Отчёт по домашнему заданию

Цель проекта:
Цель проекта — создать консольное приложение на языке Rust, которое взаимодействует с публичным API (OpenWeatherMap) для получения данных о погоде и отображения их пользователю.

Область применения:
Приложение будет использоваться для получения и отображения информации о погоде в заданном городе. Пользователь сможет вводить название города и получать текущую температуру, влажность, скорость ветра и краткое описание погоды.

Основные функции:
Запрос погоды: Пользователь вводит название города, приложение отправляет запрос к API и получает данные о погоде.
Отображение данных: Приложение выводит информацию о текущей температуре, влажности, скорости ветра и описании погоды.
Обработка ошибок: Если введено неправильное название города или произошла ошибка при запросе, приложение должно вывести понятное сообщение об ошибке.

Библиотеки:

- `reqwest` — для выполнения HTTP-запросов.
- `serde` и `serde_json` — для сериализации и десериализации JSON-данных.
- `anyhow` — для обработки ошибок и улучшения читаемости кода.

Архитектура:

main.rs:

Содержит основную логику приложения, включая инициализацию, обработку ввода и вывод данных.

api.rs:

Реализует функции для выполнения HTTP-запросов к API и обработки ответов, включая обработку ошибок.

models.rs:

Определяет структуры данных для хранения информации о погоде, такие как `WeatherResponse`, `Main`, `Wind` и т. д.

cli.rs:

Обрабатывает ввод командной строки, включая парсинг аргументов и взаимодействие с пользователем.

cache.rs:

Реализует функции для кэширования результатов запросов, включая сохранение и загрузку данных из файла.

Код программы:

```
//main.rs

mod api; //Отвечает за обработку запросов к OpenWeatherMap API
mod models; //Определяет структуры для работы с JSON
mod cli; //Отвечает за работу с пользователем
mod cache; //Отвечает за кэширование

use anyhow::Result;
use cli::get_user_input;
use api::fetch_weather_data;
use cache::Cache;


const CACHE_FILE: &str = "weather_cache.json";

fn main() -> Result<()> {
println!("Welcome to the Weather CLI App!");

// Загрузка кэша из файла
let mut cache = Cache::load_from_file(CACHE_FILE).unwrap_or_else(|_| Cache::new());

loop {
let city = get_user_input("Enter the city name: ")?;

// Проверяем, есть ли данные в кэше
if let Some(cached_data) = cache.get(&city) {
println!("\nCached Weather for {}:\n{}", city, cached_data);
} else {
// Данные не найдены в кэше, делаем запрос к API
match fetch_weather_data(&city) {
Ok(weather) => {
let weather_info = format!(
"Temperature: {}°C\nHumidity: {}%\nWind Speed: {} m/s\nDescription: {}",
weather.main.temp, weather.main.humidity, weather.wind.speed,
weather.weather[0].description
);

println!("\nWeather for {}:\n{}", city, weather_info);
```

```
// Сохраняем данные в кэш
cache.insert(city.clone(), weather_info);
cache.save_to_file(CACHE_FILE).unwrap_or_else(|err| {
eprintln!("Failed to save cache: {}", err);
});
}
Err(err) => {
eprintln!("Error fetching weather data: {}", err);
}
}
}

// Спрашиваем пользователя, хочет ли он продолжить
loop {
let continue_choice = get_user_input("Do you want to check another city? (yes/no): ")?;
match continue_choice.to_lowercase().as_str() {
"yes" => break,
"no" => {
println!("Goodbye!");
return Ok(());
}
_ => println!("Invalid input. Please type 'yes' or 'no'."),
}
}
}
}
//api.rs

use reqwest::blocking::get;
use serde_json::from_str;
use anyhow::{Result, Context};

use crate::models::WeatherResponse;

const API_URL: &str = "https://api.openweathermap.org/data/2.5/weather";
const API_KEY: &str = "d44c06d4464ffc840a547bcb50a434a2";

pub fn fetch_weather_data(city: &str) -> Result<WeatherResponse> {
let url = format!("{}?q={}&appid={}&units=metric", API_URL, city, API_KEY);
let response = get(&url).context("Failed to send request to the API")?;
let body = response.text().context("Failed to read response body")?;
let weather_data: WeatherResponse = from_str(&body).context("Failed to parse JSON response")?;
Ok(weather_data)
}
// cache.rs

use serde::{Deserialize, Serialize};
use std::collections::HashMap;
use std::fs;
use anyhow::{Result, Context};
```

```rust
#[derive(Debug, Serialize, Deserialize)]
pub struct Cache {
data: HashMap<String, String>,
}

impl Cache {
/// Создает новый пустой кэш
pub fn new() -> Self {
Self {
data: HashMap::new(),
}
}

/// Загружает кэш из файла
pub fn load_from_file(file_path: &str) -> Result<Self> {
let content = fs::read_to_string(file_path).unwrap_or_else(|_| "{}".to_string());
let cache: Cache = serde_json::from_str(&content).context("Failed to parse cache file")?;
Ok(cache)
}

/// Сохраняет кэш в файл
pub fn save_to_file(&self, file_path: &str) -> Result<()> {
let content = serde_json::to_string_pretty(self).context("Failed to serialize cache")?;
fs::write(file_path, content).context("Failed to write cache to file")?;
Ok(())
}

/// Извлекает данные из кэша
pub fn get(&self, key: &str) -> Option<&String> {
self.data.get(key)
}

/// Вставляет данные в кэш
pub fn insert(&mut self, key: String, value: String) {
self.data.insert(key, value);
}
}

//models.rs

use serde::Deserialize;

#[derive(Debug, Deserialize)]
pub struct WeatherResponse {
pub weather: Vec<Weather>,
pub main: Main,
pub wind: Wind,
}
```

```rust
#[derive(Debug, Deserialize)]
pub struct Weather {
pub description: String,
}

#[derive(Debug, Deserialize)]
pub struct Main {
pub temp: f64,
pub humidity: u64,
}

#[derive(Debug, Deserialize)]
pub struct Wind {
pub speed: f64,
}
//cli.rs

use std::io::{self, Write};
use anyhow::{Result, Context};

pub fn get_user_input(prompt: &str) -> Result<String> {
print!("{}", prompt);
io::stdout().flush().context("Failed to flush stdout")?;
let mut input = String::new();
io::stdin().read_line(&mut input).context("Failed to read input")?;
Ok(input.trim().to_string())
}
```

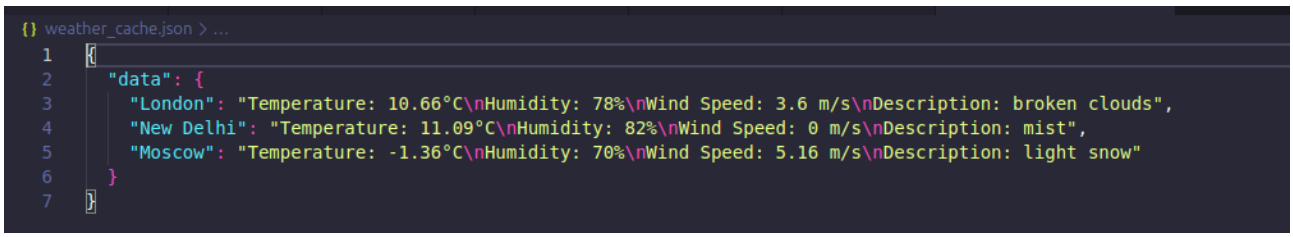
Пример выполнения программы:

```
vonrodinus@VonRodinus:~/Projects/simple_api_client$ cargo run
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.19s
     Running `target/debug/weather_cli`
Welcome to the Weather CLI App!
Enter the city name: Moscow

Weather for Moscow:
Temperature: -1.36°C
Humidity: 70%
Wind Speed: 5.16 m/s
Description: light snow
Do you want to check another city? (yes/no): yes
Enter the city name: London

Weather for London:
Temperature: 10.66°C
Humidity: 78%
Wind Speed: 3.6 m/s
Description: broken clouds
Do you want to check another city? (yes/no): yes
Enter the city name: New Delhi

Weather for New Delhi:
Temperature: 11.09°C
Humidity: 82%
Wind Speed: 0 m/s
Description: mist
Do you want to check another city? (yes/no): no
Goodbye!
vonrodinus@VonRodinus:~/Projects/simple_api_client$
```



json-файл, содержащий данные, сохраненные в кэше.