НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №7

з дисципліни

“Програмування мобільних систем / Розроблення клієнтських додатків для мобільних платформ”

Виконав:

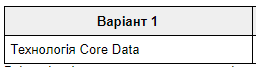
студент групи ІВ-81

ЗК ІВ-8102

Бєлов Микита Сегрійович

Київ 2021

**Варіант № 1 (використовував redux)**



**Лістинг коду**

**Picture.js**

import React, { useState, useEffect } from 'react';

import { Text, View, ScrollView, Dimensions, Platform, StyleSheet, TouchableOpacity, StatusBar, SafeAreaView } from 'react-native';

import { MaterialCommunityIcons } from '@expo/vector-icons';

import \* as ImagePicker from 'expo-image-picker';

import { addImages } from "../redux/actions";

import { useSelector, useDispatch } from 'react-redux';

import Layout from "../layout";

import { styleConfig } from "../style";

*const* window = Dimensions.get("window");

*const* screen = Dimensions.get("screen");

*const* maxSizeOfArray = (*arr* = [], *maxArrSize* = 6) *=>* {

*const* lst = [];

    for (*let* i = 0; i < Math.ceil(arr.length / maxArrSize); i++) {

        lst[i] = arr.slice(i \* maxArrSize, (i \* maxArrSize) + maxArrSize);

    }

    return lst;

};

export default *function* Picture({ *navigation* }) {

*const* { imageStorage } = useSelector(*state* *=>* state.filmsReducer);

*const* dispatch = useDispatch();

*const* addToStorage = *img* *=>* dispatch(addImages(img));

*const* handleAddImage = *img* *=>* {

        addToStorage(img);

    };

*const* [dimensions, setDimensions] = useState({ window, screen });

*const* imageSize = {

        width: dimensions.window.width / 5,

        height: dimensions.window.width / 5,

    }

*const* onChange = ({ *window*, *screen* }) *=>* {

        setDimensions({ window, screen });

    };

    useEffect(() *=>* {

        Dimensions.addEventListener("change", onChange);

        return () *=>* {

            Dimensions.removeEventListener("change", onChange);

        };

    });

    useEffect(() *=>* {

        (async () *=>* {

            if (Platform.OS !== 'web') {

*const* { status } = await ImagePicker.requestMediaLibraryPermissionsAsync();

                if (status !== 'granted') {

                    alert('We need camera roll permissions to make this work!');

                }

            }

        })();

    }, []);

    // useEffect(() => {

    //     const url = `https://pixabay.com/api/?key=19193969-87191e5db266905fe8936d565&q=small+animals&image\_type=photo&per\_page=18`;

    //     (async () => {

    //         try {

    //             const fetchResult = await fetch(url);

    //             const loadedData = await fetchResult.json();

    //             const loadedDataURIs = loadedData['hits'].map((loadData) => ({ uri: loadData['largeImageURL'] }));

    //             if (imageStorage.length <= 0) {

    //                 handleAddImage(loadedDataURIs)

    //             }

    //         } catch (e) {

    //             console.log(e.message)

    //         }

    //     })();

    // }, []);

*const* pickImage = async () *=>* {

*let* pickedImage = await ImagePicker.launchImageLibraryAsync({

            mediaTypes: ImagePicker.MediaTypeOptions.All,

            allowsEditing: false,

            quality: 1,

        });

        if (!pickedImage.cancelled) {

            handleAddImage([...imageStorage, { 'uri': pickedImage.uri }])

        }

    };

*const* PictureComponent = maxSizeOfArray(imageStorage).map(

*image* *=>* (

            <*Layout*

                key={*image*[0].uri}

                layout={*image*}

                width={imageSize.width}

                height={imageSize.height}

            />

        )

    );

    return (

        <*SafeAreaView* style={styles.container}>

            {

                imageStorage.length === 0 ?

                <*View* style={styles.textContainer}>

                    <*Text* style={styles.text}>No items found</*Text*>

                </*View*> :

                    <*View*>

                        <*TouchableOpacity*

                            style={{ zIndex: 1, position: 'absolute', bottom: 10,  right: 0, color: 'white' }}

                            activeOpacity={0.5}

                            onPress={pickImage}>

                        <*MaterialCommunityIcons* style={styles.addIcon} name="plus-circle" color={'#808082'} size={50} />

                        </*TouchableOpacity*>

                    <*ScrollView*>

                        {PictureComponent}

                    </*ScrollView*>

                </*View*>

            }

        </*SafeAreaView*>

    );

}

*const* styles = StyleSheet.create({

    container: {

        backgroundColor: styleConfig.bg,

        flex: 1,

        borderWidth: 1,

        borderColor: styleConfig.bg,

    },

    textContainer: {

        flex: 1,

        marginTop: '10%'

    },

    text: {

        textAlign: 'center',

        backgroundColor: styleConfig.bg,

        fontSize: 18,

        color: styleConfig.color

    },

    addIcon: {

        textAlign: 'right',

        marginHorizontal: 16,

        marginBottom: 5,

        marginTop: 2,

        color: 'white'

    },

});

**Movie.js**

import React, { useState, useEffect } from 'react';

import { SafeAreaView, View, VirtualizedList, StyleSheet, Text, StatusBar, Image, TextInput, TouchableOpacity, Dimensions } from 'react-native';

import { MaterialCommunityIcons } from '@expo/vector-icons';

import Swipeable from 'react-native-gesture-handler/Swipeable';

import { styleConfig } from "../style";

import \* as Network from 'expo-network';

import { useSelector, useDispatch } from 'react-redux';

import { addFilm } from '../redux/actions';

export *let* DATA = []

*const* getItemCount = (*data*) *=>* data.length;

*const* window = Dimensions.get("window");

*const* screen = Dimensions.get("screen");

*const* getItem = (*data*, *index*) *=>* {

    return ({

        id: `${data[index].imdbID}`,

        title: `${data[index].Title}`,

        year: `${data[index].Year}`,

        type: `${data[index].Type}`,

        poster: `${data[index].Poster}`

    })

};

export default *function* Movies({ *navigation* }) {

*const* { filmsData } = useSelector(*state* *=>* state.filmsReducer);

*const* dispatch = useDispatch();

*const* addToStorage = *films* *=>* dispatch(addFilm(films));

*const* handleAddFilm = *films* *=>* {

        addToStorage(films);

    };

*const* [dimensions, setDimensions] = useState({ window, screen });

*const* [selectedData, setSelectedData] = useState([]);

*const* [reloade, setReloade] = useState(false);

*const* [search, setSearch] = useState('');

*const* cachedContent = filmsData

    // const filteredItems = (items, term) => {

    //     if (term.length === 0) {

    //         return null

    //     }

    //     if (term.trim().length === 0) {

    //         return null

    //     }

    //     return items.filter((item) => {

    //         if (

    //             item.Title

    //                 .replace(/[^a-zA-Z ]/g, "")

    //                 .toLowerCase()

    //                 .indexOf(term) > -1) {

    //             return (

    //                 item

    //             )

    //         }

    //     })

    // }

*const* onChange = ({ *window*, *screen* }) *=>* {

        setDimensions({ window, screen });

    };

    useEffect(() *=>* {

        Dimensions.addEventListener("change", onChange);

        return () *=>* {

            Dimensions.removeEventListener("change", onChange);

        };

    });

*const* orientation = () *=>* {

*const* dim = Dimensions.get('screen');

        if (dim.height >= dim.width) {

            return portrait

        } else {

            return landscape

        }

    }

*function* getUniqueListBy(*arr*, *key*) {

        return [...new *Map*(*arr*.map(*item* *=>* [*item*[*key*], *item*])).values()]

    }

*const* apiSearchFunction = async (*text*) *=>* {

*const* filteredText = *text*.toLowerCase().replace(/[^a-zA-Z ]/g, "").replace(/\s+/g, ' ').trim().replace(/,/g, '')

        if (filteredText.length < 3) {

            setSelectedData([])

        } else {

*let* zminna = cachedContent

            if ((await Network.getNetworkStateAsync()).isConnected) {

*let* url = `http://www.omdbapi.com/?apikey=dbf1a99b&s=${filteredText}&page=1`;

*let* response = await fetch(url)

                    .then(*loadedData* *=>* *loadedData*.json())

                    .catch(*error* *=>* {

                        console.log(*error*);

                    })

                zminna = response.Search;

            } else {

                zminna = cachedContent;

            }

            if (zminna !== undefined) {

                setSelectedData(getUniqueListBy(zminna, 'imdbID'))

                // handleAddFilm(getUniqueListBy(zminna, 'imdbID'))

            }

        }

    };

*const* LeftActions = () *=>* {

        return (

            <*View* style={portrait.rightAction}>

                <*Text* style={portrait.actionText}>Delete</*Text*>

            </*View*>

        )

    }

*function* Item({ *id*, *title*, *year*, *type*, *poster* }) {

        return (

            <*TouchableOpacity*

                activeOpacity={0.5}

                onPress={

                    () *=>* *navigation*.navigate('Details', {

                        id: *id*,

                        title: *title*,

                        year: *year*,

                    })}>

                <*Swipeable*

                    renderRightActions={LeftActions}

                    onSwipeableRightOpen={() *=>* {

*const* obj = selectedData.findIndex(*elem* *=>* *elem*.imdbID === *id*)

                        selectedData.splice(obj, 1);

                        setReloade(!reloade)

                    }}>

                    <*View* style={portrait.item}>

                        <*View* style={portrait.posterViev}>

                            <*Image*

                                style={orientation().poster}

                                source={*poster* === 'N/A' ? require('../assets/posters/no-poster.jpg') : { uri: *poster* }}

                            />

                        </*View*>

                        <*View* style={orientation().textViev}>

                            <*Text* style={portrait.title}>{*title*}</*Text*>

                            <*Text* style={portrait.details}>{*year*}</*Text*>

                            <*Text* style={portrait.details}>{*type*}</*Text*>

                        </*View*>

                    </*View*>

                </*Swipeable*>

            </*TouchableOpacity*>

        )

    }

    return (

        <*SafeAreaView* style={portrait.container}>

            {

                React.useLayoutEffect(() *=>* {

*navigation*.setOptions({

                        headerRight: () *=>* (

                            <*TouchableOpacity*

                                activeOpacity={0.5}

                                onPress={() *=>* { navigation.navigate('Create')}}>

                                <*MaterialCommunityIcons* style={portrait.addIcon} name="plus" color={'#808082'} size={30} />

                            </*TouchableOpacity*>

                        ),

                    });

                }, [navigation])

            }

            <*View* style={portrait.sectionStyle}>

                <*MaterialCommunityIcons* style={portrait.searchIcon} name="magnify" color={'#808082'} size={26} />

                <*TextInput*

                    style={portrait.textInputStyle}

                    placeholder={'Find movie you want'}

                    clearButtonMode={'while-editing'}

                    onChangeText={(*text*) *=>* {

                        apiSearchFunction(text)

                        setSearch(text)

                        }

                    }

                />

            </*View*>

            <*VirtualizedList*

                data={selectedData}

                ItemSeparatorComponent={() *=>* {return(<*View* style={portrait.separator}/>)}}

                renderItem={({ *item* }) *=>* (

                    <*Item* id={item.id} title={item.title} year={item.year} type={item.type} poster={item.poster} />

                )}

                getItemCount={getItemCount}

                getItem={getItem}

            />

        </*SafeAreaView*>

    );

}

*const* portrait = StyleSheet.create({

    container: {

        flex: 1,

        marginTop: StatusBar.currentHeight,

        backgroundColor: styleConfig.bg,

    },

    separator: {

        marginLeft: 'auto',

        marginRight: 'auto',

        backgroundColor: styleConfig.separator,

        width: '92%',

        height: 0.5,

    },

    item: {

        flexDirection: 'row',

        backgroundColor: styleConfig.bg,

        height: 'auto',

        justifyContent: 'center',

        marginHorizontal: 0,

        padding: 20,

    },

    title: {

        fontSize: 30,

        color: styleConfig.color

    },

    poster: {

        width: 86,

        height: 150,

        borderRadius: 1,

        borderColor: styleConfig.color,

        borderWidth: 3,

        marginTop: 6

    },

    posterViev: {

        flex: 2

    },

    textViev: {

        flex: 10,

        marginLeft: 50,

    },

    details: {

        fontSize: 16,

        marginTop: 10,

        color: styleConfig.color

    },

    textInputStyle: {

        flex: 1,

        height: 40,

        margin: 2,

        borderRadius: 10,

        color: styleConfig.searchColor.color

    },

    sectionStyle: {

        flexDirection: 'row',

        justifyContent: 'center',

        alignItems: 'center',

        backgroundColor: styleConfig.searchColor.bg,

        height: 40,

        borderRadius: 12,

        marginTop: 10,

        marginHorizontal: 6,

        marginBottom: 3,

    },

    searchIcon: {

        margin: 8,

        color: styleConfig.searchColor.color

    },

    rightAction: {

        flex: 1,

        justifyContent: 'center',

        backgroundColor: 'red',

    },

    actionText: {

        color: '#fff',

        padding: 20,

        textAlign: 'right'

    },

    addIcon: {

        textAlign: 'right',

        marginHorizontal: 16,

        marginBottom: 5,

        marginTop: 2,

        color: 'white'

    }

});

*const* landscape = StyleSheet.create({

    textViev: {

        marginRight: 20,

        flex: 10,

        marginLeft: -20

    },

    poster: {

        width: 80,

        height: 135,

        borderRadius: 1,

        marginTop: 6,

        borderColor: styleConfig.color,

        borderWidth: 3,

    },

})

**Picture.js**

import React, { useState, useEffect } from 'react';

import { View, Dimensions, Text, StyleSheet, SafeAreaView, ScrollView, Image, ActivityIndicator } from 'react-native';

import { styleConfig } from "../style";

import { addFilmInfo } from "../redux/actions";

import { useSelector, useDispatch } from 'react-redux';

import \* as Network from 'expo-network';

*const* window = Dimensions.get("window");

*const* screen = Dimensions.get("screen");

export default *function* Details({ *route* }) {

*const* { filmInfoData } = useSelector(*state* *=>* *state*.filmsReducer);

*const* dispatch = useDispatch();

*const* addToStorage = *films* *=>* dispatch(addFilmInfo(*films*));

*const* handleAddFilmInfo = *films* *=>* {

        addToStorage(*films*);

    };

*const* { id } = *route*.params;

*const* [dimensions, setDimensions] = useState({ window, screen });

*const* [isLoading, setLoading] = useState(true);

*const* onChange = ({ *window*, *screen* }) *=>* {

        setDimensions({ window, screen });

    };

    useEffect(() *=>* {

        Dimensions.addEventListener("change", onChange);

        return () *=>* {

            Dimensions.removeEventListener("change", onChange);

        };

    });

*const* getUniqueInfoList = (*arr*, *key*) *=>* {

            return [...new *Map*(*arr*.map(*item* *=>* [*item*[*key*], *item*])).values()]

        }

    useEffect(() *=>* {

*const* fetchData = async () *=>* {

            try {

                await fetch(`http://www.omdbapi.com/?apikey=dbf1a99b&i=${id}`)

                    .then((*response*) *=>* *response*.json())

                    .then( (*data*) *=>* {

*const* filteredFilmInfo =

                                    getUniqueInfoList(

                                        [*data*, ...filmInfoData],

                                        'imdbID'

                                    )

                                // handleAddFilmInfo(filteredFilmInfo)

                            }

                    )

                    .finally(() *=>* setLoading(false));

            } catch (error) {

                console.error(error);

            }

        }

        fetchData()

    }, []);

*const* orientation = () *=>* {

*const* dim = Dimensions.get('screen');

        if (dim.height >= dim.width) {

            return styles

        } else {

            return landscape

        }

    }

*let* check = []

    return (

        <*SafeAreaView*>

            <*ScrollView* style={{ backgroundColor: styleConfig.bg }}>

                {

                    filmInfoData.map((*item*, *i*) *=>* {

                        if ( isLoading ) {

                            return (

                                <*View* key={*item*.imdbID} style={orientation().loading}><*ActivityIndicator* size='large' /></*View*>

                            )

                        } else if(id === *item*.imdbID) {

                            check.push('')

                            return (

                                <*View* style={orientation().infoScreen} key={*item*.imdbID}>

                                    <*View* style={orientation().infoImageSection}>

                                        <*Image*

                                            style={orientation().infoImage}

                                            source={ *item*.Poster === 'N/A' ? require('../assets/posters/no-poster.jpg') : { uri: *item*.Poster } }

                                        />

                                    </*View*>

                                    <*View* style={orientation().infoScreenTextView}>

                                        <*Text* style={orientation().titleText}>Title</*Text*>

                                        <*Text* style={orientation().subText}>{*item*.Title}</*Text*>

                                        <*Text* style={orientation().titleText}>Runtime</*Text*>

                                        <*Text* style={orientation().subText}>{*item*.Runtime}</*Text*>

                                        <*Text* style={orientation().titleText}>Genre</*Text*>

                                        <*Text* style={orientation().subText}>{*item*.Genre}</*Text*>

                                        <*Text* style={orientation().titleText}>Awards</*Text*>

                                        <*Text* style={orientation().subText}>{*item*.Awards}</*Text*>

                                        <*Text* style={orientation().titleText}>Rating</*Text*>

                                        <*Text* style={orientation().subText}>{*item*.imdbRating}</*Text*>

                                        <*Text* style={orientation().titleText}>Actors</*Text*>

                                        <*Text* style={orientation().subText}>{*item*.Actors}</*Text*>

                                        <*Text* style={orientation().titleText}>Language</*Text*>

                                        <*Text* style={orientation().subText}>{item.Language}</*Text*>

                                        <*Text* style={orientation().titleText}>Plot</*Text*>

                                        <*Text* style={orientation().subText}>{item.Plot}</*Text*>

                                        <*Text* style={orientation().titleText}>Director</*Text*>

                                        <*Text* style={orientation().subText}>{item.Director}</*Text*>

                                        <*Text* style={orientation().titleText}>Country</*Text*>

                                        <*Text* style={orientation().subText}>{item.Country}</*Text*>

                                        <*Text* style={orientation().titleText}>Production</*Text*>

                                        <*Text* style={orientation().subText}>{item.Production}</*Text*>

                                        <*Text* style={orientation().titleText}>Released</*Text*>

                                        <*Text* style={orientation().subText}>{item.Released}</*Text*>

                                        <*Text* style={orientation().titleText}>Year</*Text*>

                                        <*Text* style={orientation().subText}>{item.Year}{'\n'}</*Text*>

                                    </*View*>

                                </*View*>

                            )

                        }

                    })

                }

            </*ScrollView*>

            {

                check.length === 0 ?

                <*View* style={{backgroundColor: styleConfig.bg, justifyContent: 'center', alignItems: 'center', }}>

                    <*Text*>No Data In Database</*Text*>

                </*View*> : null

            }

        </*SafeAreaView*>

    )

}

*const* styles = StyleSheet.create({

    container: {

        flex: 1,

        justifyContent: 'center',

        backgroundColor: 'white',

        alignItems: 'center',

        backgroundColor: styleConfig.bg,

    },

     textStyle: {

        textAlign: 'center',

        color: styleConfig.color,

        fontSize: 20,

    },

    titleText: {

        color: styleConfig.color,

        fontWeight: '600',

        fontSize: 20,

        marginVertical: 2,

    },

    subText: {

        color: styleConfig.color,

        fontWeight: '400',

        marginTop: 2,

        marginBottom: 8,

        fontSize: 18,

    },

    infoScreen: {

        paddingHorizontal: 13,

        paddingTop: 10,

        paddingBottom: 40,

        backgroundColor: styleConfig.bg

    },

    infoImageSection: {

        alignItems: 'center'

    },

    infoImage: {

        width: 180,

        height: 270,

        borderWidth: 2,

        borderColor: '#EEE'

    },

    infoScreenTextView: {

        marginTop: 30,

    },

    loading: {

        position: 'absolute',

        marginTop: 50,

        left: 0,

        right: 0,

        top: 0,

        bottom: 0,

        alignItems: 'center',

        justifyContent: 'center',

        backgroundColor: styleConfig.bg,

        color: styleConfig.color

    }

});

*const* landscape = StyleSheet.create({

    infoScreen: {

        paddingHorizontal: 16,

        paddingTop: 10,

        paddingBottom: 40,

        backgroundColor: styleConfig.bg,

        flex: 1,

        flexDirection: 'row',

    },

    infoImage: {

        width: 170,

        height: 300,

        borderWidth: 2,

        borderColor: styleConfig.color,

        marginTop: 6,

    },

    infoScreenTextView: {

        paddingLeft: 14,

        flexShrink: 1

    },

    titleText: {

        color: styleConfig.color,

        fontWeight: '600',

        fontSize: 21,

        marginVertical: 1,

    },

    subText: {

        color: styleConfig.color,

        fontWeight: '400',

        fontSize: 19,

    },

    loading: {

        position: 'absolute',

        left: 0,

        right: 0,

        top: 0,

        bottom: 0,

        alignItems: 'center',

        justifyContent: 'center',

        backgroundColor: styleConfig.bg,

        color: styleConfig.color

    }

})

**Висновок**

У даній роботі я зрозумів як правильно працювати з асинхронним сховищем, додав туди окремо картинки і масив з фільмами, тепер при вимненні інтернету додаток буде підтягувати всю інформацію зі сховища.