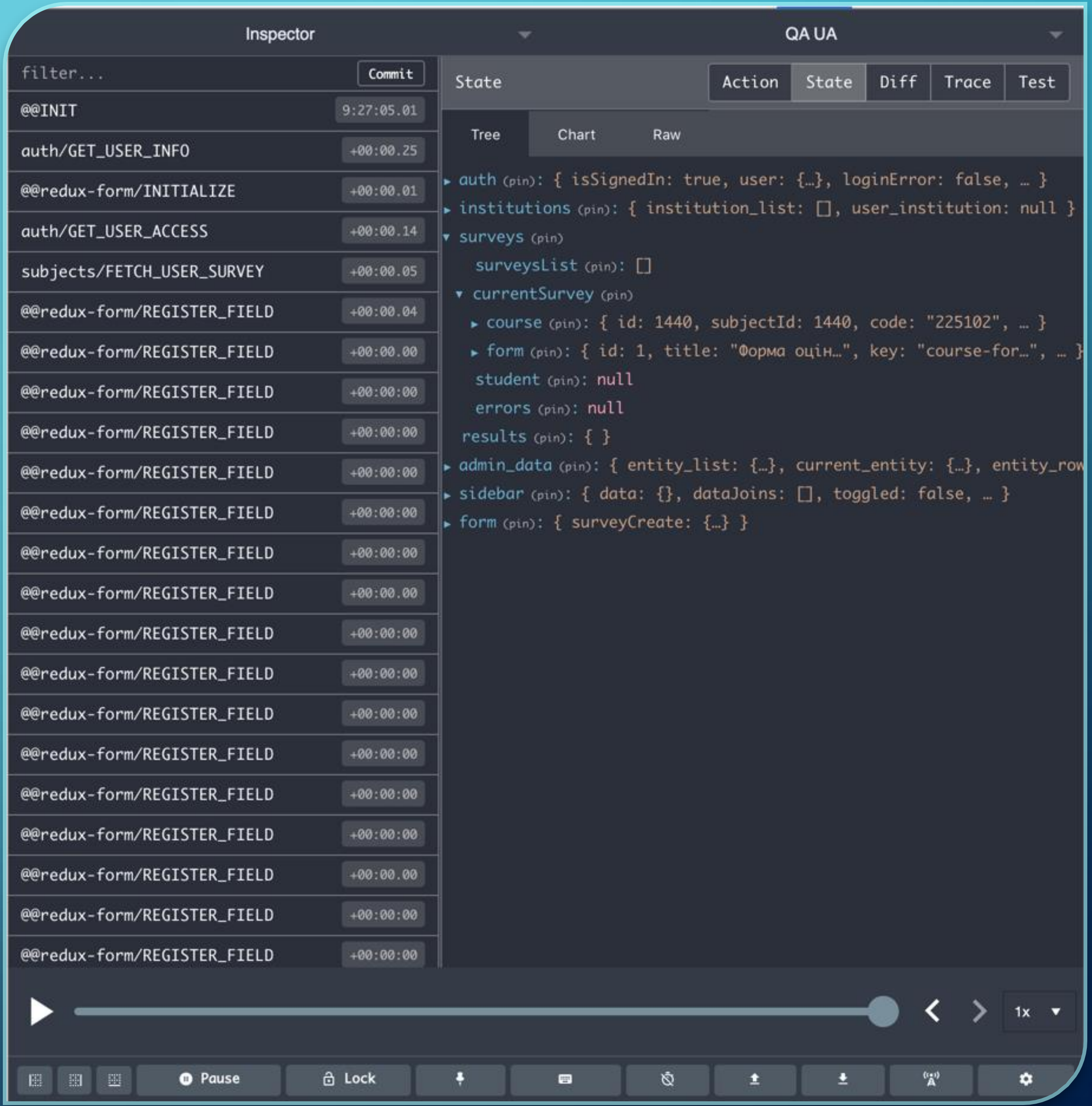
```
const Survey = (props) => {
    const user =useSelector(state=>state.auth.user);
    const survey = useSelector(state=>state.surveys.currentSurvey);
    const errors = useSelector(state=>state.surveys.errors);
    const dispatch = useDispatch();

    const st_id = user.person.students[0].code;
    const course_id = props.match.params.id;
    useEffect( effect: () => {
        dispatch(actionCreators.doFetchUserSurvey(st_id, course_id));
    }, deps: [st_id,course_id,dispatch]);
```

```
const doFetchUserSurvey = (st_id, course_id) => async (dispatch) => {
    api.get('/student/' + st_id + '/course/' + course_id + '/survey').then((response) => {
        dispatch({type: ACTION_TYPES.FETCH_USER_SURVEY, payload: response.data})
    }).catch(() => {
        addNotification( message: "Виникла помилка при отриманні анкети.");
    });
};
```




```
function reducer(state = initialState, action) {
    switch (action.type) {
        case ACTION_TYPES.GET_STUDENT_INFO:
            return {...state, student: action.payload};
        case ACTION_TYPES.FETCH_USER_SURVEYS:
            return {...state, surveysList: action.payload};
        case ACTION_TYPES.FETCH_USER_SURVEY:
            return {...state, currentSurvey: action.payload};
        case ACTION_TYPES.POST_USER_SURVEY:
            return {...state, currentSurvey: null, errors: null};
        case ACTION_TYPES.DISPLAY_ERRORS:
            return {...state, errors: action.payload};
        case ACTION_TYPES.REMOVE_ERROR:
            return {...state, errors: state.errors.filter((err)=>err !== action.payload)};
        default:
            return state;
    }
}
```

REDUX DEVTOOLS



ДЯКУЮ ЗА УВАГУ!