**Документирование и планирование работы по проектам**

Предлагаю выполнить следующие изменения по проекту - сформировать набор файлов /внести изменения в файлы:

1. **Структурированные заголовки в каждом файле** с описанием:

* Назначения модуля
* Ключевых компонентов
* Зависимостей
* Требований к данным
* Выходных данных

1. **CHANGELOG.md** для отслеживания изменений:

* Версионирование по семантической версии
* Хронология изменений
* Категории изменений (добавлено/изменено/исправлено)

1. **ARCHITECTURE.md** с подробным описанием системы:

* Общий обзор
* Описание компонентов
* Диаграмма взаимодействия
* Потоки данных
* Требования
* Масштабируемость
* Безопасность

1. **README.md** с основной информацией о проекте
2. **DEVELOPMENT\_PLAN.md** план разработки

Такая структура документации позволяет:

1. Быстро понять назначение каждого модуля
2. Отслеживать изменения в проекте
3. Понимать архитектуру системы
4. Легко находить нужную информацию

DEVELOPMENT\_PLAN.md

1. ==================
2. # План разработки
3. ## Текущие задачи
4. 1. [Задача 1]
5. - Подзадача 1.1
6. - Подзадача 1.2
7. - Зависимости: [список]
8. 2. [Задача 2]
9. - Подзадача 2.1
10. - Подзадача 2.2
11. - Зависимости: [список]
12. ## Завершенные задачи
13. 1. [Задача]
14. - Реализация: [ссылка на код]
15. - Результаты: [описание]
16. - Дата завершения: [дата]
17. ## Будущие задачи
18. 1. [Задача]
19. - Приоритет: [высокий/средний/низкий]
20. - Зависимости: [список]

Первый вопрос у меня про **CHANGELOG.md** для отслеживания изменений.

* 1. Надеюсь, что в этом файле Ты будешь дополнять информацию по тем заданиям на корректировку документации/кода, которые от меня поступают, однако есть еще и итоговые данные по результату работы измененного кода, которые в большей мере видны только мне (часть выдачи в терминал, конечно, я могу Тебе отправлять для анализа и добавления в **CHANGELOG.md** , но это только часть информации, ведь есть еще и диаграммы и зависящие от подаваемых исходных данных иные результаты). Так вот, наверное, мне также целесообразно вносить дополнения в этот файл, чтобы Ты видел и те результаты, которые видны только мне – как считаешь?

Далее еще ряд общих моментов в разработке, которых мне хотелось бы придерживаться (вместе с Тобой), в частности:

1. деление проекта на модули
2. тестирование предложенного кода
3. логгирование в коде (идеально было бы включать логгирование в настройках, когда это необходимо и даже, может быть, сделать два варианта логгирования – Log\_on=0: без вывода логов, Log\_on=1: только вывод общих данных верхнего уровня, и Log\_on=2: вывод и остальных данных
4. декомпозиция задач
5. подготовка документации по коду
6. создание файла с планом действий: plan, а также хорошо бы приучить себя сразу для необходимого в больших проектах вести также и файл зависимостей, который представляет собой реализованные уже задачи с указанием где находится соответствующий по реализации код: dependences

Также есть общие принципы разработки, которых хотелось бы придерживаться в разработке кода:

* 1. Предпочитайте простой, читаемый и понятный код. Избегайте преждевременной оптимизации. Избегайте преждевременных абстракций.
  2. Работайте инженером-программистом с 20-летним опытом работы в различных SaaS-стартапах. цените ясность, простоту и удобочитаемость.

Где мне целесообразнее прописывать всю подобную информацию, чтобы Ты перед новыми этапами работы смог удобнее и быстрее эту информацию прочитать и действовать с учетом как начальных, так и граничных условий?

С учетом Твоего ответа сейчас подготовлю план по данной задаче

1. **По поводу CHANGELOG.md**:

стоит вносить дополнения в CHANGELOG.md, особенно касающиеся результатов работы. Предлагаю следующую структуру для записей в CHANGELOG.md:

## [1.1.0] - 2024-03-20

### Результаты

- Улучшение точности кластеризации на 15%

- Сокращение времени обработки данных на 30%

- Визуализация новых паттернов в пользовательском поведении

- [Добавьте скриншоты или ссылки на графики]

### Технические изменения

- Оптимизация алгоритма кластеризации

- Добавление новых метрик оценки

- Улучшение визуализации

### Пользовательские наблюдения

- [Ваши наблюдения по результатам работы]

- [Особенности поведения системы]

- [Проблемные места]

1. Для организации всей информации по разработке предлагаю создать следующие файлы:

PROJECT\_GUIDELINES.md

====================

# Руководство по разработке

## Принципы разработки

1. Предпочитайте простой, читаемый и понятный код

2. Избегайте преждевременной оптимизации

3. Избегайте преждевременных абстракций

4. Работайте как опытный инженер-программист

5. Цените ясность, простоту и удобочитаемость

## Структура проекта

1. Модульность

- Каждый модуль должен иметь четкую ответственность

- Минимизация зависимостей между модулями

- Четкие интерфейсы между модулями

2. Тестирование

- Unit-тесты для каждого модуля

- Интеграционные тесты

- Тесты производительности

3. Логирование

- Log\_on=0: без вывода логов

- Log\_on=1: только общие данные верхнего уровня

- Log\_on=2: полный вывод данных

4. Декомпозиция задач

- Разбиение на подзадачи

- Приоритизация

- Оценка сложности

5. Документация

- Docstrings для всех функций

- README для каждого модуля

- Архитектурная документация

DEPENDENCIES.md

==============

# Зависимости и связи

## Реализованные задачи

1. [Название задачи]

- Файл: [путь к файлу]

- Функции: [список функций]

- Зависимости: [список зависимостей]

2. [Название задачи]

- Файл: [путь к файлу]

- Функции: [список функций]

- Зависимости: [список зависимостей]

## Зависимости между модулями

```mermaid

graph TD

A[Модуль 1] --> B[Модуль 2]

A --> C[Модуль 3]

B --> D[Модуль 4]

```

```

1. **\*\*Для логирования\*\* предлагаю создать отдельный модуль `logger.py`:**

```python

"""

Logging Module

=============

Provides configurable logging functionality for the project.

"""

import logging

from enum import Enum

class LogLevel(Enum):

NONE = 0

BASIC = 1

FULL = 2

class ProjectLogger:

def \_\_init\_\_(self, log\_level=LogLevel.BASIC):

self.log\_level = log\_level

self.logger = logging.getLogger('project')

self.\_setup\_logger()

def \_setup\_logger(self):

if self.log\_level != LogLevel.NONE:

handler = logging.StreamHandler()

formatter = logging.Formatter(

'%(asctime)s - %(name)s - %(levelname)s - %(message)s'

)

handler.setFormatter(formatter)

self.logger.addHandler(handler)

self.logger.setLevel(logging.INFO)

def log(self, message, level=LogLevel.BASIC):

if self.log\_level.value >= level.value:

self.logger.info(message)

def set\_level(self, level):

self.log\_level = level

```

12.6

В файле данных «processed\_data.csv» следующие столбцы:

source,group,object\_name,address,object\_rating,review\_count,review\_text,review\_rating,user\_name,review\_date,answer\_text,latitude,longitude,district

просьба подготовить скрипт для загрузки указанного файла из текущей директории, перемешивании строк файла и выборке из полученных данных части данных с записью в файл \*.csv

в терминал просьба вывести текст:

«ввести долю данных от исходного файла которая останется в новом файле»

Значение по умолчанию – 20 (доля будет 1/20=0.05)

**processed\_data\_1\_20.csv** так будет называться новый файл по умолчанию

если введем – 10 (десятая часть от исходного файла), то новый файл будет называться **processed\_data\_1\_10.csv**