**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по домашней работе

«Приложение на SolidJS и Tauri»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-32Б |  | преподаватель каф. ИУ5 |
| Мажитов В. |  | 1. Гапанюк Ю.Е. |
|  |  |  |

**Описание задания**

1. Выберите язык программирования (который Вы ранее не изучали) и (1)

напишите по нему реферат с примерами кода или (2) реализуйте на нем

небольшой проект (с детальным текстовым описанием).

1. Реферат (проект) может быть посвящен отдельному аспекту (аспектам) языка

или содержать решение какой-либо задачи на этом языке.

1. Необходимо установить на свой компьютер компилятор (интерпретатор,

транспилятор) этого языка и произвольную среду разработки.

1. В случае написания реферата необходимо разработать и откомпилировать

примеры кода (или модифицировать стандартные примеры).

1. В случае создания проекта необходимо детально комментировать код.
2. При написании реферата (создании проекта) необходимо изучить и корректно

использовать особенности парадигмы языка и основных конструкций данного языка.

1. Приветствуется написание черновика статьи по результатам выполнения ДЗ.
2. Черновик статьи может быть подготовлен группой студентов, которые

исследовали один и тот же аспект в нескольких языках или решили одинаковую задачу на нескольких языках.

**Текст программы**

Файл app.tsx

import { createSignal } from "solid-js"

import { invoke } from "@tauri-apps/api/tauri"

function App() {

const [best, setBest] = createSignal<number | null>(null)

const [firstThing, setFirstThing] = createSignal("")

const [secondThing, setSecondThing] = createSignal("")

const [loading, setLoading] = createSignal(false)

async function getBest() {

// Learn more about Tauri commands at https://tauri.app/v1/guides/features/command

setLoading(true)

invoke("get\_best", { firstThing: firstThing(), secondThing: secondThing() }).then(value => {

setBest((\_) => value as number)

setLoading(false)

})

}

return (

<form

class="flex flex-col space-y-4 pt-4 px-4 h-screen dark:text-white dark:bg-black"

onSubmit={(e) => {

e.preventDefault()

getBest()

}}

>

<h1 class="text-3xl">Что круче?</h1>

<div class="flex flex-row items-center space-x-4">

<input

id="1"

class="input"

onInput={(e) => {setFirstThing(e.currentTarget.value); setBest(null)}}

placeholder="Введи что угодно"

/>

<p class="text-xl">или</p>

<input

class="input"

onInput={(e) => {setSecondThing(e.currentTarget.value); setBest(null)}}

placeholder="Введи что угодно"

/>

</div>

<button

disabled={firstThing().length === 0 || secondThing().length === 0 || loading()}

type="submit"

class="btn-primary transition-all"

>

Узнать

</button>

{best() && <p class="text-xl">{[firstThing(), secondThing()][best()!-1]} круче!</p>}

</form>

);

}

export default App;

Файл index.css

@tailwind base;

@tailwind components;

@tailwind utilities;

@layer components {

.btn-primary {

@apply transition-all py-2 px-4 bg-blue-500 text-white font-semibold rounded-lg shadow-md hover:bg-blue-700 focus:outline-none focus:ring-2 focus:ring-blue-400 focus:ring-opacity-75 disabled:bg-gray-700;

}

.input {

@apply bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5 dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500

}

}

Файл main.rs

// Prevents additional console window on Windows in release, DO NOT REMOVE!!

#![cfg\_attr(not(debug\_assertions), windows\_subsystem = "windows")]

use rtrend::{Client, Country, Keywords, SearchInterest};

use serde\_json::Number;

fn get\_metric\_for\_keyword(keyword: &str) -> u64 {

let keywords: &'static str = String::from(keyword.split(" ").collect::<Vec<&str>>()[0]).leak();

let client = Client::new(Keywords::from(keywords), Country::ALL).build();

let search\_interest = SearchInterest::new(client).get();

let metric = search\_interest["default"]["timelineData"]

.as\_array()

.unwrap()

.last()

.unwrap()

.get("value")

.unwrap()[0]

.as\_u64()

.unwrap();

metric

}

// Learn more about Tauri commands at https://tauri.app/v1/guides/features/command

#[tauri::command]

fn get\_best(first\_thing: &str, second\_thing: &str) -> Number {

let first\_rank = get\_metric\_for\_keyword(first\_thing);

let second\_rank = get\_metric\_for\_keyword(second\_thing);

if first\_rank > second\_rank {

Number::from(1)

} else {

Number::from(2)

}

}

fn main() {

tauri::Builder::default()

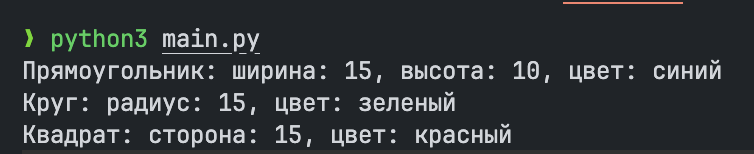
.invoke\_handler(tauri::generate\_handler![get\_best])

.run(tauri::generate\_context!())

.expect("error while running tauri application");

}

**Пример выполнения программы**

****