ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

Профессор				Татарникова Т. М.
должность, уч. степень, звание		подпись, дата		инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

МОДЕЛИРОВАНИЕ БАЗОВОЙ СЛУЧАЙНОЙ ВЕЛИЧИНЫ

Вариант 5

по курсу: Моделирование систем

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	4128				Воробьев В. А.
			подпись, дата		инициалы, фамилия

Санкт-Петербург 2024

СОДЕРЖАНИЕ

1 Введение

В рамках производственной практики был осуществлен проект под названием “Реализация Dart пакета для работы с Yandex GPT API”. Основной целью данного проекта являлось создание специализированного программного средства, упрощающего процесс интеграции и взаимодействия с Yandex GPT API средствами языка программирования Dart.

Современные системы искусственного интеллекта, такие как Yandex GPT, предоставляют широкие возможности для решения разнообразных задач, начиная от генерации естественного текста и заканчивая различными видами анализа данных. Для максимально эффективного использования этих возможностей необходимы инструменты, позволяющие разработчикам легко подключаться к API и использовать его функционал в своих проектах.

Процесс реализации Dart пакета включал в себя несколько ключевых этапов: исследование документации Yandex GPT API, разработку и тестирование различных функциональных модулей, а также написание сопроводительной документации для пользователей пакета. Подробное внимание уделялось обеспечению стабильности работы пакета, его производительности и удобству использования.

В ходе выполнения проекта задействовались глубокие знания в области разработки программного обеспечения, тестирования и документирования кода. Этот проект не только продемонстрировал возможность эффективной интеграции с Yandex GPT API, но и предоставил полезный инструмент для Dart-разработчиков, способных значительно сократить время и усилия на реализацию аналогичных задач.

2 Цель работы

Целью работы является разработка Dart пакета для упрощения взаимодействия с Yandex GPT API, обеспечивая удобный и эффективный доступ к его функционалу для создания и интеграции решений, основанных на искусственном интеллекте, в приложениях на языке Dart.

3 Постановка задачи

Требуется следующий функционал:

1. Интеграция с Yandex GPT API для аутентификации, генерации текста, асинхронных операций, эмбеддингов и токенизации.
2. Возможность создания клиента API с различными настройками HTTP, включая изменение токена.
3. Корректное добавление аутентификационных заголовков ко всем запросам, поддержка асинхронной обработки.
4. Поддержка отмены запросов с помощью CancelToken.
5. Логирование запросов и ответов, корректная обработка ошибок.
6. Реализация юнит-тестов с высоким покрытием для основных методов.
7. Написание подробной документации и примеров использования.
8. Настройка автоматизированной проверки, сборки и развертывания через CI/CD.
9. Реализация методов для работы с асинхронными операциями.
10. Обеспечение кросс-платформенной совместимости с различными платформами, поддерживаемыми Dart.

4 Выполнение задания

4.1 Методические сведения

API (Application Programming Interface) - набор правил и инструментов для создания программного обеспечения, который позволяет взаимодействовать между различными программными компонентами.

RESTful API - архитектурный стиль для сетевых сервисов, который использует HTTP запросы для операций создания, чтения, обновления и удаления (CRUD).

Dart - язык программирования, разработанный компанией Google, который используется для создания веб-приложений, мобильных приложений и серверных приложений.

Dio - популярная HTTP-клиент библиотека для Dart, обеспечивающая простое и эффективное создание HTTP-запросов.

Асинхронное программирование - модель программирования, в которой задачи выполняются параллельно с основной программой, не блокируя выполнение других операций.

Future - объект в Dart, представляющий результат асинхронной операции, который будет доступен в будущем [1].

JSON - формат обмена данными, основанный на лёгком текстовом формате для представления структурированных данных.

OAuth (Open Authorization) - протокол аутентификации, позволяющий предоставлять сторонним приложениям ограниченный доступ к защищённым ресурсам.

Токен аутентификации - уникальная строка, используемая для идентификации и авторизации пользователя или приложения при доступе к защищенным ресурсам API.

Покрытие кода тестами (Code Coverage) - метрика, показывающая, какую часть кода программы проверено с помощью тестов [2] [3].

CI/CD (Continuous Integration/Continuous Deployment) - набор практик и инструментов для автоматизации процесса интеграции кода, его проверки и развертывания на рабочем сервере [4].

HTTP-заголовки - метаданные, отправляемые совместно с HTTP-запросами и ответами, содержащие информацию об отправителе, получателе и контенте.

Логирование - процесс записи событий, происходящих в программе, для последующего анализа и диагностики.

Обработка ошибок - процесс управления ошибками, возникающими во время выполнения программы, и предоставления информации об этих ошибках пользователю или системе.

Кросс-платформенное развитие - процесс разработки программного обеспечения, которое может работать на нескольких вычислительных платформах.

Управление зависимостями - процесс управления библиотеками и пакетами, которые использует проект для обеспечения необходимого функционала.

Tokenize (Токенизация) - процесс разбивки текста на отдельные элементы (токены), такие как слова, числа или другие значимые единицы.

Embedding (Эмбеддинг) - метод преобразования текста или слов в числовые векторы в многомерном пространстве для машинного обучения и анализа.

Обработка естественного языка (NLP) - область искусственного интеллекта, занимающаяся взаимодействием между компьютерами и человеческими языками.

Асинхронная операция - операция, результат выполнения которой становится доступен не сразу, а по завершению подсчета или выполнения3.

CancelToken - объект, используемый для отмены асинхронных операций.

BaseOptions - класс, используемый в Dio для настройки базовых опций HTTP-клиента (таймауты, заголовки, базовый URL и т. д.) [5].

Yandex GPT - модель искусственного интеллекта для генерации текста, созданная компанией Яндекс, использующая технологию GPT (Generative Pre-trained Transformer).

HTTP-методы - базовые методы HTTP-протокола (GET, POST, PUT, DELETE и т.д.), используемые для выполнения CRUD операций над ресурсами сервера.

Интерсептор (Interceptor) - компонент, который перехватывает HTTP-запросы и ответы, предоставляя возможность изменения их до того, как данные будут отправлены или получены.

Деплоймент (Deployment) - процесс развертывания программного обеспечения на рабочем сервере для доступности конечным пользователям.

4.2 Разработка

Разработка велась в методологии TDD.

Структура папок представлена на рисунках 4.1 и 4.2.

Рисунок 4.1 - Структура проекта

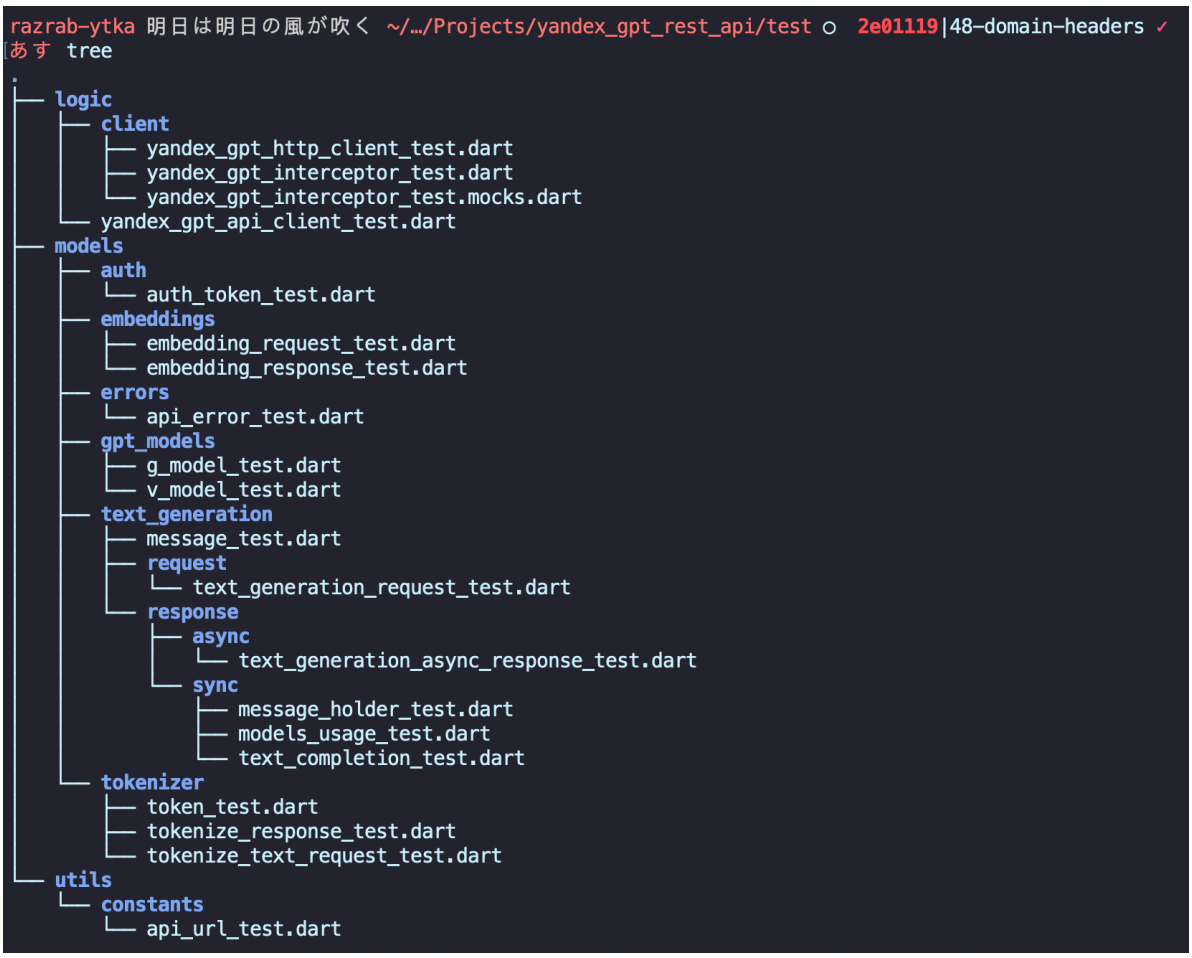

Рисунок 4.2 - Структура тестов

4.2.1 Проработка моделей

Был создан абстрактный класс GptModel, представляющий основу для всех моделей GPT, используемый для задания URI каждой модели. Конструкторы класса позволяют задавать URI напрямую через GptModel.raw или формировать его с добавлением версии через GptModel.version, что упрощает управление версиями моделей. Переопределенный метод toString предоставляет удобный способ получения строкового представления URI модели.

Для совместимости с различными моделями генерации текста Яндекс, был создан класс GModel, который наследует GptModel и включает несколько конструкторов для разных типов моделей. Конструкторы GModel формируют специфические URI для моделей Yandex GPT, Yandex GPT Light, обобщающих моделей и моделей с пользовательским обучением, используя заданные константы для формирования URI. Это позволяет гибко и централизованно управлять URI компонентами через изменения в одном месте.

```
1  abstract class GptModel {
```

```
  final String uri;

  const GptModel.raw({required this.uri});

  const GptModel.version({required String uri, String?
      version})
      : uri = '$uri/${version ?? 'latest'}';

  @override
  String toString() => uri;
}
```

```
import 'package:yandex_gpt_rest_api/src/models/gpt_models/
    gpt_model.dart';
import 'package:yandex_gpt_rest_api/src/utils/constants/
    models_uri.dart';

/// Generation models.
///
/// Check [Yandex docs](https://cloud.yandex.ru/en/docs/
    yandexgpt/concepts/models#yandexgpt-generation) for more
    info.
class GModel extends GptModel {
  const GModel.raw(String uri) : super.raw(uri: uri);

  const GModel.yandexGpt(String folderId, {super.version})
      : super.version(uri: gptPrefix + folderId +
          yandexGptPostfix);

  const GModel.yandexGptLight(String folderId, {super.version
      })
      : super.version(uri: gptPrefix + folderId +
          yandexGptLitePostfix);

  const GModel.summary(String folderId, {super.version})
      : super.version(uri: gptPrefix + folderId +
          summaryPostfix);

  const GModel.fineTuned(String modelId) : this.raw(dsPrefix
      + modelId);
}
```

4.2.2 Проработка ответов

Класс `TextGenerationRequest` предназначен для создания запросов на генерацию текста с использованием Yandex GPT API. Он содержит три ключевых параметра: model (представлен через GModel для указания URI модели), completionOptions (позволяет настроить параметры генерации, такие как максимальное число токенов и разнообразие текста) и messages (список сообщений, задающих контекст генерации). Конструктор класса позволяет задавать эти параметры, а для удобства сериализации и десериализации предусмотрены методы fromJson и toJson.

Метод toString был переопределён для удобного вывода информации о запросе в виде строки. Имплементация методов fromJson и toJson обеспечивает преобразование объекта в JSON и обратно, что упрощает передачу данных при взаимодействии с API. В результате, класс TextGenerationRequest предоставляет удобный и структурированный способ создания запросов для генерации текста, а также интеграцию с форматами JSON для простоты взаимодействия с Yandex GPT API.

Код всех моделей приведен в Листинге.

4.2.3 Реализация HTTP клиента

Класс YandexGptHeaderInterceptor отвечает за добавление аутентификационных данных в заголовки запросов к Yandex GPT API. В конструкторе класса принимаются параметры catalog и token, которые инициализируют соответствующие поля. Метод onRequest добавляет заголовки аутентификации и идентификатор каталога к запросам, направленным на хост, указанный в ApiUrl.host. Дополнительно, метод changeToken позволяет обновлять токен аутентификации в течение жизни объекта.

Класс YandexGptHttpClient предоставляет фасад для работы с HTTP-клиентом Dio, упрощая отправку запросов и обработку ответов. Основные методы класса - post и get - выполняют соответствующие HTTP-запросы, принимая URL и данные тела запроса. Оба метода используют приватный метод _fetch для выполнения запросов и обработки ошибок. Метод _fetch пытается выполнить запрос, декодирует JSON-ответ и возвращает его в виде словаря. В случае ошибки, если ответ содержит ApiError, исключение перехватывается и выбрасывается объект ApiError.

Вместе классы YandexGptHeaderInterceptor и YandexGptHttpClient обеспечивают надежную и удобную работу с HTTP-запросами к Yandex GPT API, упрощая интеграцию и обработку аутентификации и ошибок.

```dart
import 'package:dio/dio.dart';
import 'package:yandex_gpt_rest_api/src/models/auth/auth_token.dart';
import 'package:yandex_gpt_rest_api/src/utils/constants/api_url.dart';
import 'package:yandex_gpt_rest_api/src/utils/constants/headers.dart';

/// Interceptor for adding authentication data to the request.
class YandexGptHeaderInterceptor extends Interceptor {
  final String _catalog;
  AuthToken _token;

  YandexGptHeaderInterceptor({
    required String catalog,
    required AuthToken token,
  }) : _catalog = catalog,
       _token = token;

  @override
  void onRequest(RequestOptions options, RequestInterceptorHandler handler) {
    if (options.uri.host == ApiUrl.host) {
      options.headers.addAll({
        authHeaderName: _token.toString(),
        catalogIdHeaderName: _catalog,
      });
    }
    handler.next(options);
  }

  void changeToken(AuthToken token) {
    _token = token;
  }
}
```

```dart
import 'dart:async';
```

```dart
import 'dart:convert';

import 'package:dio/dio.dart';
import 'package:yandex_gpt_rest_api/src/models/errors/
    api_error.dart';

/// Facade for working with `Dio`.
class YandexGptHttpClient {
  final Dio _dio;

  const YandexGptHttpClient(Dio dio) : _dio = dio;

  Future<Map<String, dynamic>> post(
    String url, {
    Map<String, dynamic>? body,
    CancelToken? cancelToken,
  }) =>
      _fetch(
        _dio.post<String>(
          url,
          data: jsonEncode(body),
          cancelToken: cancelToken,
        ),
      );

  Future<Map<String, dynamic>> get(
    String url, {
    Map<String, dynamic>? body,
    CancelToken? cancelToken,
  }) =>
      _fetch(
        _dio.get<String>(
          url,
          data: jsonEncode(body),
          cancelToken: cancelToken,
        ),
      );

  /// A wrapper for handling [ApiError] responses.
  Future<Map<String, dynamic>> _fetch(Future<Response<String
    >> request) async {
```

```
41      try {
42        final response = await request;
43        final jsonBody = jsonDecode(response.data!) as Map<
              String, dynamic>;
44        return jsonBody;
45      } on DioException catch (e) {
46        // Check if response contains [ApiError].
47        final body = jsonDecode(e.response?.data as String? ??
              "{}");
48        final apiError =
49            ApiError.tryParseJson(body is Map<String, dynamic>
                ? body : {});
50        if (apiError == null) rethrow;
51        throw apiError;
52      }
53    }
54  }
```

4.2.4 Реализация API

Создан клиент YandexGptApi, который упрощает выполнение запросов к API и управление ими. Для этого внедрен класс YandexGptHeaderInterceptor, добавляющий аутентификационные заголовки в каждый запрос.

Основные функции включают генерацию текста синхронно и асинхронно, получение статуса асинхронных операций, получение текстовых эмбеддингов и токенизацию текста. Взаимодействие с API осуществляется через HTTP-клиент Dio, с обработкой запросов и ответов.

Для управления запросами и их отмены используется CancelToken. Ошибки API корректно обрабатываются и декодируются. Реализация методов была оптимизирована для надежной и быстрой работы с минимизацией задержек.

```
1  import 'package:dio/dio.dart';
2  import 'package:yandex_gpt_rest_api/src/logic/client/
       yandex_gpt_header_interceptor.dart';
3  import 'package:yandex_gpt_rest_api/src/logic/client/
       yandex_gpt_http_client.dart';
4  import 'package:yandex_gpt_rest_api/src/models/models.dart';
5  import 'package:yandex_gpt_rest_api/src/utils/constants/
       api_url.dart';
6
```

```
7  /// Client for YandexGPT RESTful API.
8  final class YandexGptApi {
9    final YandexGptHttpClient _client;
10   final YandexGptHeaderInterceptor _headerInterceptor;
11
12   /// Create default API Client.
13   ///
14   /// If [catalog] is not specified, [catalog] = [AuthToken]
       directory.
15   YandexGptApi({
16     required AuthToken token,
17     String? catalog,
18   }) : this.withDio(dio: Dio(), token: token, catalog:
       catalog);
19
20   /// Create API Client using Dio with [options].
21   ///
22   /// If [catalog] is not specified, [catalog] = [AuthToken]
       directory.
23   YandexGptApi.withOptions({
24     required BaseOptions options,
25     required AuthToken token,
26     String? catalog,
27   }) : this.withDio(
28           dio: Dio()..options = options,
29           token: token,
30           catalog: catalog,
31         );
32
33   /// Create API Client using [dio]. Adds authentication
       headers to all [dio] requests.
34   ///
35   /// ONE [dio] should only be used by ONE [YandexGptApi].
36   ///
37   /// If [catalog] is not specified, [catalog] = [AuthToken]
       directory.
38   YandexGptApi.withDio({
39     required Dio dio,
40     required AuthToken token,
41     String? catalog,
42   })  : _client = YandexGptHttpClient(dio),
```

```
43          _headerInterceptor = YandexGptHeaderInterceptor(
44            catalog: catalog ?? "",
45            token: token,
46          ) {
47      dio.interceptors.add(_headerInterceptor);
48    }
49
50    /// Change authentication to [token].
51    ///
52    /// Current requests will not be stopped.
53    void changeToken(AuthToken token) {
54      _headerInterceptor.changeToken(token);
55    }
56
57    Future<TextCompletion> generateText(
58      TextGenerationRequest request, {
59      CancelToken? cancelToken,
60    }) async {
61      final res = await _client.post(
62        ApiUrl.textGeneration,
63        body: request.toJson(),
64        cancelToken: cancelToken,
65      );
66      return TextCompletion.fromJson(
67        res['result'] as Map<String, dynamic>,
68      );
69    }
70
71    Future<TextGenerationAsyncResponse> generateAsyncText(
72      TextGenerationRequest request, {
73      CancelToken? cancelToken,
74    }) async {
75      final res = await _client.post(
76        ApiUrl.textGenerationAsync,
77        body: request.toJson(),
78        cancelToken: cancelToken,
79      );
80      return TextGenerationAsyncResponse.fromJson(res);
81    }
82
83    /// Get text generation operation status for [operationId].
```

```
84    ///
85    /// See also:
86    /// - [generateAsyncText]
87    Future<TextGenerationAsyncResponse>
          getOperationTextGenerate(
88      String operationId, {
89      CancelToken? cancelToken,
90    }) async {
91      final res = await _client.get(
92        ApiUrl.operation(operationId),
93        cancelToken: cancelToken,
94      );
95      return TextGenerationAsyncResponse.fromJson(res);
96    }
97
98    Future<EmbeddingResponse> getTextEmbedding(
99      EmbeddingRequest request, {
100     CancelToken? cancelToken,
101   }) async {
102     final res = await _client.post(
103       ApiUrl.textEmbedding,
104       body: request.toJson(),
105       cancelToken: cancelToken,
106     );
107     return EmbeddingResponse.fromJson(res);
108   }
109
110   Future<TokenizeResponse> tokenizeCompletion(
111     TextGenerationRequest request, {
112     CancelToken? cancelToken,
113   }) async {
114     final res = await _client.post(
115       ApiUrl.tokenizeCompletion,
116       body: request.toJson(),
117       cancelToken: cancelToken,
118     );
119     return TokenizeResponse.fromJson(res);
120   }
121
122   Future<TokenizeResponse> tokenizeText(
123     TokenizeTextRequest request, {
```

```
124       CancelToken? cancelToken,
125     }) async {
126       final res = await _client.post(
127         ApiUrl.tokenizeText,
128         body: request.toJson(),
129         cancelToken: cancelToken,
130       );
131       return TokenizeResponse.fromJson(res);
132     }
133   }
```

4.2.5 Подключение CI/CD

Во время производственной практики была настроена автоматизация публикации пакетов на pub.dev и проведения непрерывной интеграции (CI) для тестирования кода.

Для автоматизации процесса публикации был создан workflow, который запускается при пуше тегов, соответствующих шаблону версии. Это позволяет автоматически публиковать новые версии пакета на pub.dev. Workflow использует сценарий из репозитория dart-lang/setup-dart и обеспечивает аутентификацию через токен с разрешениями на запись.

Для обеспечения качества кода и стабильности была настроена система CI, запускающаяся при создании pull request и пуше в ветку main. Эта система выполняет клонирование репозитория, установку Flutter, установку зависимостей проекта и запуск тестов с измерением покрытия кода. В завершение система загружает отчеты о покрытии кода в Codecov, используя токен для аутентификации.

Настройка этих процессов позволила автоматизировать публикацию пакетов и поддерживать высокое качество кода.

```
1  name: Publish to pub.dev
2
3  on:
4    push:
5      tags:
6      - 'v[0-9]+.[0-9]+.[0-9]'
7
8  jobs:
9    publish:
10     permissions:
```

```
11        id-token: write
12      uses: dart-lang/setup-dart/.github/workflows/publish.
          yml@v1
13
14  name: Test CI
15
16  on:
17    pull_request:
18      branches: [ main ]
19    push:
20      branches: [ main ]
21
22  jobs:
23    coverage:
24      runs-on: ubuntu-latest
25
26      steps:
27        - uses: actions/checkout@v3
28        - uses: subosito/flutter-action@v2
29          with:
30            channel: 'stable'
31        - run: flutter pub get
32
33        - name: Run test coverage
34          run: flutter test --coverage
35
36        - name: Upload coverage reports to Codecov
37          uses: codecov/codecov-action@v3
38          if: '!cancelled()'
39          with:
40            directory: coverage
41          env:
42            CODECOV_TOKEN: ${{ secrets.CODECOV_TOKEN }}
```

4.2.6 Написание тестов

Для всех классов были написаны исчерпывающие юнит тесты. Охват тестами изображен на риснуке 4.3.

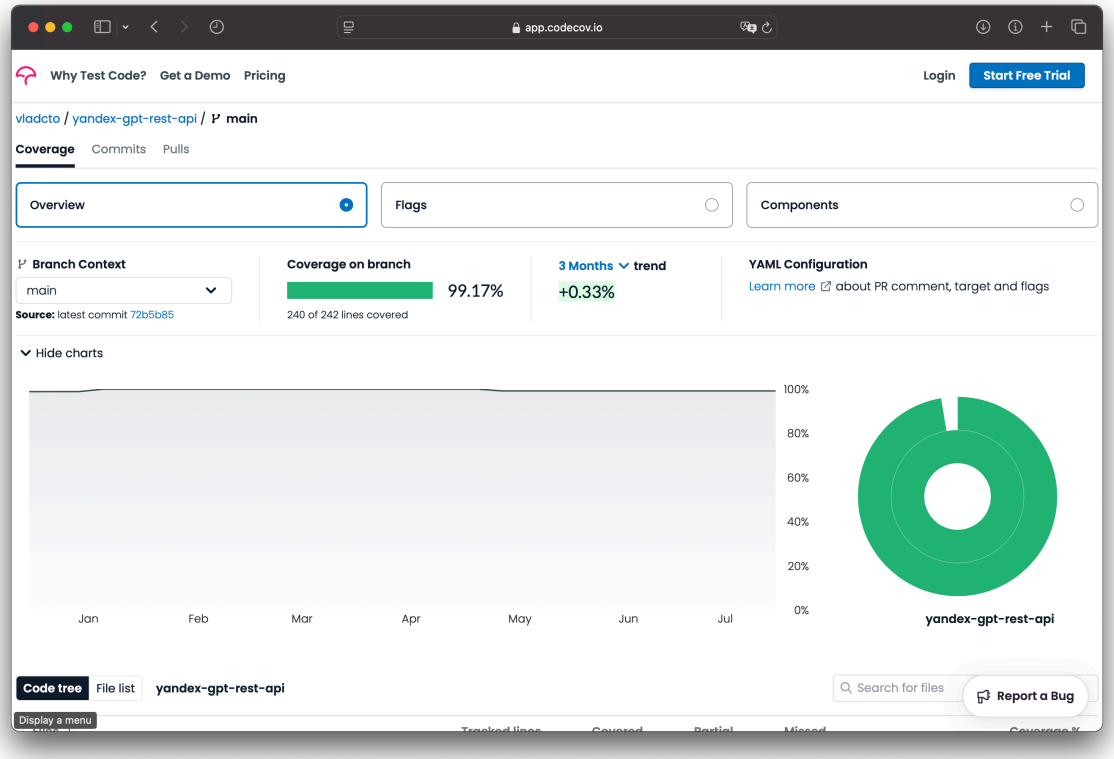

Рисунок 4.3 - Code-coverage

5 Заключение

В ходе производственной практики была успешно реализована библиотека для взаимодействия с Yandex GPT RESTful API, что позволило упростить выполнение запросов и управление ими. Также была настроена автоматизация процессов публикации пакетов на pub.dev и проведения непрерывной интеграции (CI) для тестирования кода.

Были достигнуты основные цели проекта: - Создан клиент YandexGptApi с функционалом для генерации текста, получения эмбеддингов и токенизации текста. - Внедрена автоматическая система добавления аутентификационных заголовков в запросы с использованием YandexGptHeaderInterceptor. - Настроены workflows для автоматической публикации пакетов и проверки качества кода через CI, что обеспечило стабильность и надежность проекта.

Итоговый пакет был выложен на pub.dev (URI - `https://pub.dev/packages/yandex_gpt_rest_api`), а код доступен в репозитории GitHub (URI - `https://github.com/vladcto/yandex-gpt-rest-api`).

Таким образом, выполненная работа обеспечивает удобное взаимодействие с Yandex GPT API и поддержание высокого качества кода с минимальными затратами на ручное тестирование и управление релизами.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Hall C. Dart Apprentice: Learning Dart as a Beginner / C. Hall, Boston: Razeware LLC, 2022.

2. Meier J. Automate the Boring Stuff with GitHub Actions: Boost your GitHub workflows / J. Meier, San Francisco: Independent, 2020.

3. Patters F. Practical Flutter: Improve your Mobile Development with Google's Latest SDK / F. Patters, New York: Apress, 2021.

4. Ruiter C. Beginning App Development with Dart: Create Cross-Platform Apps with Google's Flutter / C. Ruiter, New York: Apress, 2019.

5. Smith T. Essential Workflow Automation with GitHub Actions: Managing Code Deployments and DevOps Pipelines / T. Smith, Birmingham: Packt Publishing, 2020.

ПРИЛОЖЕНИЕ

```dart
yandex_gpt_http_client.dart
import 'dart:async';
import 'dart:convert';

import 'package:dio/dio.dart';
import 'package:yandex_gpt_rest_api/src/models/errors/
    api_error.dart';

/// Facade for working with `Dio`.
class YandexGptHttpClient {
  final Dio _dio;

  const YandexGptHttpClient(Dio dio) : _dio = dio;

  Future<Map<String, dynamic>> post(
    String url, {
    Map<String, dynamic>? body,
    CancelToken? cancelToken,
  }) =>
      _fetch(
        _dio.post<String>(
          url,
          data: jsonEncode(body),
          cancelToken: cancelToken,
        ),
      );

  Future<Map<String, dynamic>> get(
    String url, {
    Map<String, dynamic>? body,
    CancelToken? cancelToken,
  }) =>
      _fetch(
        _dio.get<String>(
          url,
          data: jsonEncode(body),
          cancelToken: cancelToken,
        ),
      );

```

```
40    /// A wrapper for handling [ApiError] responses.
41    Future<Map<String, dynamic>> _fetch(Future<Response<String
        >> request) async {
42      try {
43        final response = await request;
44        final jsonBody = jsonDecode(response.data!) as Map<
            String, dynamic>;
45        return jsonBody;
46      } on DioException catch (e) {
47        // Check if response contains [ApiError].
48        final body = jsonDecode(e.response?.data as String? ??
            "{}");
49        final apiError =
50            ApiError.tryParseJson(body is Map<String, dynamic>
                ? body : {});
51        if (apiError == null) rethrow;
52        throw apiError;
53      }
54    }
55  }
56
57  yandex_gpt_header_interceptor.dart
58  import 'package:dio/dio.dart';
59  import 'package:yandex_gpt_rest_api/src/models/auth/
      auth_token.dart';
60  import 'package:yandex_gpt_rest_api/src/utils/constants/
      api_url.dart';
61  import 'package:yandex_gpt_rest_api/src/utils/constants/
      headers.dart';
62
63  /// Interceptor for adding authentication data to the request
      .
64  class YandexGptHeaderInterceptor extends Interceptor {
65    final String _catalog;
66    AuthToken _token;
67
68    YandexGptHeaderInterceptor({
69      required String catalog,
70      required AuthToken token,
71    }) : _catalog = catalog,
72         _token = token;
```

```
73
74    @override
75    void onRequest(RequestOptions options,
          RequestInterceptorHandler handler) {
76      if (options.uri.host == ApiUrl.host) {
77        options.headers.addAll({
78          authHeaderName: _token.toString(),
79          catalogIdHeaderName: _catalog,
80        });
81      }
82      handler.next(options);
83    }
84
85    void changeToken(AuthToken token) {
86      _token = token;
87    }
88  }
```