Министерство образования Тульской области

Государственное профессиональное образовательное учреждение Тульской области «Донской политехнический колледж»

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКИХ РАБОТ

по междисциплинарному курсу
«МДК.01.03 Разработка мобильных приложений»
по теме: «Разработка приложения с использованием внешних API»
для обучающихся по программе подготовки специалистов среднего звена
по специальности 09.02.07 Информационные системы и программирование,
квалификация «Программист»

Автор:

С.М. Гвоздев, преподаватель ГПОУ ТО «ДПК»

2024 г.

Лист согласования:

Автор разработки:

Гвоздев Сергей Михайлович, преподаватель ГПОУ ТО «ДПК»

Рецензенты:

Евтехова О.А., заместитель директора по учебной и научно-методической работе ГПОУ ТО «ДПК»

Панченко Т.А., заместитель директора по организации образовательного процесса ГПОУ ТО «ДПК».

Филатова Е.А., старший методист ГПОУ ТО «ДПК».

Методические рекомендации предназначены для студентов 3 курса, обучающихся по специальности 09.02.07 Информационные системы и программирование, квалификация «Программист». Конкретные примеры данного пособия окажут практическую помощь при выполнении практических заданий по дисциплине «МДК.01.03 Разработка мобильных приложений»: «Разработка приложения с использованием внешних API» с использованием онлайн-редактора кода Dartpad.

СОГЛАСОВАНО

на заседании предметной (цикловой) комиссии
дисциплин профессионального цикла отделения
«Информационная безопасность и администрирование»
Протокол № 2

от «03» октября 2024 г.

Председатель ПЦК Гвоздев С.М.

СОДЕРЖАНИЕ

Введение

Методические рекомендации составлены в соответствии с рабочей программой ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем» специальности 09.02.07 Информационные системы и программирование, квалификация «Программист».

В ходе освоения профессионального модуля обучающийся должен:

знать:

- основные этапы разработки программного обеспечения;
- основные принципы технологии структурного и объектно-ориентированного программирования;

уметь:

- разрабатывать приложения на языке Dart с использованием внешних API.

В процессе обучения по «МДК.01.03 Разработка мобильных приложений» у студентов формируется комплексное понимание и практические навыки, необходимые для создания функциональных и удобных мобильных приложений для различных платформ, таких как iOS и Android. Это достигается через изучение теоретических основ, включая архитектуру мобильных приложений, принципы работы мобильных операционных систем, основные компоненты приложений, такие как пользовательский интерфейс и бизнес-логика, а также различные паттерны проектирования.

На практике студенты осваивают инструменты и технологии разработки, включая языки программирования, интегрированные среды разработки (IDE), системы управления версиями и другие полезные инструменты. Они учатся проектировать интуитивно понятный и привлекательный пользовательский интерфейс, оптимизировать производительность приложений, обеспечивать безопасность данных и взаимодействие с различными сервисами и API.

Кроме того, студенты получают навыки тестирования и отладки мобильных приложений, а также знакомятся с процессом публикации приложений в магазинах приложений и монетизации. В целом, цель дисциплины – подготовить студентов к успешной карьере в области разработки мобильных приложений, обеспечив их глубоким пониманием технологий и практическими навыками, необходимыми для создания качественных и востребованных продуктов.

Задание № 1: Получение данных о погоде

Напишите программу, которая получает данные о погоде в вашем городе с использованием *API* *OpenWeatherMap*. Отобразите температуру, описание погоды и иконку.

Описание алгоритма решения:

1. Зарегистрируйтесь на *OpenWeatherMap* и получите *API ключ*.
2. Используйте библиотеку *http* для отправки *GET* запроса к *API*.
3. Распарсите *JSON* ответ и извлеките необходимые данные.
4. Используйте *print* для отображения информации

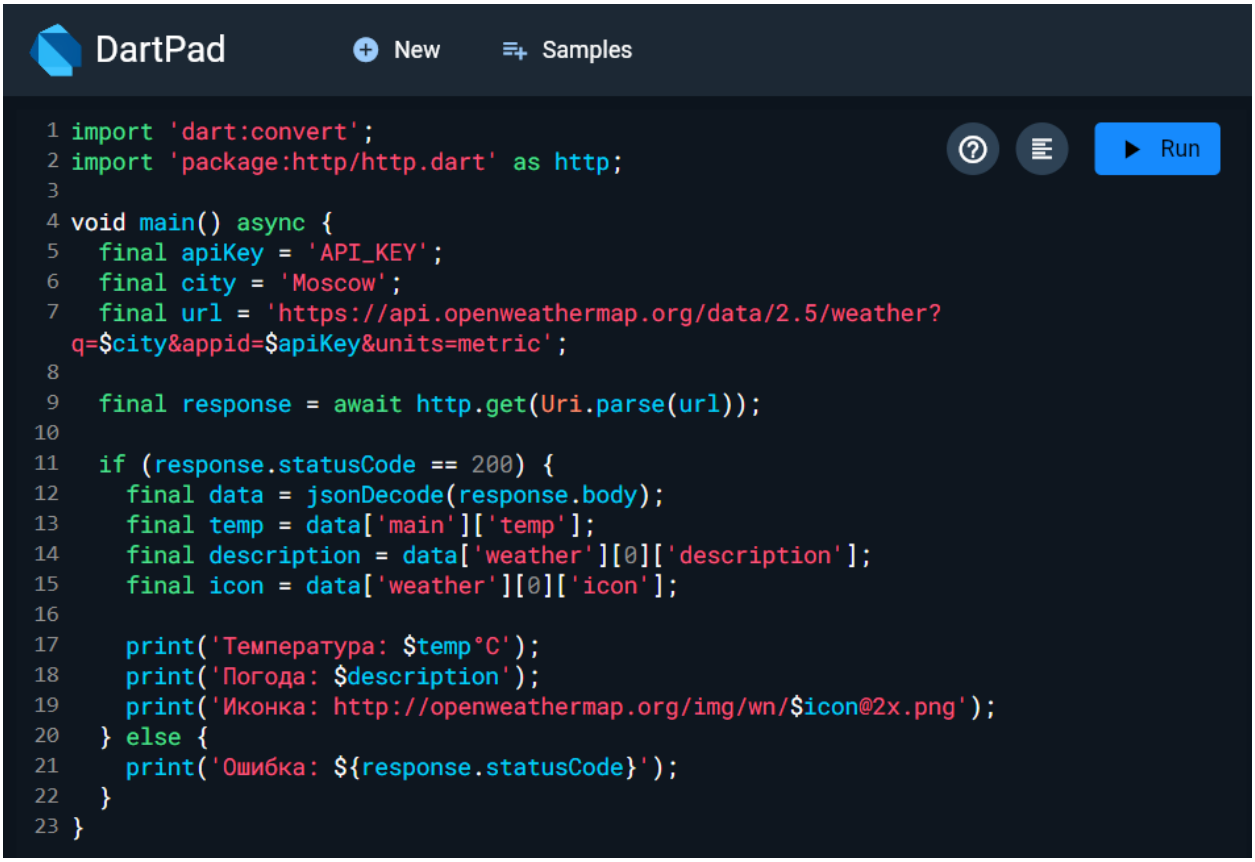


Рисунок №1 – Результат решения задания №1.

Задание № 2: Поиск фильмов

Напишите программу, которая позволяет пользователю искать фильмы по названию с использованием *API* *OMDB*. Отобразите название, год выпуска и постер.

Описание алгоритма решения:

1. Зарегистрируйтесь на *OMDB* и получите *API ключ*.
2. Создайте текстовое поле для ввода названия фильма.
3. Используйте библиотеку *http* для отправки *GET* запроса к *API* с параметром поиска.
4. Распарсите *JSON* ответ и извлеките необходимые данные.
5. Используйте *print* для отображения информации.

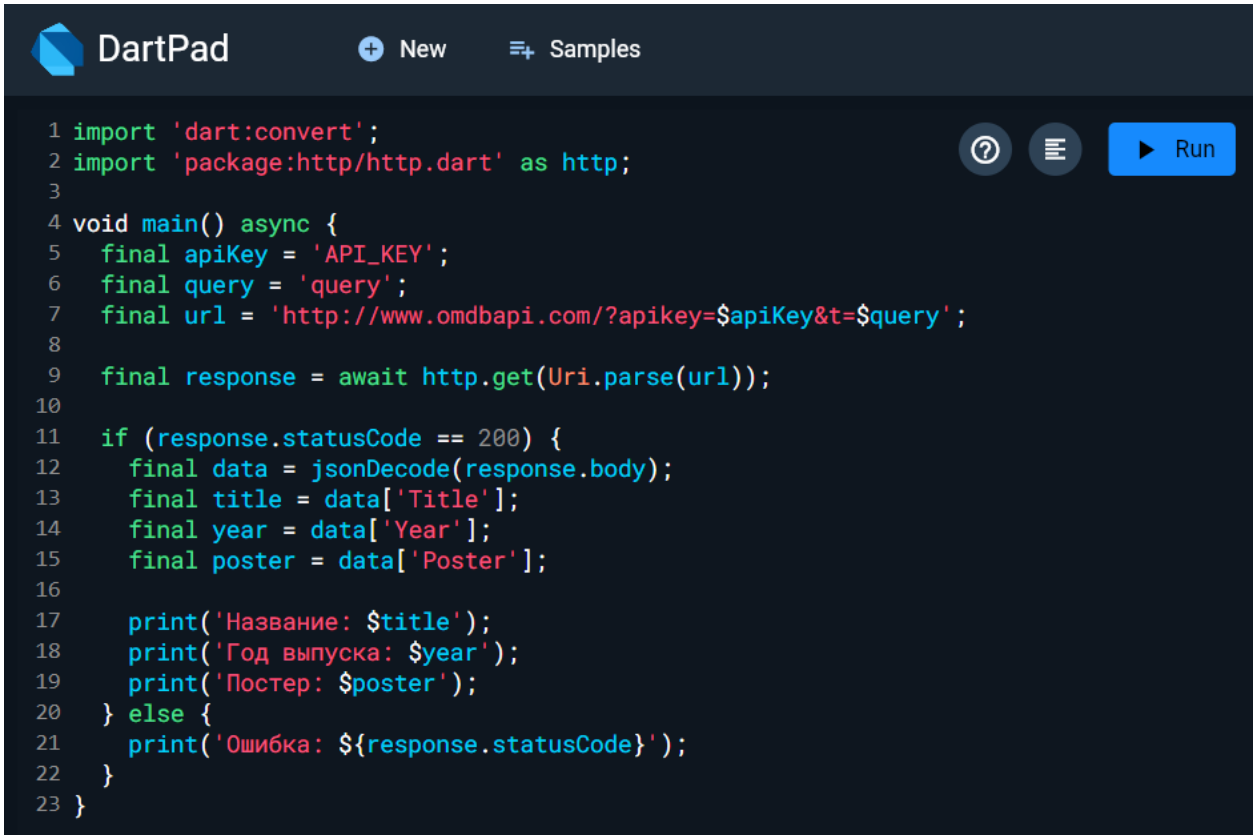


Рисунок №2 – Результат решения задания №2.

Задание № 3: Получение курса валют

Напишите программу, которая получает курс валют с использованием *API Exchange Rates*. Отобразите курс *USD* к *EUR*.

Описание алгоритма решения:

1. Зарегистрируйтесь на *Exchange Rates* и получите *API ключ*.
2. Используйте библиотеку *http* для отправки *GET* запроса к *API*.
3. Распарсите *JSON* ответ и извлеките курс *USD* к *EUR*.
4. Используйте *print* для отображения информации

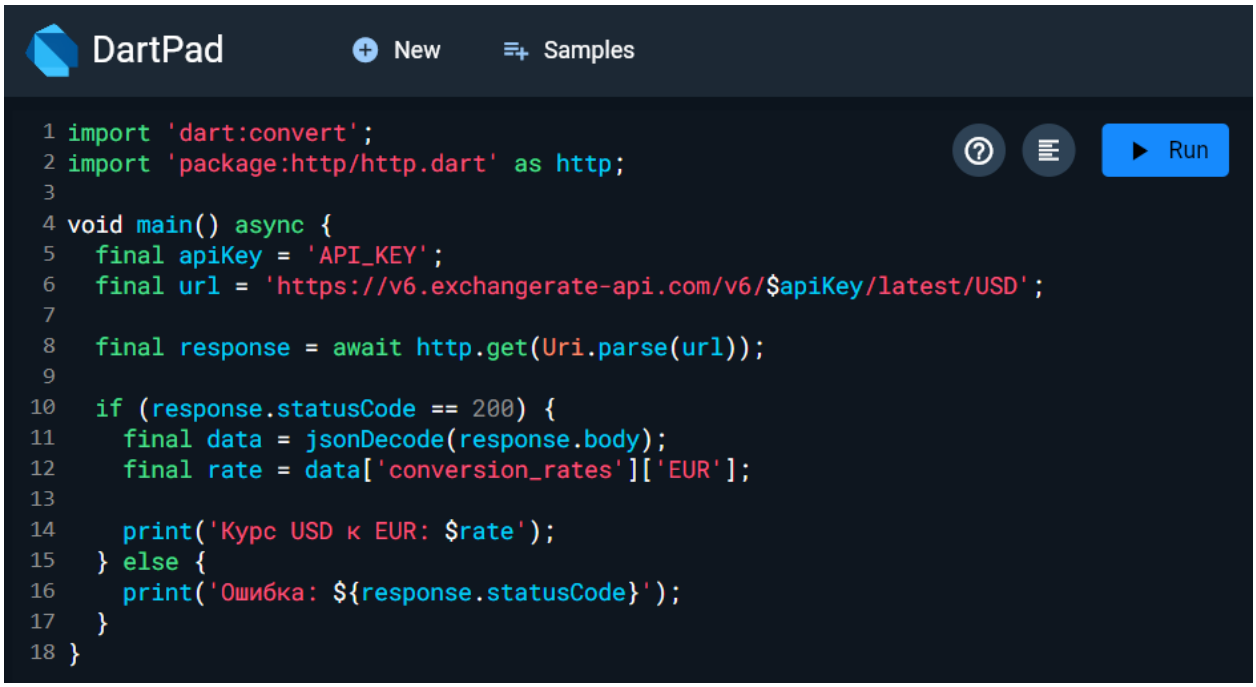


Рисунок №3 – Результат решения задания №3.

Задание № 4: Получение данных о GitHub пользователе

Напишите программу, которая получает данные о *GitHub* пользователе по его имени пользователя. Отобразите *имя*, *био* и количество репозиториев.

Описание алгоритма решения:

1. Используйте библиотеку *http* для отправки *GET* запроса к *API GitHub*.
2. Распарсите *JSON* ответ и извлеките необходимые данные.
3. Используйте *print* для отображения информации.

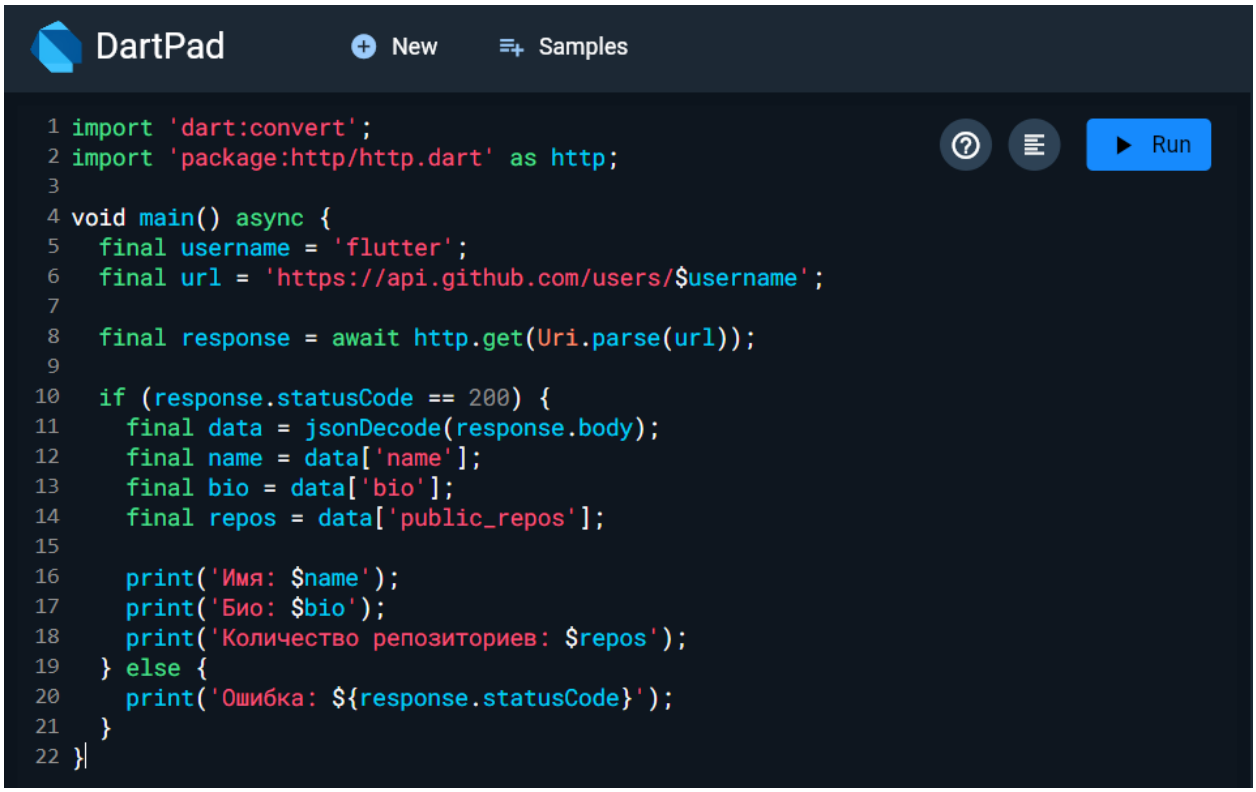


Рисунок №4 – Результат решения задания №4.

Задание № 5: Получение новостей

Напишите программу, которая получает последние новости с использованием *API* *NewsAPI*. Отобразите заголовок, описание и ссылку на новость.

Описание алгоритма решения:

1. Зарегистрируйтесь на *NewsAPI* и получите *API ключ*.
2. Используйте библиотеку *http* для отправки *GET* запроса к *API*.
3. Распарсите *JSON* ответ и извлеките необходимые данные.
4. Используйте *print* для отображения информации

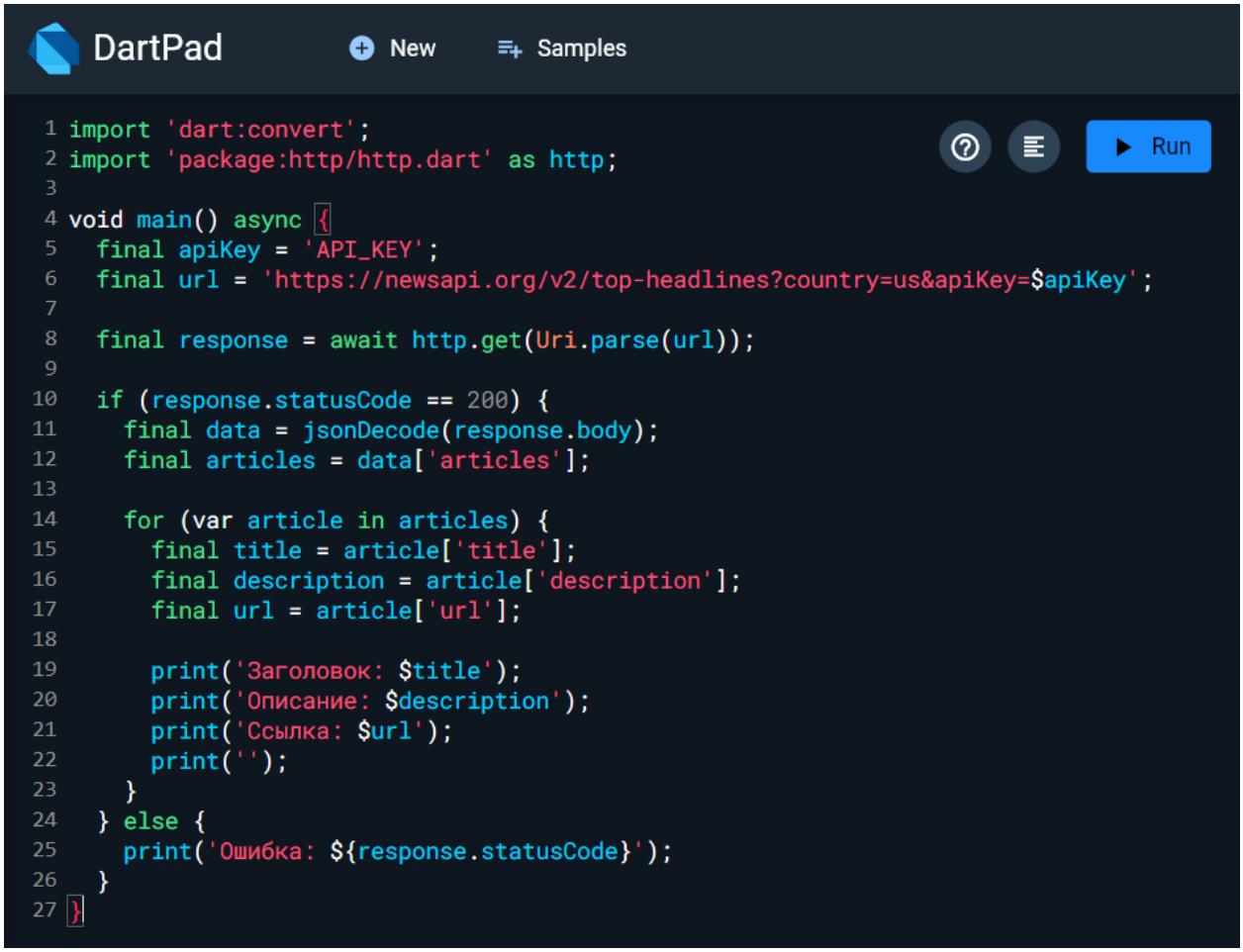


Рисунок №5 – Результат решения задания №5.

Задание № 6: Получение данных о погоде с использованием библиотеки dio

Перепишите задание с использованием библиотеки *dio* вместо *http*.

Описание алгоритма решения:

1. Добавьте зависимость *dio* в *pubspec.yaml*.
2. Используйте *Dio* для отправки *GET* запроса к *API*.
3. Распарсите *JSON* ответ и извлеките необходимые данные.
4. Используйте *print* для отображения информации.

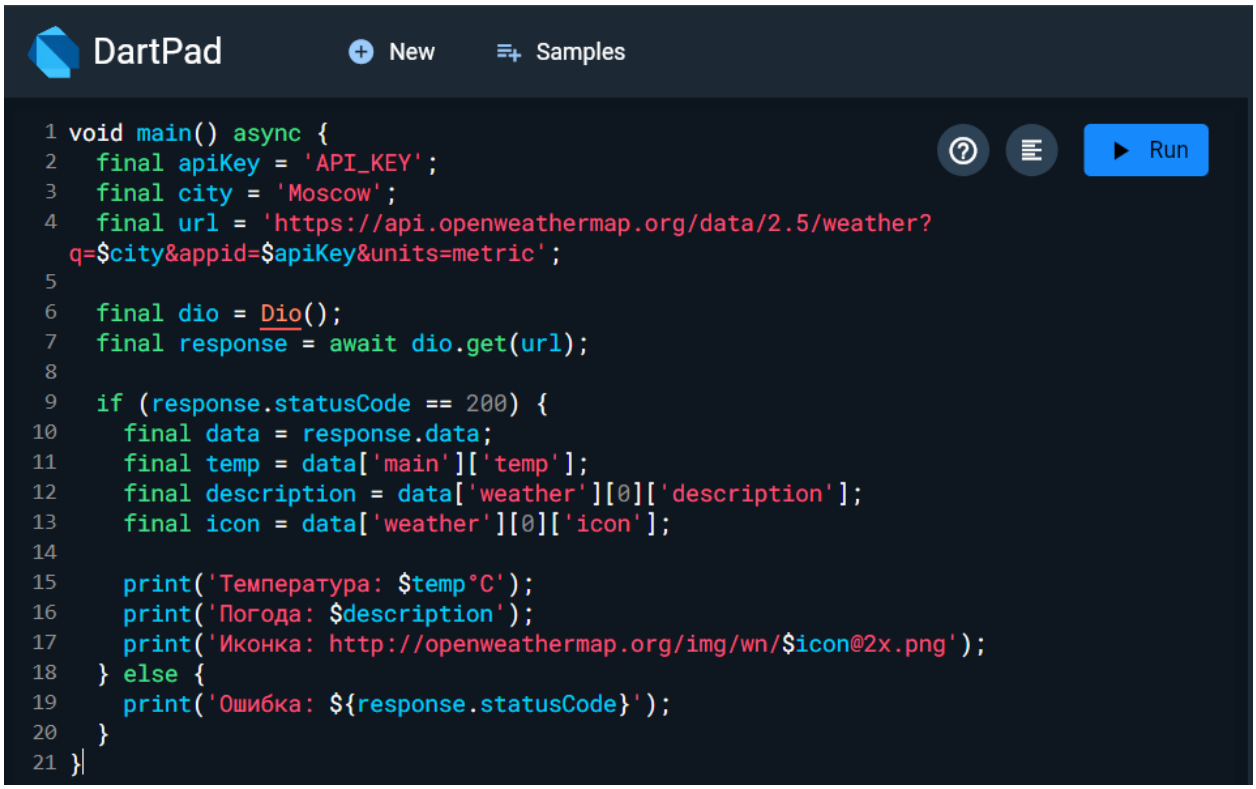


Рисунок №6 – Результат решения задания №6.

Задание № 7: Получение данных о фильме с использованием библиотеки dio

Перепишите задание с использованием библиотеки *dio* вместо *http*.

Описание алгоритма решения:

1. Добавьте зависимость *dio* в *pubspec.yaml*.
2. Используйте *Dio* для отправки *GET* запроса к *API*.
3. Распарсите *JSON* ответ и извлеките необходимые данные.
4. Используйте *print* для отображения информации

```dart
void main() async {
  final apiKey = 'API_KEY';
  final query = 'query';
  final url = 'http://www.omdbapi.com/?apikey=$apiKey&t=$query';

  final dio = Dio();
  final response = await dio.get(url);

  if (response.statusCode == 200) {
    final data = response.data;
    final title = data['Title'];
    final year = data['Year'];
    final poster = data['Poster'];

    print('Название: $title');
    print('Год выпуска: $year');
    print('Постер: $poster');
  } else {
    print('Ошибка: ${response.statusCode}');
  }
}
```

Рисунок №7 – Результат решения задания №7.

Задание № 8: Получение курса валют с использованием библиотеки dio

Перепишите задание с использованием библиотеки *dio* вместо *http*.

Описание алгоритма решения:

1. Добавьте зависимость *dio* в *pubspec.yaml*.
2. Используйте *Dio* для отправки *GET* запроса к *API*.
3. Распарсите *JSON* ответ и извлеките необходимые данные.
4. Используйте *print* для отображения информации

```dart
void main() async {
  final apiKey = 'API_KEY';
  final url = 'https://v6.exchangerate-api.com/v6/$apiKey/latest/USD';

  final dio = Dio();
  final response = await dio.get(url);

  if (response.statusCode == 200) {
    final data = response.data;
    final rate = data['conversion_rates']['EUR'];

    print('Курс USD к EUR: $rate');
  } else {
    print('Ошибка: ${response.statusCode}');
  }
}
```

Рисунок №8 – Результат решения задания №8.

Задание № 9: Получение новостей с использованием библиотеки dio

Перепишите задание с использованием библиотеки *dio* вместо *http*.

Описание алгоритма решения:

1. Добавьте зависимость *dio* в *pubspec.yaml*.
2. Используйте *Dio* для отправки *GET* запроса к *API*.
3. Распарсите *JSON* ответ и извлеките необходимые данные.
4. Используйте *print* для отображения информации.

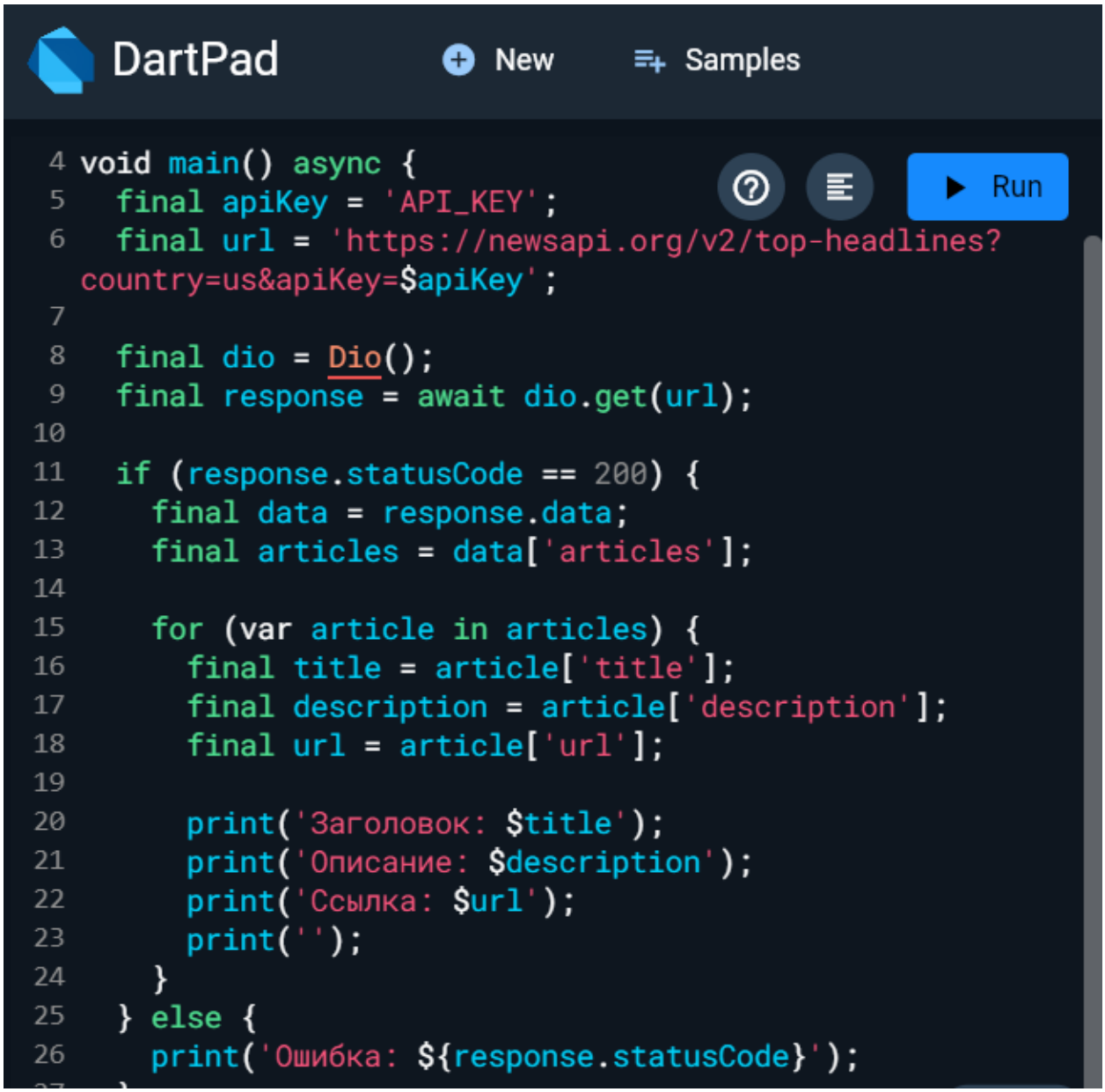


Рисунок №9 – Результат решения задания №9.

Задание № 10: Получение данных о GitHub пользователе с использованием библиотеки dio

Перепишите задание с использованием библиотеки *dio* вместо *http*.

Описание алгоритма решения:

1. Добавьте зависимость *dio* в *pubspec.yaml*.
2. Используйте *Dio* для отправки *GET* запроса к *API*.
3. Распарсите *JSON* ответ и извлеките необходимые данные.
4. Используйте *print* для отображения информации.

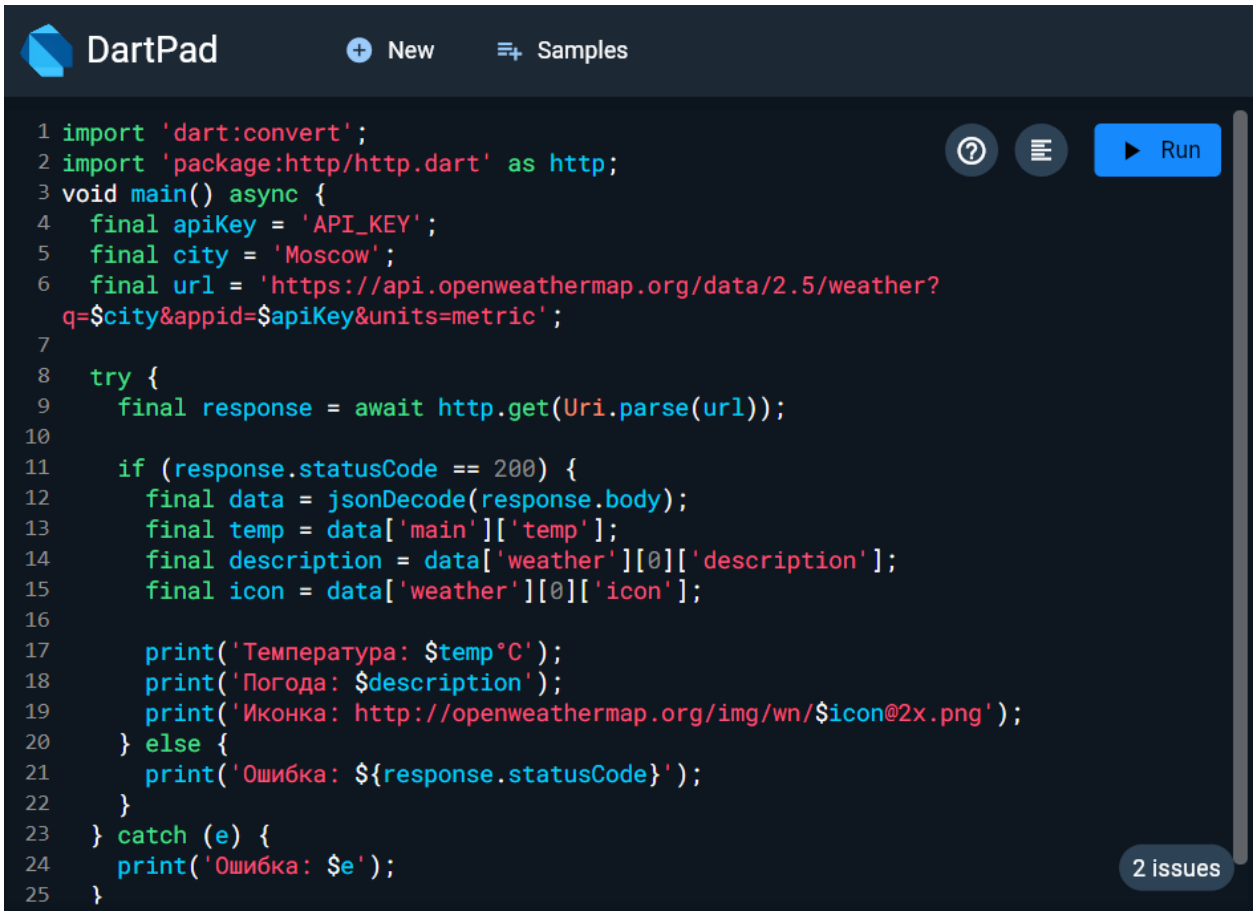
Рисунок №10 – Результат решения задания №10.

Задание № 11: Получение данных о погоде с использованием библиотеки http и обработкой ошибок

Перепишите задание с добавлением обработки ошибок.

Описание алгоритма решения:

1. Используйте try-catch для обработки возможных ошибок.
2. В блоке catch выведите сообщение об ошибке




```dart
import 'dart:convert';
import 'package:http/http.dart' as http;
void main() async {
  final apiKey = 'API_KEY';
  final city = 'Moscow';
  final url = 'https://api.openweathermap.org/data/2.5/weather?q=$city&appid=$apiKey&units=metric';

  try {
    final response = await http.get(Uri.parse(url));

    if (response.statusCode == 200) {
      final data = jsonDecode(response.body);
      final temp = data['main']['temp'];
      final description = data['weather'][0]['description'];
      final icon = data['weather'][0]['icon'];

      print('Температура: $temp°C');
      print('Погода: $description');
      print('Иконка: http://openweathermap.org/img/wn/$icon@2x.png');
    } else {
      print('Ошибка: ${response.statusCode}');
    }
  } catch (e) {
    print('Ошибка: $e');
  }
```

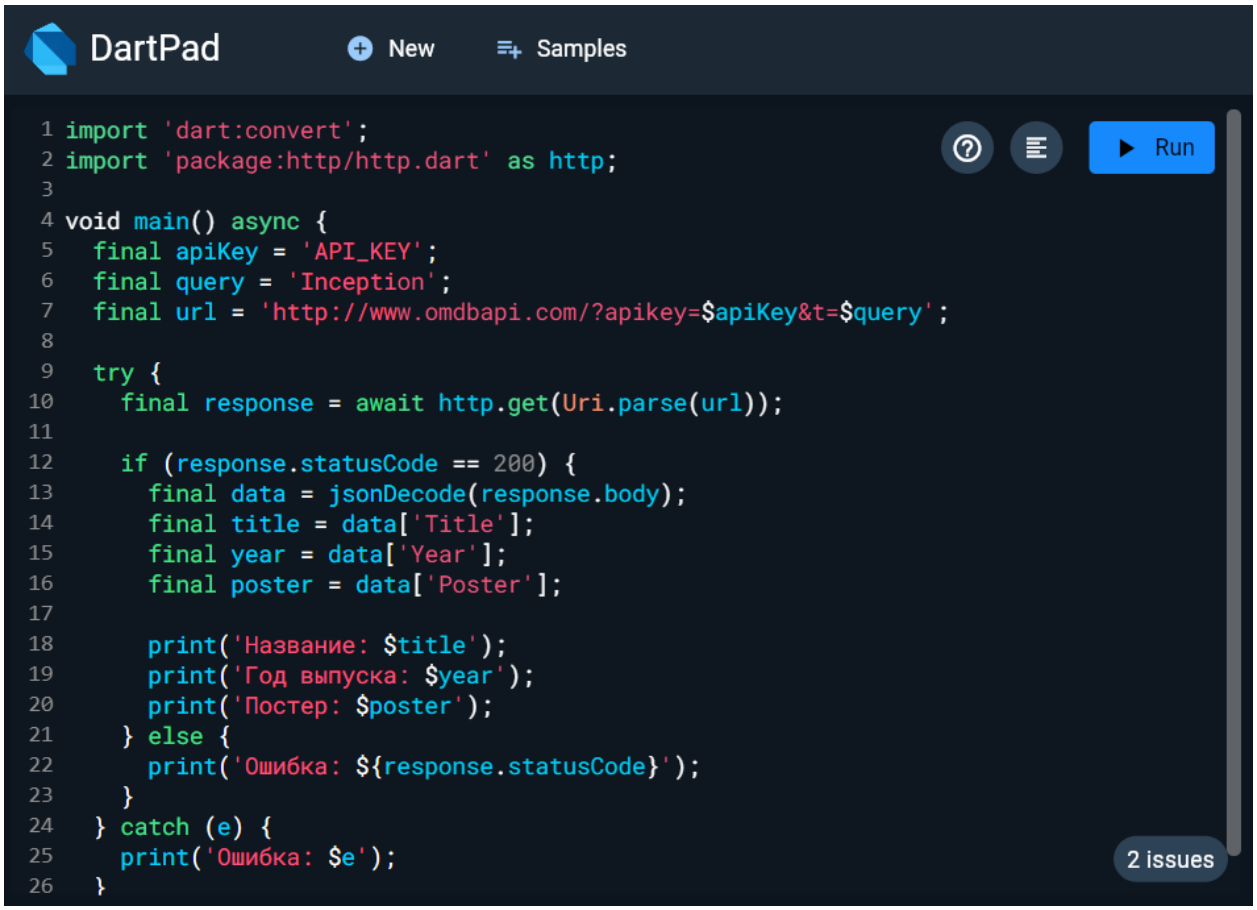
Рисунок №11 – Результат решения задания №11.

Задание № 12: Получение данных о фильме с использованием библиотеки http и обработкой ошибок

Перепишите задание с добавлением обработки ошибок.

Описание алгоритма решения:

1. Используйте try-catch для обработки возможных ошибок.
2. В блоке catch выведите сообщение об ошибке




```dart
import 'dart:convert';
import 'package:http/http.dart' as http;

void main() async {
  final apiKey = 'API_KEY';
  final query = 'Inception';
  final url = 'http://www.omdbapi.com/?apikey=$apiKey&t=$query';

  try {
    final response = await http.get(Uri.parse(url));

    if (response.statusCode == 200) {
      final data = jsonDecode(response.body);
      final title = data['Title'];
      final year = data['Year'];
      final poster = data['Poster'];

      print('Название: $title');
      print('Год выпуска: $year');
      print('Постер: $poster');
    } else {
      print('Ошибка: ${response.statusCode}');
    }
  } catch (e) {
    print('Ошибка: $e');
  }
```

Рисунок №12 – Результат решения задания №12.

Задание № 13: Получение курса валют с использованием библиотеки http и обработкой ошибок

Перепишите задание с добавлением обработки ошибок.

Описание алгоритма решения:

1. Используйте *try-catch* для обработки возможных ошибок.
2. В блоке *catch* выведите сообщение об ошибке.

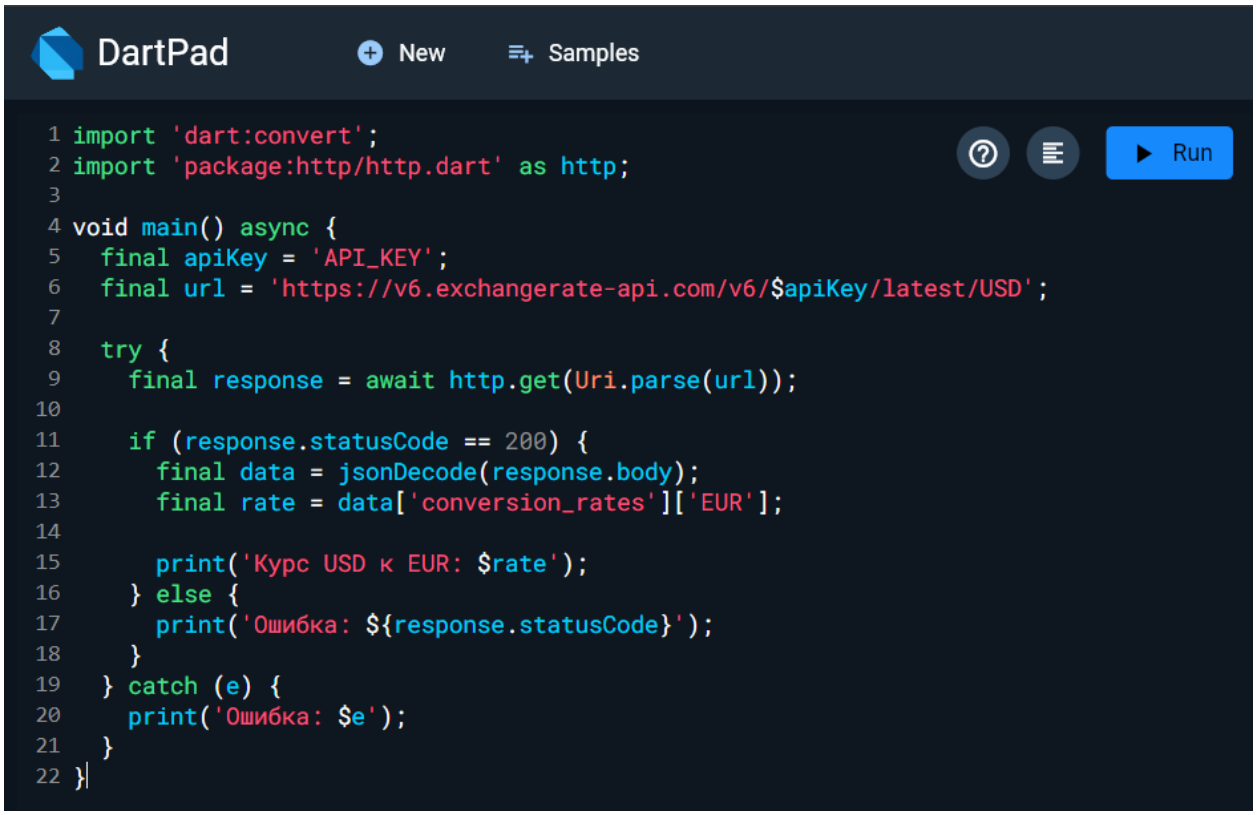


Рисунок №13 – Результат решения задания №13.

Задание № 14: Получение новостей с использованием библиотеки http и обработкой ошибок

Перепишите задание с добавлением обработки ошибок.

Описание алгоритма решения:

1. Используйте *try-catch* для обработки возможных ошибок.
2. В блоке *catch* выведите сообщение об ошибке.

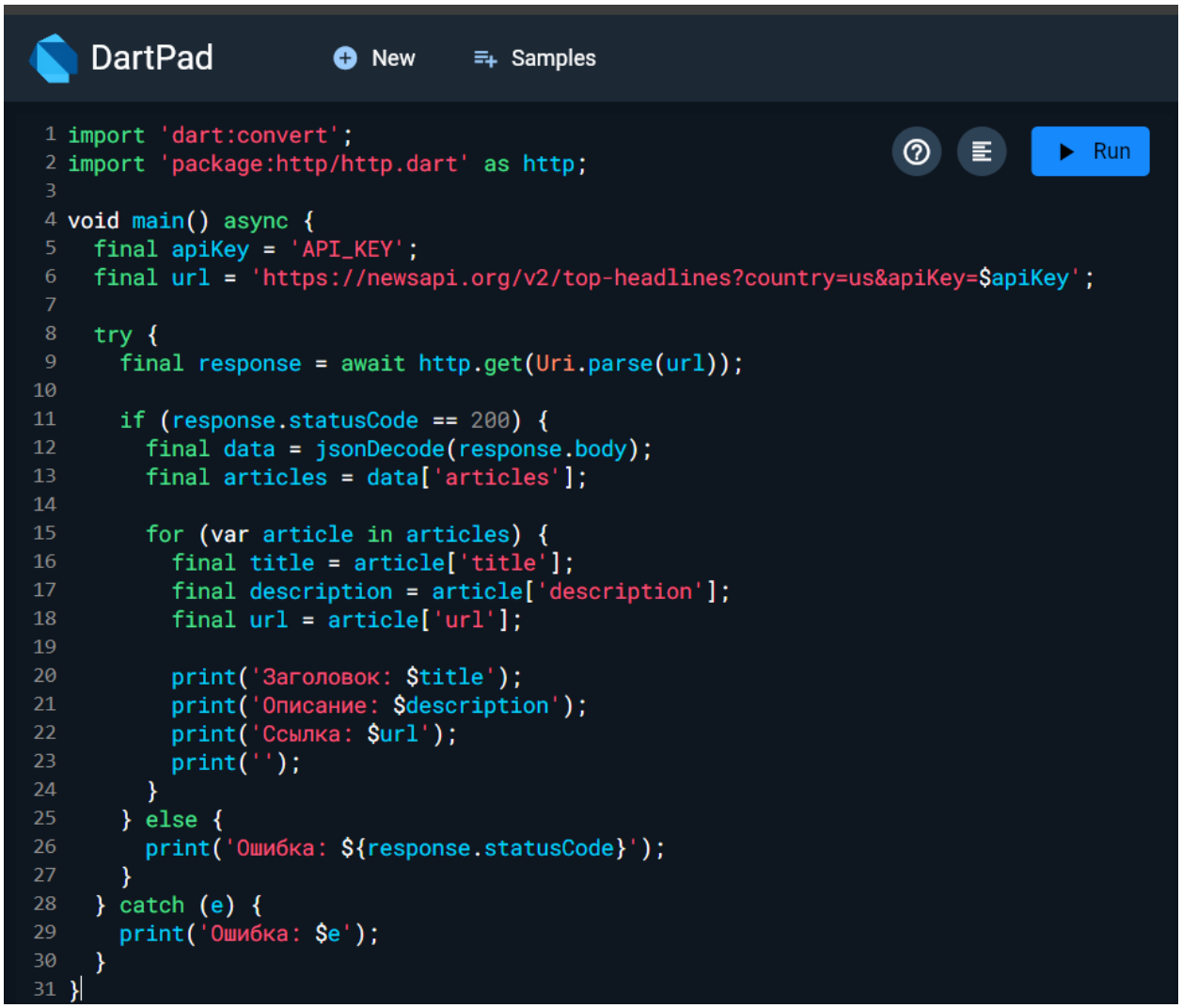


Рисунок №14 – Результат решения задания №14.

Задание № 15: Получение данных о GitHub пользователе с использованием библиотеки http и обработкой ошибок

Перепишите задание с добавлением обработки ошибок.

Описание алгоритма решения:

1. Используйте *try-catch* для обработки возможных ошибок.
2. В блоке *catch* выведите сообщение об ошибке

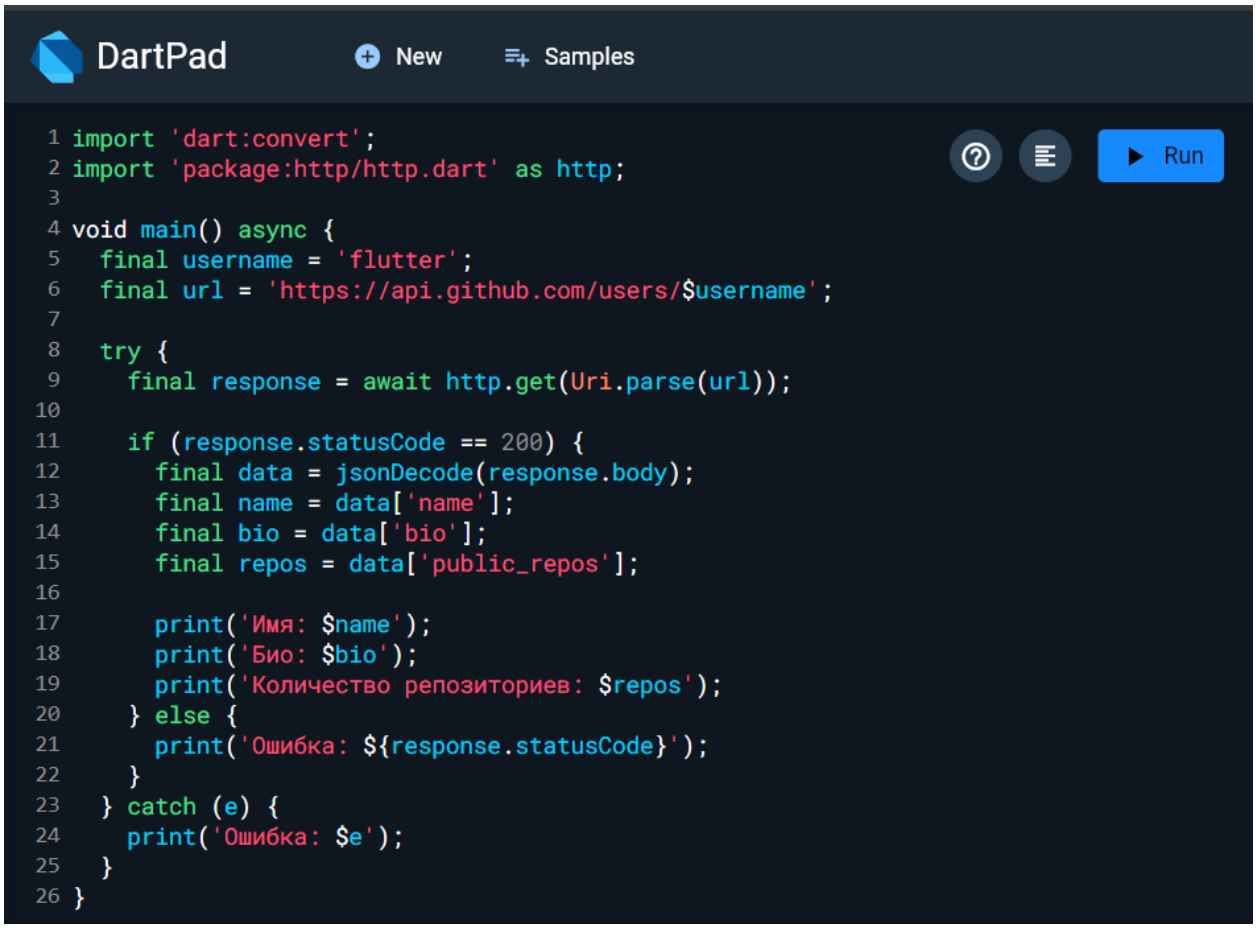


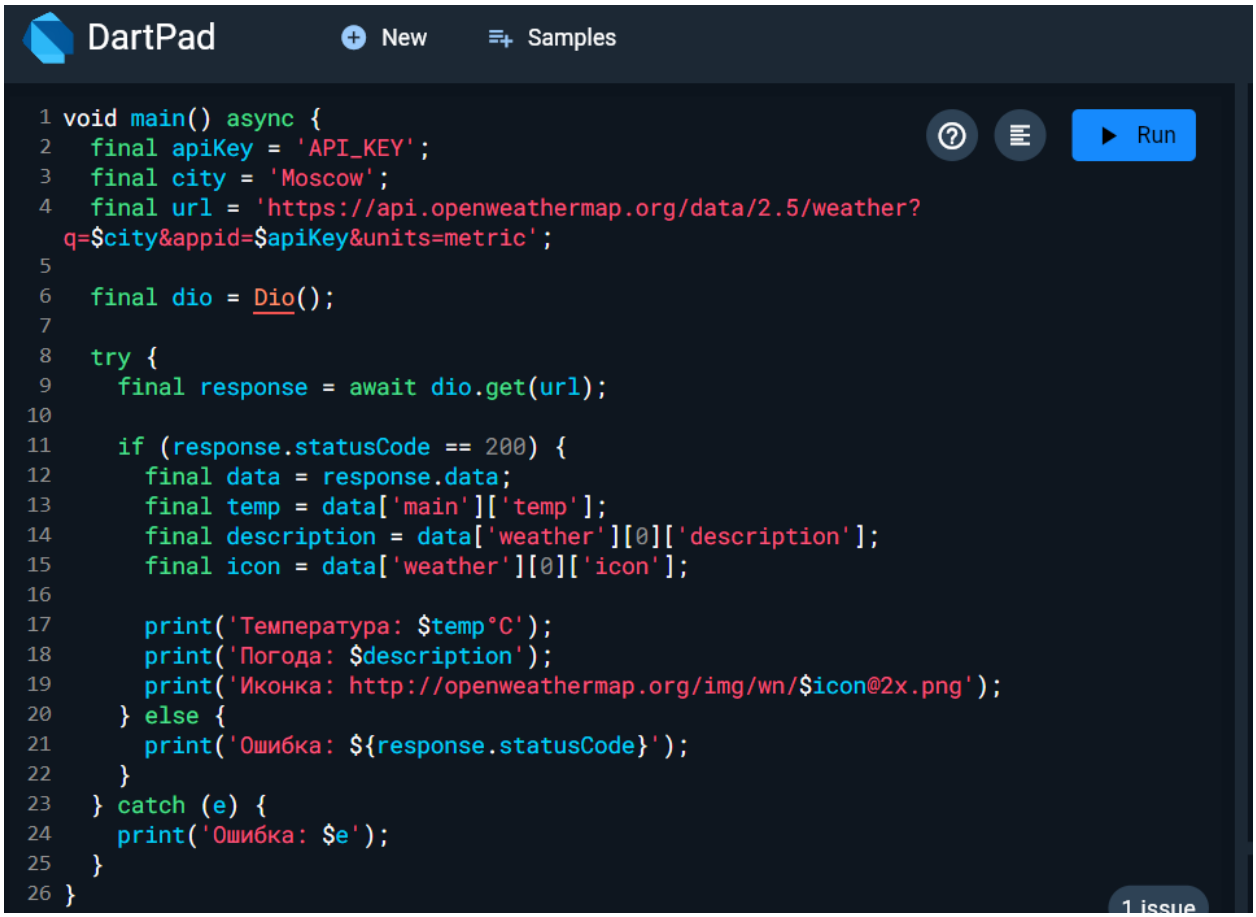
Рисунок №15 – Результат решения задания №15.

Задание № 16: Получение данных о погоде с использованием библиотеки dio и обработкой ошибок

Перепишите задание с добавлением обработки ошибок.

Описание алгоритма решения:

1. Используйте *try-catch* для обработки возможных ошибок.
2. В блоке *catch* выведите сообщение об ошибке




```dart
void main() async {
  final apiKey = 'API_KEY';
  final city = 'Moscow';
  final url = 'https://api.openweathermap.org/data/2.5/weather?
q=$city&appid=$apiKey&units=metric';

  final dio = Dio();

  try {
    final response = await dio.get(url);

    if (response.statusCode == 200) {
      final data = response.data;
      final temp = data['main']['temp'];
      final description = data['weather'][0]['description'];
      final icon = data['weather'][0]['icon'];

      print('Температура: $temp°C');
      print('Погода: $description');
      print('Иконка: http://openweathermap.org/img/wn/$icon@2x.png');
    } else {
      print('Ошибка: ${response.statusCode}');
    }
  } catch (e) {
    print('Ошибка: $e');
  }
}
```

Рисунок №16 – Результат решения задания №16.

Задание № 17: Получение данных о фильме с использованием библиотеки dio и обработкой ошибок

Перепишите задание с добавлением обработки ошибок.

Описание алгоритма решения:

1. Используйте try-catch для обработки возможных ошибок.
2. В блоке catch выведите сообщение об ошибке.

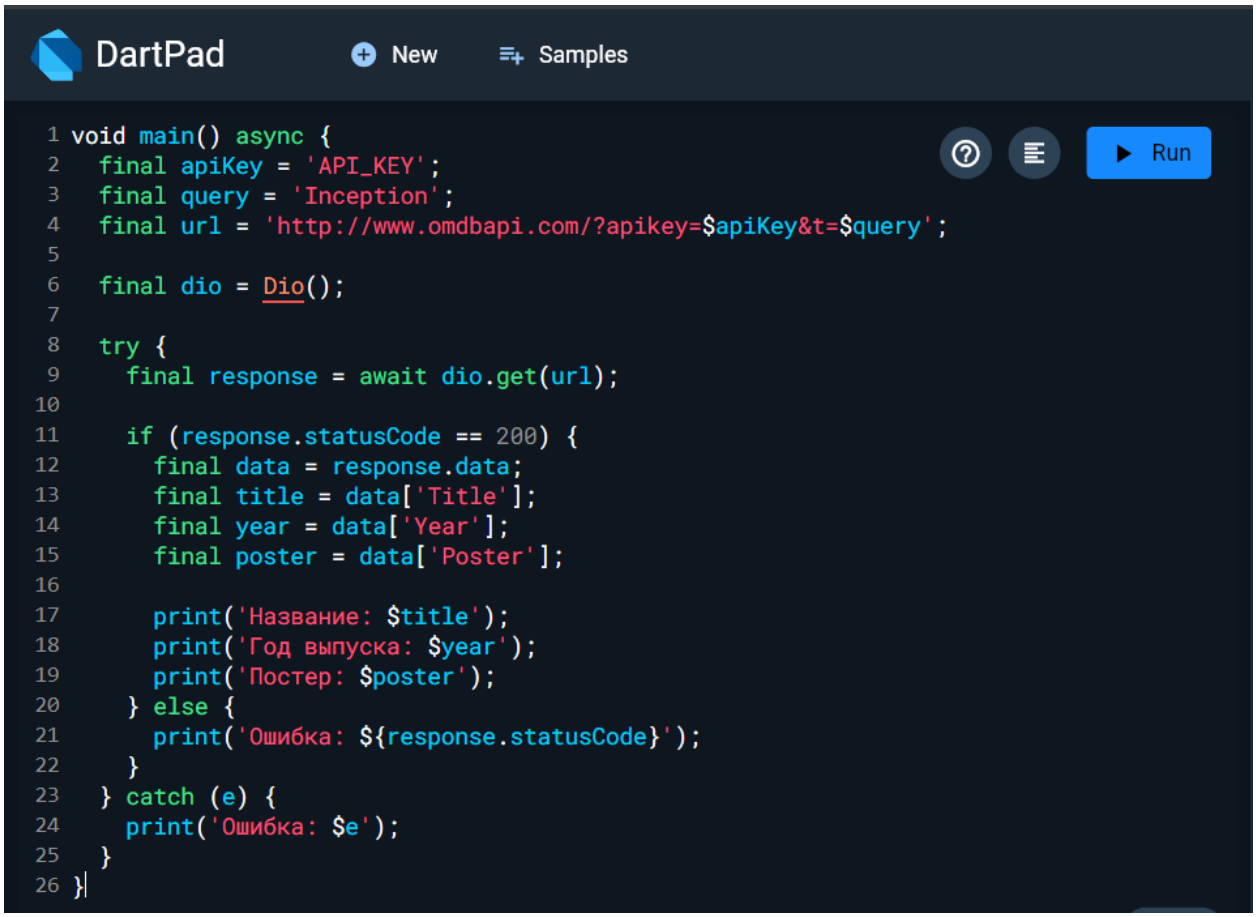


Рисунок №17 – Результат решения задания №17.

Задание № 18: Получение курса валют с использованием библиотеки dio и обработкой ошибок

Перепишите задание с добавлением обработки ошибок.

Описание алгоритма решения:

1. Используйте try-catch для обработки возможных ошибок.
2. В блоке catch выведите сообщение об ошибке.

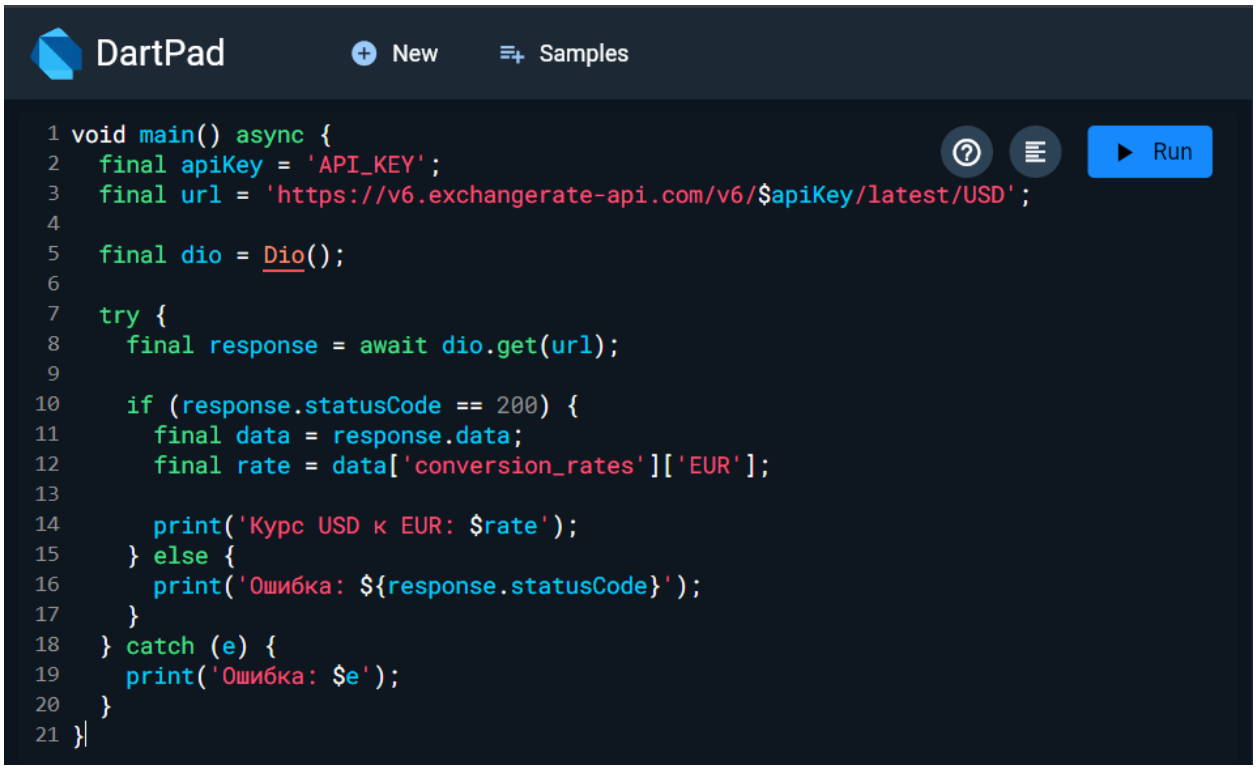


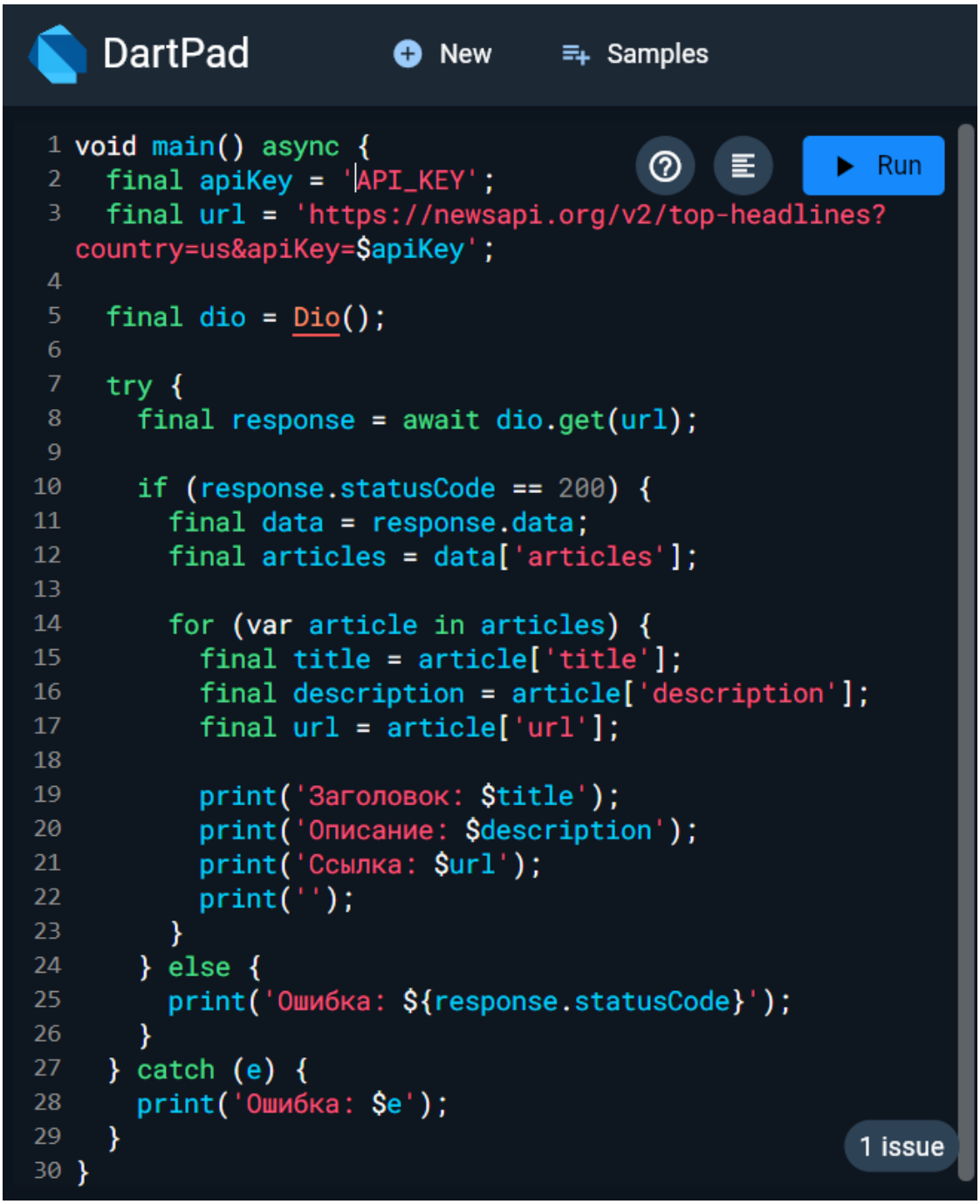
Рисунок №18 – Результат решения задания №18.

Задание № 19: Получение новостей с использованием библиотеки dio и обработкой ошибок

Перепишите задание с добавлением обработки ошибок.

Описание алгоритма решения:

1. Используйте try-catch для обработки возможных ошибок.
2. В блоке catch выведите сообщение об ошибке




```dart
void main() async {
  final apiKey = 'API_KEY';
  final url = 'https://newsapi.org/v2/top-headlines?country=us&apiKey=$apiKey';

  final dio = Dio();

  try {
    final response = await dio.get(url);

    if (response.statusCode == 200) {
      final data = response.data;
      final articles = data['articles'];

      for (var article in articles) {
        final title = article['title'];
        final description = article['description'];
        final url = article['url'];

        print('Заголовок: $title');
        print('Описание: $description');
        print('Ссылка: $url');
        print('');
      }
    } else {
      print('Ошибка: ${response.statusCode}');
    }
  } catch (e) {
    print('Ошибка: $e');
  }
}
```

Рисунок №19 – Результат решения задания №19.

Задание № 20: Получение данных о GitHub пользователе с использованием библиотеки dio и обработкой ошибок

Перепишите задание 10 с добавлением обработки ошибок.

Описание алгоритма решения:

1. Используйте try-catch для обработки возможных ошибок.
2. В блоке catch выведите сообщение об ошибке

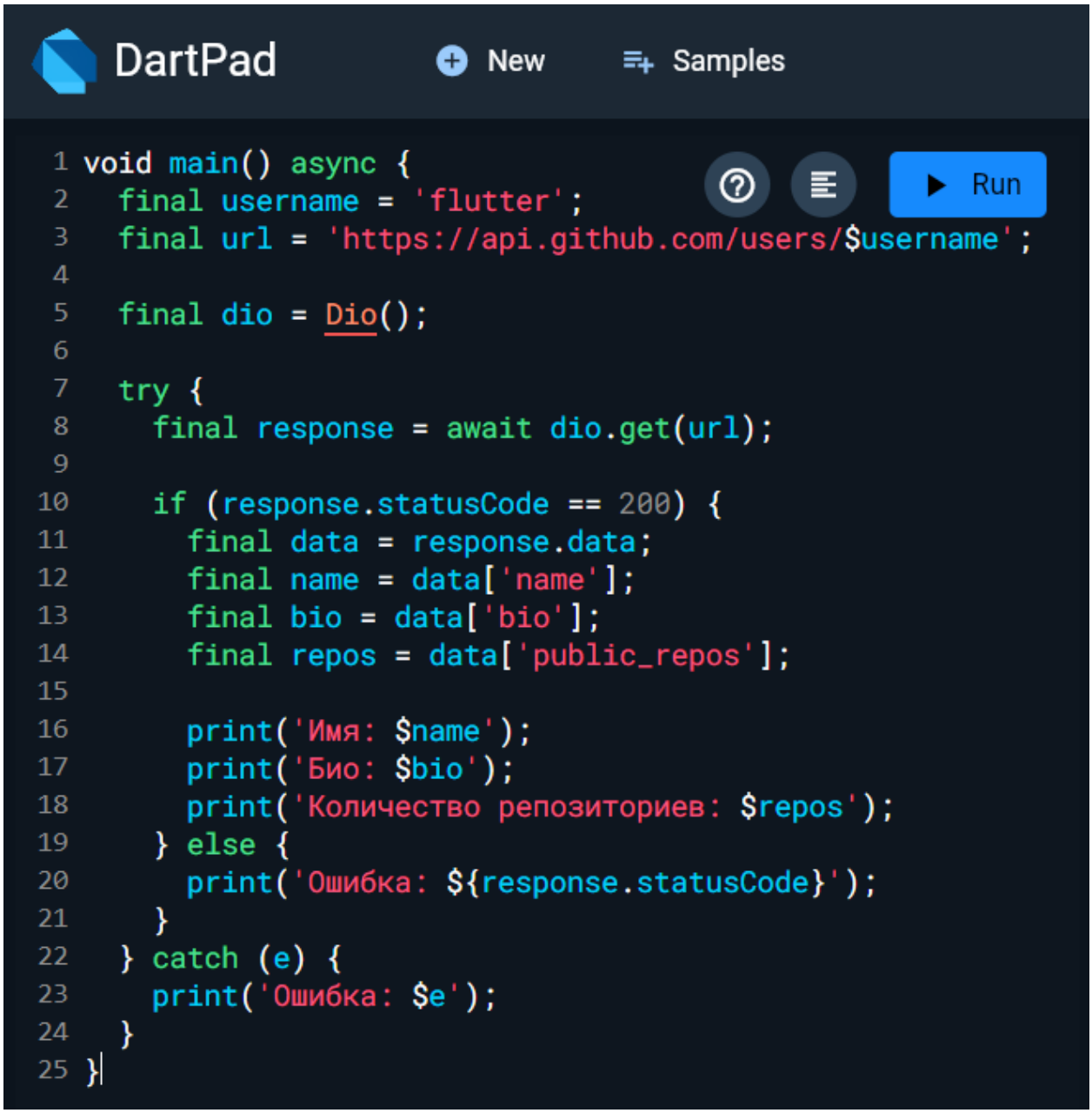


Рисунок №20 – Результат решения задания №20.

Индивидуальные задания для закрепления материала

1. **Генератор случайных цитат**:
   Создайте приложение, которое получает случайную цитату с использованием API, например, https://api.quotable.io/random. Отобразите цитату и автора.
2. **Поиск книг**:
   Разработайте приложение, которое позволяет пользователю искать книги по названию с использованием API Google Books. Отобразите название, автора и обложку книги.
3. **Анализ настроения текста**:
   Создайте приложение, которое анализирует настроение текста, введенного пользователем, с использованием API, например, https://api.deepai.org/api/sentiment-analysis. Отобразите результат анализа.
4. **Переводчик**:
   Разработайте приложение, которое переводит текст с одного языка на другой с использованием API, например, https://libretranslate.com/translate.
5. **Генератор случайных картинок**:
   Создайте приложение, которое получает случайную картинку с использованием API, например, https://picsum.photos/200/300. Отобразите картинку.
6. **Поиск GIF-изображений**:
   Разработайте приложение, которое позволяет пользователю искать GIF-изображения по ключевому слову с использованием API, например, https://api.giphy.com/v1/gifs/search. Отобразите найденные GIF-изображения.
7. **Прогноз погоды на неделю**:
   Создайте приложение, которое получает прогноз погоды на неделю с использованием API OpenWeatherMap. Отобразите прогноз для каждого дня.
8. **Поиск рецептов**:
   Разработайте приложение, которое позволяет пользователю искать рецепты по ингредиентам с использованием API, например, https://api.spoonacular.com/recipes/findByIngredients. Отобразите найденные рецепты.
9. **Анализ новостей**:
   Создайте приложение, которое анализирует новости на определенную тему с использованием API NewsAPI и отображает статистику, например, количество новостей, ключевые слова и т.д.

10. **Генератор случайных фактов**:
    Разработайте приложение, которое получает случайный факт с использованием API, например, https://uselessfacts.jsph.pl/random.json. Отобразите факт.
11. **Поиск музыки**:
    Создайте приложение, которое позволяет пользователю искать музыкальные треки по названию с использованием API, например, https://api.deezer.com/search. Отобразите найденные треки.
12. **Анализ тональности текста**:
    Разработайте приложение, которое анализирует тональность текста, введенного пользователем, с использованием API, например, https://api.deepai.org/api/text-generator. Отобразите результат анализа.
13. **Генератор случайных имен**:
    Создайте приложение, которое получает случайное имя с использованием API, например, https://uinames.com/api/. Отобразите имя.
14. **Поиск видео**:
    Разработайте приложение, которое позволяет пользователю искать видео по ключевому слову с использованием API, например, https://developers.google.com/youtube/v3/docs/search/list.
    Отобразите найденные видео.
15. **Анализ настроения изображения**:
    Создайте приложение, которое анализирует настроение изображения, загруженного пользователем, с использованием API, например, https://api.deepai.org/api/image-sentiment-analysis.
    Отобразите результат анализа.
16. **Генератор случайных карт**:
    Разработайте приложение, которое получает случайную карту с использованием API, например, https://deckofcardsapi.com/api/deck/new/draw/.Отобразите карту.
17. **Поиск фильмов по жанру**:
    Создайте приложение, которое позволяет пользователю искать фильмы по жанру с использованием API OMDB. Отобразите найденные фильмы.

18. **Анализ тональности аудио**:
    Разработайте приложение, которое анализирует тональность аудиофайла, загруженного пользователем, с использованием API, например, https://api.deepai.org/api/audio-sentiment-analysis.
    Отобразите результат анализа.
19. **Генератор случайных паролей**:
    Создайте приложение, которое генерирует случайный пароль с использованием API, например, https://passwordwolf.com/api. Отобразите пароль.
20. **Поиск книг по автору**:
    Разработайте приложение, которое позволяет пользователю искать книги по автору с использованием API Google Books. Отобразите найденные книги.
21. **Анализ настроения текста с использованием библиотеки dio**:
    Перепишите задание 3 с использованием библиотеки dio вместо http.
22. **Переводчик с использованием библиотеки dio**:
    Перепишите задание 4 с использованием библиотеки dio вместо http.
23. **Генератор случайных картинок с использованием библиотеки dio**:
    Перепишите задание 5 с использованием библиотеки dio вместо http.
24. **Поиск GIF-изображений с использованием библиотеки dio**:
    Перепишите задание 6 с использованием библиотеки dio вместо http.
25. **Прогноз погоды на неделю с использованием библиотеки dio**:
    Перепишите задание 7 с использованием библиотеки dio вместо http.

Дополнительные рекомендации:

1. Используйте комментарии для пояснения работы приложения.
2. Оформляйте код в соответствии с общепринятыми стандартами.
3. Тестируйте программу на различных наборах данных.
4. Представить результат работы в рукописном виде в рабочей тетради и в электронном формате с использованием Git-репозитория с предоставление QR-кода на репозиторий.

Критерии оценивания:

«Отлично» - выполнены все тренировочные задания и пятнадцать индивидуальных заданий.

«Хорошо» - выполнены все тренировочные задания и десять индивидуальных заданий.

«Удовлетворительно» - выполнены все тренировочные задания и пять индивидуальных заданий.