~ *Проект 2, модуль №3* ~

Предисловие

В рамках этого проекта вам предстоит разработать консольное приложение для решения конкретной задачи. Этот проект имитирует реальную рабочую ситуацию, где вам необходимо разработать программное обеспечение в соответствии с техническим заданием.

Цель проекта – применить все знания и навыки, полученные в течении курса, для создания функционального и качественного программного продукта. Вам предстоит самостоятельно спроектировать архитектуру приложения, реализовать необходимую функциональность, протестировать и отладить ваше решение.

У проекта есть базовая часть, а потом 2 ветки развития:

- A-side (темная сторона)

  Эта ветка создана для тех, кто хочет получить проект, который не выходит за рамки пройденного на лекциях. Максимальная оценка при таком выборе ограничивается 8 баллами.

- B-side (светлая сторона)

  Эта ветка для тех, кто хочет поработать с чем-то новым. Ограничения по оценке на данной ветке нету.

Можно выбрать ТОЛЬКО одну ветку. Выбранную ветку **обязательно** укажите в комментариях при сдаче проекта. Все, что будет сделано в рамках ветки, которая не была выбрана не будет оцениваться.

ВАЖНО: Проект должен быть выполнен в строгом соответствии с техническим заданием и общими требованиями к учебным работам. Внимательно изучите все разделы этого документа перед началом работы. Если у вас возникают вопросы, задавайте их в форме Q&A по проекту.

Общие слова

Что делаем?

Проект предполагает самостоятельную разработку ~~консольного~~ приложения в соответствии с техническим заданием (см. раздел «Основная задача» ниже). Вам потребуется:

- Внимательно изучить техническое задание и общие требования к работе.
- Спроектировать архитектуру приложения, включая классы, методы и структуры данных.
- Разработать программный код на языке C#, реализующий всю необходимую функциональность.
- Провести тестирование и отладку приложения для обеспечения его корректной работы.
- Подготовить отчет о выполненной работе (если требуется, см. раздел «Дополнительные задачи»).
- Сдать в SmartLMS заархивированную папку с решением (Solution) Visual Studio, содержащую проект(ы) с вашим решением и другие необходимые файлы.

Когда сдаем?

Определяется датами, назначенными в SmartLMS.

Что сдаём?

На проверку предоставляется заархивированная папка с решением Visual Studio, в которую включен(ы) проект(ы), через который(е) реализована программа. В названии решения и архива обязательно указать фамилию автора.

Ограничения

В основной и дополнительной задачах **требуется** (да, это блокирующие критерии):

- весь программный код написан на языке программирования C# с учётом использования .net 8.0;
- представленная к проверке библиотека классов решает все поставленные задачи и успешно компилируется.
- в приложении реализован цикл повторения решения, позволяющий повторить работу на других данных без завершения сеанса работы;
- запрещено использовать:
  - NuGet-пакеты, использование которых явно не указано в техническом задании или не согласовано с преподавателем;
  - Прочий публично доступный код / библиотеки, не относящиеся к стандартным библиотекам .NET, без явного указания авторства и согласования с преподавателем;

Общие требования к работе

Ваше приложение должно:

- Соответствовать заданию (см. раздел «Основная задача» ниже).
- Быть работоспособным: Запускаться и выполнять все заявленные в техническом задании функции без критических ошибок.
- Обеспечивать корректный ввод данных в соответствии с требованиями технического задания (если это необходимо). Ввод может быть из консоли, из файла, или генерироваться внутри программы (например, для демонстрации).
- Обеспечивать корректный вывод результатов работы в соответствии с требованиями технического задания (если это необходимо). Вывод может быть в консоль, в файл, или визуализироваться в консоли. Формат вывода должен быть четким и понятным.
- Обрабатывать возможные ошибки ввода данных и исключительные ситуации: Программа не должна «падать» при некорректном вводе пользователя или при возникновении непредвиденных ситуаций. Сообщения об ошибках должны быть информативными и позволять пользователю понять причину проблемы.
- Иметь чистый и хорошо структурированный код, соответствующий принципам объектно-ориентированного программирования (если применимо).
- Содержать подробные комментарии в коде, объясняющие логику работы каждого важного участка кода.
- Иметь понятную структуру проекта (разделение на папки, файлы, классы, методы).
- Быть оформлено в соответствии с общепринятыми стандартами оформления кода на C# (стиль, отступы, именование переменных и т.д.).

Также обратите внимание на примечания в конце файла.

~ *Индивидуальные варианты* ~

~ *Вариант №1* ~

Базовая часть

Разработать консольное приложение «Анализатор логов». Приложение должно:

1. Читать данные из текстового файла логов.

   Каждая строка представляет собой отдельный лог и имеет следующий вид:

   ```
   [Дата и время] [Уровень важности] [Сообщение]
   ```

   Пример:

   ```
   [2024-07-26 10:30:15] [INFO] Запуск приложения.
   [2024-07-26 10:30:20] [WARNING] Не удалось подключиться к базе данных.
   [2024-07-26 10:31:00] [ERROR] Критическая ошибка в модуле обработки данных.
   [2024-07-26 10:35:45] [INFO] Приложение завершило работу.
   ```

- Путь к файлу логов должен запрашиваться у пользователя.
- Приложение должно проверять корректность введенного пути и обрабатывать возможные ошибки (файл не найден, ошибка чтения файла).

2. Предоставлять пользователю возможность фильтровать записи логов по следующим критериям:

- Уровень важности (INFO, WARNING, ERROR, DEBUG, ...).
- Диапазон дат и времени.
- Поиск по ключевому слову в сообщении (регистронезависимый поиск).

Стоит отметить, что перечня уровней важности не существует. В уровне важности может быть все, что угодно.

3. Выводить отфильтрованные записи на консоль в следующем формате:

```
----------------------------------------
[2024-07-26 10:30:20] [WARNING]
Не удалось подключиться к базе данных.
----------------------------------------
[2024-07-26 10:31:00] [ERROR]
Критическая ошибка в модуле обработки данных.
----------------------------------------
```

4. Сделать экспорт отфильтрованных данных в формате, описанном в 1 пункте

A-side

Основная часть

1. Сортировка

   - Дате и времени (по возрастанию или убыванию).
   - Уровню важности (по возрастанию или убыванию).
   - Длине сообщения (по возрастанию или убыванию).

2. Статистика

   Добавить возможность вывода статистики по логам:
   - Общее количество записей.
   - Количество записей по каждому уровню важности.
   - Самая ранняя и самая поздняя записи.
   - Средняя длина сообщения.

3. Сохранение (дополнение)

   Реализовать сохранение отфильтрованных и отсортированных записей и/или результатов статистики в новый текстовый файл (по выбору пользователя). Путь к файлу и формат сохранения (для статистики) запрашивать у пользователя.

Дополнительная часть

1. Множественные файлы

   Реализовать возможность обработки нескольких файлов логов одновременно. Приложение должно:
   - Запрашивать у пользователя список файлов логов (пути к файлам).
   - Объединять данные из всех указанных файлов.
   - Предоставлять возможность фильтрации и сортировки объединенных данных.
   - Добавить в вывод в консоль и в файл (при сохранении) информации об имени файла, из которого взята каждая запись.

2. Настраиваемый формат ввода/вывода

   Добавить возможность настройки формата ввода и вывода записей в консоль и в файл через конфигурационный файл и через интерактивное меню. Настройки должны включать:
   - Порядок следования полей (дата, уровень, сообщение).
   - Разделитель между полями.
   - Формат даты и времени.

B-side

Основная часть

1. Визуализация

   Реализовать визуализацию данных логов с помощью библиотеки Spectre.Console:
   - **Таблица:** Отображение отфильтрованных записей в табличном виде (с возможностью прокрутки, если записей много).
   - **Breakdown Chart:** Построение диаграммы разбивки по уровням важности за выбранный период. График и все значения должны быть подписаны.
   - **Календарь:** Отображение календаря с выделением дней, в которые были записи логов (интенсивность цвета дня должна соответствовать количеству записей).
   - **Интерактивное меню:** Предоставление интерактивного меню для выбора параметров фильтрации, визуализации и сохранения.

2. Асинхронность

   Использовать асинхронные операции для чтения файлов логов, чтобы избежать блокировки основного потока при работе с большими файлами.

Дополнительная часть

1. Построение графиков

   Используйте ScottPlot для построения графика:

   - количества записей логов по времени
   - количества ошибок по времени

   Должна быть возможность сохранения графиков в виде изображений.

2. Добавить возможность вычисления и отображения расширенной статистики

   - **Частота ошибок:** Вычисление частоты появления ошибок (ERROR) за определенный период (час, день, неделя).
   - **Поиск аномалий:** Реализовать простейший алгоритм поиска аномалий (например, этот).
   - **Топ сообщений:** Вывод списка наиболее часто встречающихся сообщений (с указанием количества повторений).

3. REST API сервер

   Реализовать **базовый** REST API сервер для управления логами. Сервер должен предоставлять следующие возможности:

   - **Добавление логов (Endpoint: `/logs`, Method: POST):**
     - Принимает данные лога в теле запроса в формате JSON.
     - Сервер должен сохранять полученные логи.
     - Возвращает код ответа HTTP 201 Created в случае успешного добавления.
   - **Просмотр логов (Endpoint: `/logs`, Method: GET):**
     - **Без параметров:** Возвращает **все** сохраненные логи в формате JSON (массив объектов, где каждый объект - лог).
     - **С параметрами диапазона (Query Parameters):**
       - `from`: Начальная дата и время для диапазона.
       - `to`: Конечная дата и время для диапазона.
       - Возвращает логи, отфильтрованные по указанному диапазону дат и времени, в формате JSON. Если параметры `from` или `to` не указаны, диапазон не применяется (возвращаются все логи).
   - **Получение простейшей статистики (Endpoint: `/logs/statistics`, Method: GET):**
     - **С параметрами диапазона (Query Parameters):**
       - `from`: Начальная дата и время для диапазона.
       - `to`: Конечная дата и время для диапазона.
     - Возвращает JSON объект со следующей статистикой **для указанного диапазона**:
       - `totalMessages`: Общее количество лог-сообщений в диапазоне.
       - `logLevelsCount`: Объект, где ключи - уровни важности (например, «INFO», «WARNING», «ERROR»), а значения - количество логов каждого уровня в диапазоне.
       - Если параметры `from` или `to` не указаны, статистика считается по **всем** сохраненным логам.

~ *Вариант №2* ~

- Базовая часть
- A-side
- B-side

Базовая часть

Разработать консольное приложение «Менеджер задач». Приложение должно:

1. Хранение задач.

Формат файла не регламентирован. Например, каждая строка может описывать отдельную задачу и иметь следующий вид:

```
[ID задачи] [Статус] [Приоритет] [Описание задачи]
```

Список статусов:
- TODO
- IN_PROGRESS
- DONE

Список приоритетов:
- Низкий
- Средний
- Высокий

Пример файла:

```
[1] [TODO] [Высокий] Написать отчет по проекту
[2] [IN_PROGRESS] [Средний] Провести встречу с командой
[3] [DONE] [Низкий] Ответить на письма
```

- Приложение должно читать задачи из файла при запуске.
- Путь к файлу задач должен запрашиваться у пользователя.
- Приложение должно проверять корректность пути и обрабатывать ошибки.

2. Предоставлять пользователю возможность:

   - **Просмотра всех задач.**
   - **Добавления новой задачи.** Запрашивать у пользователя описание и приоритет, устанавливать статус TODO, уникальный ID генерировать автоматически.
   - **Изменения статуса задачи.** Пользователь выбирает ID задачи и новый статус.
   - **Удаления задачи.** Пользователь выбирает ID задачи.

3. Выводить задачи на консоль в формате:

```
ID: 1
Статус: TODO
Приоритет: Высокий
Описание: Написать отчет по проекту
--------------------
ID: 2
Статус: IN_PROGRESS
Приоритет: Средний
Описание: Провести встречу с командой
--------------------
```

4. Сохранять изменения задач обратно в файл при завершении работы приложения.

5. Экспорт/Импорт

   - Реализовать экспорт задач в CSV и JSON для обмена данными с другими приложениями или для резервного копирования.
   - Реализовать импорт задач из этих форматов.
   - При добавлении новых полей в структуру, эти поля должны также добавляться в экспортируемые файлы

[К варианту]

A-side

Основная часть

1. Фильтрация и сортировка задач:

- Фильтрация по статусу и/или приоритету.
- Сортировка (по возрастанию/убыванию):
  - ID;
  - Статусу;
  - Приоритету.

2. Поиск задач:

   Поиск по ключевым словам в описании задачи (регистронезависимый поиск).

3. Статистика:
   - Вывод статистики по количеству задач в каждом статусе.
   - Вывод общего количества задач.

Дополнительная часть

1. Категории задач:

- Добавить возможность присваивать задачам пользовательскую категорию (например, «Работа», «Личное», «Учеба»). Должна быть возможность создавать собственные категории.
- Реализовать фильтрацию и статистику по категориям.

2. Напоминания:

- Реализовать возможность установки напоминаний для задач (дата и время)
- Приложение должно уведомлять пользователя о приближающихся напоминаниях (например, за 24 часа)

[К варианту]

B-side

Основная задача

1. Визуализация с помощью Spectre.Console:
   - Таблица: Отображение задач в табличном виде с возможностью прокрутки.
   - Интерактивная фильтрация и сортировка в таблице: Интерактивное меню в таблице Spectre.Console, позволяющее пользователю динамически фильтровать задачи по различным критериям
2. Работа с датами:
   - Добавить в формат задачи поля
     - «Дата создания» (автоматически при создании);
     - «Дата изменения» (автоматически при любом действии с задачей)
   - Добавить возможность сортировки и фильтрации задач по этим датам.
3. Зависимости задач
   - Реализовать возможность указывать зависимости между задачами. Например, задача «Написать отчет» может зависеть от задачи «Собрать данные».

     Задача может зависеть от нескольких других задач. Необходимо проверять не появляется ли циклическая зависимость при изменении связей.
   - При изменении статуса задачи проверять, не блокирует ли это какие-либо другие задачи.
4. Сроки выполнения и просроченные задачи:
   - Добавить поле «Срок выполнения» (Deadline) для задач (дата и время).
   - Приложение должно отслеживать сроки выполнения задач.
   - Визуально выделять просроченные задачи в списке.
   - Предупреждение о приближающихся сроках выполнения
5. Асинхронность и фоновые операции:
   - Использовать асинхронные операции для чтения и записи файла задач, чтобы избежать блокировки основного потока, особенно при работе с большим количеством задач или сетевыми файловыми хранилищами
   - Реализовать фоновые операции для периодической проверки сроков выполнения и отправки уведомлений

Дополнительная задача

1. Управление Проектами

   Расширить «Менеджер задач» до уровня «Мини-Менеджера Проектов».
   - Ввести понятие «Проект», который может содержать несколько задач.
   - Возможность создавать, удалять, переименовывать проекты.
   - Задачи должны быть привязаны к проектам.
   - Отображение задач по проектам.
   - Статистика по проектам:
     - количество задач в проекте
     - количество выполненных задач в проекте
     - процент выполнения проекта
2. Уведомления в Telegram

   Нужно отправлять уведомления о приближающихся задачах в Telegram.
3. Отслеживание прогресса задач
   - Добавить возможность указывать процент выполнения каждой задачи;
   - Отображать прогресс задач визуально в виде progress-bar’a

~ *Вариант №3* ~

- Базовая часть
- A-side
- B-side

Базовая часть

Разработать консольное приложение «Кулинарная книга». Приложение должно:

1. Хранить рецепты.

   Формат файла не регламентирован. Каждый рецепт может быть представлен отдельным блоком:

   ```
   Название рецепта: [Название]
   Категория: [Категория рецепта (например, Завтрак, Обед, Ужин, Десерт)]
   Ингредиенты:
   - [Ингредиент 1]
   - [Ингредиент 2]
   ...
   Инструкция:
   - [Шаг 1]
   - [Шаг 2]
   ...
   ```

   Пример файла:

   ```
   Название рецепта: Омлет
   Категория: Завтрак
   Ингредиенты:
   - Яйца - 2 шт.
   - Молоко - 50 мл
   - Соль, перец по вкусу
   - Души грешников
   Инструкция:
   - Взбить яйца с молоком, солью и перцем.
   - Вылить на разогретую сковороду.
   - Готовить до готовности.

   Название рецепта: Борщ
   Категория: Обед
   Ингредиенты:
   - Вода - 3 л
   - Картофель - 3 шт.
   ...
   Инструкция:
   - ...
   ```

   Путь к файлу рецептов должен запрашиваться у пользователя.

   Приложение должно проверять корректность введенного пути и обрабатывать ошибки.

2. Предоставлять пользователю возможность:
   - Просмотра всех рецептов (вывод списка названий).
   - Просмотра детальной информации о рецепте (по выбору названия из списка).
   - Добавления нового рецепта. Запрашивать у пользователя название, категорию, ингредиенты (списком, пока не будет введен пустой ингредиент), инструкцию (списком, пока не будет введен пустой шаг).
   - Удаления рецепта (по выбору названия из списка).
3. Сохранять изменения рецептов обратно в файл при завершении работы приложения.

A-side

Основная часть

1. **Поиск рецептов:**
   - По названию рецепта (регистронезависимый поиск).
   - По ингредиентам (поиск по совпадению хотя бы одного ингредиента из запроса).
   - По категории.
   - Возможность комбинировать критерии поиска.
2. **Сортировка рецептов:**
   - По названию (в алфавитном порядке).
   - По категории.
3. **Статистика:**
   - Вывод количества рецептов по каждой категории.
   - Вывод общего количества рецептов.

Дополнительная часть

1. **Избранное:**
   - Добавить возможность отмечать рецепты как «избранные».
   - При выводе рецептов списком, по особенному выделять избранные рецепты.
   - Реализовать отдельный просмотр только избранных рецептов.
   - Сохранять информацию об избранных рецептах (например, в отдельном поле в файле или в отдельном файле).
2. **Расширенный формат ингредиентов:**
   - Добавить возможность указывать количество и единицу измерения для каждого ингредиента (например, «Яйца - 2 шт.», «Молоко - 50 мл»).
   - При выводе рецепта, ингредиенты отображать в новом формате.

B-side

Основная задача

1. **Визуализация с помощью Spectre.Console:** Таблица: **Отображение списка рецептов в виде таблицы (с возможностью прокрутки). Интерактивное меню:** Для выбора действий (просмотр, добавление, поиск, сортировка и т.д.) и навигации по рецептам.
2. **Работа с изображениями: Добавить возможность прикреплять изображения к рецептам (путь к файлу изображения).** При просмотре детальной информации о рецепте, отображать изображение при помощи Spectre.Console.
3. **Импорт/Экспорт:**
   - Реализовать импорт рецептов из JSON, CSV.
   - Реализовать экспорт рецептов в форматы JSON и CSV.
4. **Случайный рецепт дня:**
   - Реализовать функцию, которая выбирает случайный рецепт из кулинарной книги и выводит его как «Рецепт дня».

Дополнительная задача

1. **Сетевая кулинарная книга: Реализовать возможность загрузки и сохранения рецептов в облачное хранилище (например, используя Yandex Disk или аналогичный сервис) или в репозиторий.** Возможность синхронизации кулинарной книги между несколькими устройствами.

   Важно:
   - Сделайте установку API ключа (или user+password) через переменные среды (.env)
   - Если выбираете нестандартный сервис, то предоставьте необходимые данные для тестирования

     К стандартным относится Яндекс.Диск и GitHub.

     Google Drive не относится к стандартным, так как выпуск API ключа затруднен.
2. **Генерация списка покупок:** Реализовать функцию автоматического создания списка покупок на основе выбранного рецепта (или нескольких рецептов).

   Приложение должно уметь извлекать ингредиенты из рецепта и формировать список. Желательно, чтобы список можно было сохранить в текстовый файл. Нужно добавить логику объединения одинаковых ингредиентов из разных рецептов (например, если в двух рецептах используется «лук», в списке покупок должен быть один пункт «лук» с общим количеством, если количество указано в рецепте).
3. **ИИ**

   Даже у тостера сегодня должен быть ИИ, так чем наша программа хуже?

   Возьмите БЕСПЛАТНЫЙ токен любой существующей нейросети (например, GigaChat) и с использованием этого токена генерируйте рецепты.

   Нужно, чтобы пользователь написал, что именно ему сейчас хочется, а в ответ получить предложение на добавление рецептов в книгу. У пользователя должна быть возможность отклонить добавление рецепта.

~ *Вариант №4* ~

- Базовая часть
- A-side
- B-side

[К варианту]

Базовая часть

Разработать консольное приложение «Менеджер личных финансов». Приложение должно:

1. **Хранить данные о транзакциях.**

   Формат файла не регламентирован. Каждая строка может представлять собой отдельную транзакцию и иметь следующий вид:

   ```
   [Дата (ГГГГ-ММ-ДД)] [Сумма] [Категория] [Описание]
   ```

   Список категорий:
   - Продукты
   - Транспорт
   - Развлечения
   - Коммунальные платежи
   - Зарплата
   - Другое (по умолчанию)

   Пример файла:

   ```
   2024-10-26 500 Продукты Покупка продуктов в супермаркете
   2024-10-25 -150 Транспорт Поездка на автобусе
   2024-10-25 30000 Зарплата Зарплата за октябрь
   ```

   - Путь к файлу транзакций должен запрашиваться у пользователя.
   - Приложение должно проверять корректность введенного пути и обрабатывать ошибки.

2. **Предоставлять пользователю возможность:**
   - **Просмотра всех транзакций** (в виде таблицы: Дата, Сумма, Категория, Описание).
   - **Добавления новой транзакции.** Запрашивать у пользователя дату, сумму (доход или расход, знак +/-), категорию (выбор из списка), описание.
   - **Удаления транзакции** (по номеру строки в списке транзакций).

3. **Выводить информацию о транзакциях на консоль в формате таблицы:**

   ```
   № | Дата       | Сумма    | Категория         | Описание
   ------------------------------------------------------------------
   1 | 2024-10-26 | 500     | Продукты          | Покупка продуктов ...
   2 | 2024-10-25 | -150    | Транспорт         | Поездка на автобусе
   3 | 2024-10-25 | 30000   | Зарплата          | Зарплата за октябрь
   ...
   ------------------------------------------------------------------
   Итого доход:  [Сумма доходов]
   Итого расход: [Сумма расходов]
   Баланс:       [Баланс]
   ```

   Вывод в формате таблицы обязателен, даже если дальше требуется использование других библиотек для их вывода. Просто добавьте отдельную кнопку в меню.

4. **Сохранять изменения транзакций обратно в файл** при завершении работы приложения.

[К варианту]

[К варианту]

A-side

Основная часть

1. **Фильтрация транзакций:**
   - По дате (диапазон дат).
   - По категории.
   - Возможность комбинировать фильтры.
2. **Сортировка транзакций:**
   - По дате (по возрастанию/убыванию).
   - По сумме (по возрастанию/убыванию).
   - По категории.
3. **Статистика и отчеты:**
   - Вывод общей суммы доходов и расходов за выбранный период (месяц, год, произвольный диапазон).
   - Вывод статистики расходов по категориям за выбранный период (в виде таблицы или диаграммы в консоли, если возможно, или просто текстовый отчет).

Дополнительная часть

1. **Расширенные категории:**
   - Добавить возможность пользователю добавлять и удалять категории транзакций.
   - Сохранять список пользовательских категорий.
2. **Валюты:**
   - Добавить возможность указывать валюту для каждой транзакции (например, рубли, доллары, евро).
   - Выводить общую статистику в выбранной пользователем валюте (по умолчанию - рубли).

[К варианту]

B-side

Основная задача

1. **Визуализация с помощью Spectre.Console:**
   - **Таблица:** Отображение транзакций в виде интерактивной таблицы с возможностью фильтрации и сортировки прямо в таблице.
   - **Диаграммы:** Построение диаграмм распределения расходов по категориям за месяц (например, Bar Chart).
2. **Бюджетирование:**
   - Добавить возможность устанавливать месячный бюджет по категориям.
   - При просмотре статистики за месяц, показывать сравнение фактических расходов с бюджетом по каждой категории, а также общий процент выполнения бюджета.
   - Визуально выделять категории, где бюджет превышен.
3. **Прогнозирование:**
   - Реализовать простой прогноз расходов на следующий месяц на основе данных за предыдущий период (например, средние расходы по каждой категории за последние 3 месяца).
4. **Анализ трендов:**

   Реализовать построение графиков при помощи ScottPlot. Нужно строить график расходов/доходов за месяц и гистограмму расходов по категориям. Придумайте и реализуйте еще 2 графика на ваш вкус.

Дополнительная задача

1. **Интеграция с Telegram-ботом:** Создать Telegram-бота, который должен уметь делать то же самое, что и консольное приложение.

~ *Вариант №5* ~

- Базовая часть
- A-side
- B-side

Базовая часть

Разработать консольное приложение «Менеджер библиотеки». Приложение должно:

1. **Хранить данные о книгах.**

   Формат файла не регламентирован. Например, каждая строка может представлять собой отдельную книгу и иметь следующий вид:

   ```
   [Название] [Автор] [Жанр] [Год издания] [ISBN]
   ```

   Список жанров:
   - Фантастика
   - Детектив
   - Роман
   - История
   - Научная литература
   - ... (и другие, на ваш выбор)

   Пример файла:

   ```
   [Мастер и Маргарита] [Михаил Булгаков] [Роман] [1967] [xxx]
   [1984] [Джордж Оруэлл] [Фантастика] [1949] [xxx]
   [Убийство в Восточном экспрессе] [Агата Кристи] [Детектив] [1934] [xxx]
   ```

   - Путь к файлу библиотеки должен запрашиваться у пользователя.
   - Приложение должно проверять корректность введенного пути и обрабатывать возможные ошибки.
   - Поле ISBN должно валидироваться (и 10, и 13)

     При редактировании - проверять контрольную цифру.
2. **Предоставлять пользователю возможность:**
   - **Просмотра всех книг.**
   - **Добавления новой книги.**

     Запрашивать у пользователя название, автора, жанр, год издания и добавлять запись в файл.
   - **Редактирования информации о книге.**

     Пользователь выбирает книгу и может изменить любое поле (название, автор, жанр, год).
   - **Удаления книги.**

     Пользователь выбирает книгу и удаляет запись из файла.
3. **Выводить информацию о книгах на консоль в формате:**

   ```
   Название: Мастер и Маргарита
   Автор: Михаил Булгаков
   Жанр: Роман
   Год издания: 1967
   --------------------
   Название: 1984
   Автор: Джордж Оруэлл
   Жанр: Фантастика
   Год издания: 1949
   --------------------
   ...
   ```

4. **Сохранять изменения библиотеки обратно в файл при завершении работы приложения.**

A-side

Основная часть

1. **Фильтрация книг:**
   - По жанру.
   - По автору.
   - По году издания (диапазон).
   - Возможность комбинировать фильтры.
2. **Сортировка книг (в прямом/обратном порядке):**
   - По названию.
   - По автору.
   - По году издания.
3. **Статистика:**
   - Общее количество книг в библиотеке.
   - Количество книг по каждому жанру.
   - Самая старая и самая новая книга.

Дополнительная часть

1. **Читатели:**
   - Добавить возможность учета читателей. Хранить информацию о читателях:
     - ID читателя
     - ФИО
     - Контактные данные
     - ...
   - Связать книги с читателями (кто взял какую книгу).
   - Реализовать функционал выдачи и возврата книг.
2. **Поиск книг по ISBN:**

   Статья на Wikipedia

   - Реализовать поиск книг по ISBN.
     - по стране
     - по коду издательства
     - уникальному номеру издания

   Создайте небольшой перечень заранее вбитых стран и издательств, чтобы можно было просто выбрать «рускоязычные страны», «Хорватия» или «Издательство ЭКСМО». В каждом перечне не должно быть менее 5 элементов.

B-side

Основная задача

1. **Визуализация с помощью Spectre.Console:**
   - **Таблица:** Отображение списка книг в виде интерактивной таблицы (с возможностью фильтрации и сортировки прямо в таблице).
   - **Календарь:** Отображение календаря с выделением годов издания книг (интенсивность цвета года должна соответствовать количеству книг, изданных в этот год).
2. **Рекомендации:**
   - Добавить к каждой книге поле «Оценка»
   - Реализовать простейшую систему рекомендаций книг на основе оценок пользователя.
3. **Импорт/Экспорт:**
   - Реализовать импорт/экспорт библиотеки в форматы JSON и CSV для обмена данными с другими библиотечными системами или для резервного копирования.

Дополнительная задача

1. **Интеграция с openlibrary.org:**
   - Добавить возможность поиска информации о книгах в openlibrary по ISBN и автоматического добавления найденной информации в свою библиотеку. Считать информацию с openlibrary приоритетной. Если найдены нестыковки - исправлять их. Например, установить корректное название издателя или категорию книги.
   - Добавить поддержку обложек для книг, которые будут скачиваться с openlibrary. Приложение должно выводить изображения в терминал (через Spectre.Console) и путь до изображения.

~ *Вариант №6* ~

- Базовая часть
- A-side
- B-side

Базовая часть

Разработать консольное приложение «Менеджер контактов». Приложение должно:

1. **Хранить данные о контактах.**

   Формат файла не регламентирован. Например, каждая строка может представлять собой запись о контакте и иметь следующий вид:

```
[ID контакта] [Имя] [Фамилия] [Телефон] [Email]
```

Пример:

```
[1] [Иван] [Иванов] [+79123456789] [ivan@example.com]
[2] [Петр] [Петров] [+79234567890] [petr@example.com]
[3] [Анна] [Сидорова] [+79345678901] [anna@example.com]
```

- Приложение должно читать контакты из файла при запуске.
- Путь к файлу контактов должен запрашиваться у пользователя.
- Приложение должно проверять корректность пути и обрабатывать ошибки.

2. **Предоставлять пользователю возможность:**

- **Просмотра всех контактов.**
- **Добавления нового контакта.** Запрашивать у пользователя имя, фамилию, телефон, email, устанавливать уникальный ID автоматически.
- **Редактирования контакта.** Пользователь выбирает ID контакта и может изменить любое поле (имя, фамилию, телефон, email).
- **Удаления контакта.** Пользователь выбирает ID контакта.

3. **Выводить контакты на консоль в формате:**

```
ID: 1
Имя: Иван
Фамилия: Иванов
Телефон: +79123456789
Email: ivan@example.com

ID: 2
Имя: Петр
Фамилия: Петров
Телефон: +79234567890
Email: petr@example.com
```

4. **Сохранять изменения контактов обратно в файл при завершении работы приложения.**

5. **Поиск контактов по имени, фамилии или телефону (частичное совпадение).**

[К варианту]

A-side

Основная часть

1. **Фильтрация и сортировка контактов:**

- Фильтрация по имени, фамилии (начинается с...).
- Сортировка (по возрастанию/убыванию):
  - ID;
  - Имя;
  - Фамилия.
- Возможность комбинировать фильтры и сортировки.

2. **Группы контактов (категории):**

- Добавить возможность присваивать контактам категории (например, «Работа», «Семья», «Друзья»).
- Реализовать фильтрацию и статистику по категориям (сколько контактов в каждой категории).

3. **Импорт/Экспорт:**

- Реализовать экспорт контактов в CSV и JSON для обмена данными с другими приложениями или для резервного копирования.
- Реализовать импорт контактов из этих форматов.

Дополнительная задача

1. Групповые операции и метки времени:

- **Групповое добавление контактов:**
  - Реализовать возможность добавления нескольких контактов за одну операцию. Пользователь может ввести данные нескольких контактов, разделяя их определенным образом (например, каждый контакт с новой строки, а поля контакта разделены запятыми). Приложение должно корректно разобрать введенные данные и добавить все контакты.
- **Групповое удаление контактов:**
  - Предоставить возможность удаления нескольких контактов одновременно. Пользователь может выбрать несколько контактов (например, указав их ID через запятую или выбрав из списка) и удалить их одной командой.
- **Метки времени создания и изменения:**
  - Добавить к каждому контакту поля «Дата создания» и «Дата последнего изменения».
  - Эти поля должны автоматически заполняться при создании и редактировании контакта, соответственно.
  - Предоставить возможность фильтрации и сортировки контактов по этим меткам времени.

2. Экспорт/Импорт выбранных контактов:
   - Реализовать функцию экспорта только выбранных контактов в CSV или JSON. Пользователь должен иметь возможность выбрать контакты (например, по ID, по категории, по результатам фильтрации) и экспортировать только их.
   - Реализовать функцию импорта, которая корректно обрабатывает ситуацию, когда в импортируемом файле уже есть контакты с ID, которые существуют в текущей базе. Предоставить пользователю выбор:
     - Пропустить дубликаты.
     - Обновить существующие контакты данными из файла.
     - Создать новые контакты с другими ID (сгенерировать автоматически).

[К варианту]

[К варианту]

B-side

Основная задача

1. **Визуализация с помощью Spectre.Console:**

- **Таблица:** Отображение контактов в табличном виде с возможностью прокрутки и интерактивной фильтрации/сортировки в таблице.
- **Breakdown Chart:** Построить диаграмму распределения контактов по категориям.

2. **Работа с датами:**

- Добавить поле «Дата рождения» для контактов.

  Стоит предусмотреть возможность хранения дня рождения без года.

- Добавить возможность фильтрации и сортировки контактов по дате рождения.
- Выводить список ближайших дней рождения (на текущую неделю).

3. **Синхронизация с облаком (простое решение):**

- Реализовать простую синхронизацию файла контактов с облачным хранилищем (например, Yandex Disk или Google Drive) для доступа к контактам с разных устройств (без сложной реализации конфликтов, просто последняя версия файла перезаписывает предыдущую).

Дополнительная задача

1. **Добавить рассылки по базе**

- Добавить возможность отправки Email выбранному контакту/контактам прямо из приложения.

2. **Распознавание телефонных номеров и email-адресов в текстовом файле:**

- Если формат входного файла менее структурированный, реализовать простейшее распознавание имени, фамилии, телефона и email-адреса с использованием регулярных выражений (без ИИ).

3. **Расширенные взаимосвязи контактов:**

- **Реализовать возможность устанавливать взаимосвязи между контактами.** Например, пользователь может указать, что контакт «Иван Иванов» «работает с» контактом «Петр Петров» или «является родственником» контакта «Анна Сидорова». Предусмотреть несколько типов взаимосвязей (например, «работает с», «друг», «родственник», «знакомый» и т.д.).
- **Визуализировать взаимосвязи контактов.** Отобразить взаимосвязи в виде простого графа. Можно использовать SkiaSharp.
- **Предоставить возможность фильтрации и поиска контактов с учетом взаимосвязей.** Например, найти всех контактов, «работающих с» определенным контактом, или «друзей друзей» определенного контакта (ограничив глубину поиска, чтобы не перегружать систему).

[К варианту]

~ *Вариант №7* ~

- Базовая часть
- A-side
- B-side

Базовая часть

Разработать консольное приложение «Менеджер личных заметок». Приложение должно:

1. **Хранить данные о заметках.**

   Формат файла не регламентирован. Например, каждая строка может представлять собой отдельную заметку и иметь следующий вид:

   ```
   [ID заметки] [Заголовок] [Текст заметки] [Дата создания (ГГГГ-ММ-ДД)]
   ```

   Пример файла:

   ```
   [1] [Идеи для проекта] [Рассмотреть возможность использования Spectre.Console для UI.]
   [2024-11-10]
   [2] [Список покупок] [Молоко, хлеб, яйца, сыр.] [2024-11-12]
   [3] [Рецепт пирога] [Найти рецепт яблочного пирога в интернете.] [2024-11-14]
   ```

   - Приложение должно читать заметки из файла при запуске.
   - Путь к файлу заметок должен запрашиваться у пользователя.
   - Приложение должно проверять корректность пути и обрабатывать ошибки.

2. **Предоставлять пользователю возможность:**
   - **Просмотра всех заметок** (в виде списка: ID, Заголовок, Дата создания).
   - **Просмотра детальной информации о заметке** (по выбору ID из списка): Заголовок, Текст заметки, Дата создания.
   - **Добавления новой заметки.** Запрашивать у пользователя заголовок и текст заметки, дата создания устанавливается автоматически.
   - **Редактирования заметки.** Пользователь выбирает ID заметки и может изменить заголовок и текст заметки.
   - **Удаления заметки.** Пользователь выбирает ID заметки.

3. **Выводить информацию о заметках на консоль в формате:**

   ```
   Список заметок:
   ------------------------------------------------------
   ID: 1
   Заголовок: Идеи для проекта
   Дата создания: 2024-11-10

   ID: 2
   Заголовок: Список покупок
   Дата создания: 2024-11-12
   ...
   ```

4. **Сохранять изменения заметок обратно в файл при завершении работы приложения.**

A-side

Основная часть

1. **Фильтрация заметок:**
   - По дате создания (диапазон дат).
   - По ключевым словам в заголовке или тексте заметки (регистронезависимый поиск).
   - Возможность комбинировать фильтры.
2. **Сортировка заметок:**
   - По дате создания (по возрастнию/убыванию).
   - По заголовку (в алфавитном порядке).
3. **Категории заметок (теги):**
   - Добавить возможность присваивать заметкам категории (теги, например, «Работа», «Личное», «Идеи»). Хранить теги в заметках, например, через запятую в отдельном поле.
   - Реализовать фильтрацию заметок по категориям.
   - Статистика: Вывод количества заметок по каждой категории.

Дополнительная часть

1. **Архивация заметок.**

- Реализовать функциональность «архивации» заметок.
- Добавить команду для «архивации» заметки по ее ID.
- Архивированные заметки не должны отображаться в общем списке заметок по умолчанию.
- Реализовать отдельную команду для просмотра списка архивированных заметок.
- Предусмотреть возможность «разархивировать» заметку, возвращая ее в общий список.

2. **Редактор многострочных заметок.**

- **Реализовать возможность ввода и редактирования многострочного текста заметок.**
- При добавлении и редактировании заметки:
  - **Перейти от однострочного ввода текста заметки к многострочному.** Пользователь должен иметь возможность вводить текст заметки, нажимая Enter для переноса на новую строку. Должна быть возможность возврата к предыдущим строкам.
  - **Обеспечить понятный способ завершения ввода многострочного текста.**
  - При отображении детальной информации о заметке, **многострочный текст должен корректно отображаться в консоли, сохраняя форматирование переносов строк.**

B-side

Основная задача

1. **Визуализация с помощью Spectre.Console:**
   - **Таблица:** Отображение заметок в виде интерактивной таблицы с возможностью фильтрации и сортировки прямо в таблице.
   - **Календарь:** Отображение календаря с выделением дней, в которые были созданы заметки (интенсивность цвета дня должна соответствовать количеству заметок).
2. **Форматирование текста заметок:**
   - Реализовать возможность простого форматирования текста заметок (например, использование Markdown-подобной разметки для выделения **курсивом**, **жирным**, создания списков).
   - Отображать отформатированный текст заметок при просмотре детальной информации (можно использовать Spectre.Console для форматирования).
3. **Синхронизация с облаком (простое решение):**
   - Реализовать простую синхронизацию файла заметок с облачным хранилищем (например, Yandex Disk или Google Drive) для доступа к заметкам с разных устройств (без сложной реализации конфликтов, просто последняя версия файла перезаписывает предыдущую).

Дополнительная задача

1. **Интеграция с онлайн-сервисом синонимов (тезаурусом) для обогащения текста заметок.**

- Выбрать **бесплатный и доступный API** сервиса синонимов (например, WordNet API, Big Huge Thesaurus API).
- Добавить команду или опцию в контекстном меню для «поиска синонимов» для выделенного слова или всего текста заметки.
- При выборе этой опции, приложение должно:
  - Отправить запрос к выбранному API, передав слово или текст заметки.
  - Получить список синонимов от сервиса.
  - Отобразить полученные синонимы пользователю в консоли (можно использовать Spectre.Console для более красивого отображения).
  - Предоставить пользователю возможность **заменить** исходное слово/текст на выбранный синоним (или группу синонимов).

2. **Редактор многострочных заметок.**

- **Реализовать возможность ввода и редактирования многострочного текста заметок.**
- При добавлении и редактировании заметки:
  - **Перейти от однострочного ввода текста заметки к многострочному.** Пользователь должен иметь возможность вводить текст заметки, нажимая Enter для переноса на новую строку. Должна быть возможность возврата к предыдущим строкам.
  - **Обеспечить понятный способ завершения ввода многострочного текста.**
  - При отображении детальной информации о заметке, **многострочный текст должен корректно отображаться в консоли, сохраняя форматирование переносов строк.**

~ *Вариант №8* ~

- Базовая часть
- A-side
- B-side

Базовая часть

Разработать консольное приложение «Каталог фильмов». Приложение должно:

1. **Хранить данные о фильмах.**

   Формат файла не регламентирован. Например, каждая строка может представлять собой отдельный фильм и иметь следующий вид:

   ```
   [Название фильма] [Жанр] [Год выпуска] [Рейтинг (от 1 до 10)]
   ```

   Список жанров: (можно предложить несколько вариантов, например, «Фантастика», «Комедия», «Драма», «Боевик», «Триллер» и т.д.)

   Пример файла:

   ```
   [Интерстеллар] [Фантастика] [2014] [9]
   [Назад в будущее] [Фантастика, Комедия] [1985] [8]
   [Криминальное чтиво] [Триллер, Драма] [1994] [9]
   ```

   - Приложение должно читать данные о фильмах из файла при запуске.
   - Путь к файлу каталога должен запрашиваться у пользователя.
   - Приложение должно проверять корректность пути и обрабатывать ошибки.

2. **Предоставлять пользователю возможность:**
   - **Просмотра всех фильмов** (в виде списка: Название фильма, Жанр, Год выпуска, Рейтинг).
   - **Добавления нового фильма.** Запрашивать у пользователя название, жанр, год выпуска, рейтинг.
   - **Редактирования рейтинга фильма.** Пользователь выбирает фильм и может изменить его рейтинг.
   - **Удаления фильма.** Пользователь выбирает фильм для удаления.

3. **Выводить информацию о фильмах на консоль в формате:**

   ```
   Каталог фильмов:
   -------------------------------------------------
   Название: Интерстеллар
   Жанр: Фантастика
   Год выпуска: 2014
   Рейтинг: 9
   -------------------------------------------------
   Название: Назад в будущее
   Жанр: Фантастика, Комедия
   Год выпуска: 1985
   Рейтинг: 8
   -------------------------------------------------
   ...
   ```

4. **Сохранять изменения каталога обратно в файл при завершении работы приложения.**

A-side

Основная часть

1. **Фильтрация и сортировка фильмов:**
   - Фильтрация по жанру (можно выбирать несколько жанров).
   - Фильтрация по диапазону годов выпуска.
   - Фильтрация по диапазону рейтинга.
   - Сортировка (по возрастанию/убыванию):
     - По названию фильма.
     - По году выпуска.
     - По рейтингу.
   - Возможность комбинировать фильтры и сортировки.
2. **Статистика:**
   - Вывод статистики по количеству фильмов каждого жанра.
   - Вывод среднего рейтинга для каждого жанра.
3. **Поиск фильмов:**
   - Поиск фильма по названию (частичное совпадение).

Дополнительная часть

1. **Расширенный поиск и фильтрация по актерам и режиссерам:**
   - Добавить поле для хранения списка актеров и режиссера(ов) для каждого фильма.
   - **Поиск по актерам и режиссерам:** Реализовать поиск фильмов не только по названию и жанру, но и по имени актера или режиссера (частичное совпадение, регистронезависимый поиск). Пользователь должен иметь возможность искать фильмы с участием конкретного актера или режиссера.
   - **Фильтрация по актерам и режиссерам:** Добавить возможность фильтрации фильмов по актерам и режиссерам. Например, пользователь может выбрать актера и увидеть список всех фильмов в каталоге, где он снимался. Можно реализовать множественный выбор актеров/режиссеров для фильтрации («показать фильмы с участием актеров X И Y»).
   - **Комбинированный поиск и фильтрация:** Сохранить возможность комбинировать все виды поиска и фильтрации (по названию, жанру, году, рейтингу, актерам, режиссерам) для создания сложных запросов. Например, «найти все фильмы жанра ’Фантастика’, выпущенные после 2000 года, с рейтингом выше 8, и с участием актера ’N’ «.
2. **Персонализированные списки «Избранное» и «Хочу посмотреть» с тегами:**
   - **Списки:** Добавить возможность создавать несколько пользовательских списков фильмов, например, «Избранное», «Хочу посмотреть», «Рекомендации друзей» и т.д. Пользователь должен иметь возможность добавлять фильмы в разные списки и просматривать фильмы в каждом списке отдельно.
   - **Теги для списков:** Для списков «Избранное» и «Хочу посмотреть» добавить возможность присваивать фильмам внутри списка пользовательские теги или метки (например, «Для семейного просмотра», «На вечер», «К просмотру с другом»). Это позволит пользователю дополнительно организовывать фильмы в своих списках. Реализовать фильтрацию фильмов внутри списка по этим тегам.

B-side

Основная задача

1. **Визуализация с помощью Spectre.Console:**
   - **Таблица:** Отображение фильмов в табличном виде с возможностью прокрутки и интерактивной фильтрации/сортировки в таблице.
   - **Breakdown Chart:** Диаграмма распределения фильмов по жанрам или рейтингам.
   - **Карусель:** Визуализация обложек фильмов (если добавлена возможность хранить пути к файлам обложек).
2. **Рекомендации фильмов:**
   - Реализовать простую систему рекомендаций фильмов на основе жанров. Например, предлагать фильмы похожих жанров на уже просмотренные фильмы с высоким рейтингом.
3. **Интеграция с OMDb:**
   - Реализовать возможность добавления фильма по названию с автоматическим подтягиванием информации (жанр, год выпуска, обложка) из онлайн-базы данных.

Дополнительная задача

1. **Расширешшая интеграция с OMDb:**

   - **Автоматическое обогащение данных:** При добавлении фильма по названию, приложение должно автоматически запрашивать у API как можно больше информации о фильме:
     - Подробное описание сюжета.
     - Список актеров и режиссеров (с извлечением и разделением на отдельные поля).
     - Постер фильма (и сохранение пути к локальному файлу постера).
     - Рейтинги с разных сайтов (IMDb, Rotten Tomatoes, Metacritic и т.д.).
     - Трейлер.
   - **Обновление данных:** Реализовать функцию обновления данных о фильмах. Приложение должно уметь периодически (или по запросу пользователя) проверять API на наличие новой информации о фильмах в каталоге и обновлять локальные данные.

2. **Визуализация рейтингов:**

   - **Визуализация рейтингов:** Построить графики распределения рейтингов фильмов в каталоге (гистограмма рейтингов, распределение по жанрам). Использовать `ScottPlot` для создания графиков и отображать их (возможно, в виде ASCII-графики в консоли или сохранение в файл изображения).

~ *Вариант №9* ~

- Базовая часть
- A-side
- B-side

Базовая часть

Разработать консольное приложение «Музыкальный органайзер». Приложение должно:

1. **Хранить данные о музыкальных треках в текстовом файле.**

   Формат файла не регламентирован. Например, каждая строка может представлять собой отдельный трек и иметь следующий вид:

   ```
   [Название трека] [Исполнитель] [Альбом] [Год выпуска] [Жанр]
   ```

   Пример файла:

   ```
   [Bohemian Rhapsody] [Queen] [A Night at the Opera] [1975] [Rock]
   [Stairway to Heaven] [Led Zeppelin] [Led Zeppelin IV] [1971] [Rock]
   [Billie Jean] [Michael Jackson] [Thriller] [1982] [Pop]
   ```

   - Приложение должно читать данные о треках из файла при запуске.
   - Путь к файлу каталога должен запрашиваться у пользователя.
   - Приложение должно проверять корректность пути и обрабатывать ошибки.

2. **Предоставлять пользователю возможность:**

   - Просмотра всех треков (в виде списка: Название, Исполнитель, Альбом, Год, Жанр).
   - Добавления нового трека. Запрашивать у пользователя название, исполнителя, альбом, год выпуска, жанр.
   - Редактирования информации о треке. Пользователь выбирает трек и может изменить любое поле.
   - Удаления трека. Пользователь выбирает трек для удаления.

3. **Выводить информацию о треках на консоль в формате:**

   ```
   Каталог треков:
   ----------------------------------
   Название: Bohemian Rhapsody
   Исполнитель: Queen
   Альбом: A Night at the Opera
   Год выпуска: 1975
   Жанр: Rock
   ----------------------------------
   Название: Stairway to Heaven
   Исполнитель: Led Zeppelin
   Альбом: Led Zeppelin IV
   Год выпуска: 1971
   Жанр: Rock
   ----------------------------------
   ...
   ```

4. Сохранять изменения каталога обратно в файл при завершении работы приложения.

[К варианту]

A-side

Основная часть

1. **Фильтрация и сортировка треков:**
   - Фильтрация по исполнителю, альбому, году выпуска (диапазон) или жанру.
   - Сортировка (по возрастанию/убыванию):
     - По названию трека.
     - По исполнителю.
     - По альбому.
     - По году выпуска.
   - Возможность комбинировать фильтры и сортировки.
2. **Статистика:**
   - Вывод общего количества треков.
   - Вывод количества треков по каждому жанру.
   - Вывод количества треков по каждому исполнителю.
   - Вывод количества треков по каждому альбому.
3. **Поиск треков:**
   - Поиск по названию трека, исполнителю или альбому (частичное совпадение).

Дополнительная часть

1. **Плейлисты:**
   - Добавить возможность создавать плейлисты.
   - Пользователь может давать плейлистам названия и добавлять в них треки из каталога.
   - Реализовать просмотр, редактирование и удаление плейлистов.
   - Сохранять информацию о плейлистах (например, в отдельном файле или в основном файле в специальном формате).
2. **Рейтинги треков:**
   - Добавить поле для хранения рейтинга трека (например, от 1 до 5 звезд).
   - Пользователь может ставить рейтинг треку.
   - Отображать средний рейтинг песен в плейлисте при просмотре информации о нем.
   - Добавить возможность фильтрации и сортировки плейлистов по рейтингу.

[К варианту]

B-side

Основная задача

1. **Визуализация с помощью Spectre.Console:**
   - **Таблица:** Отображение треков в табличном виде с возможностью прокрутки и интерактивной фильтрации/сортировки в таблице.
   - **Дерево:** Отображение иерархии «Исполнитель -> Альбом -> Трек». Можно реализовать с помощью `TreeView` из Spectre.Console.
   - **Progress Bar:** Отображение прогресса воспроизведения трека (если добавлена эмуляция воспроизведения). Можно использовать `Progress` из Spectre.Console.
2. **Интеграция с MusicBrainz API:**
   - Реализовать автоматическое получение информации по треку (исполнитель, альбом, год выпуска, жанр, обложка альбома) на основе названия трека и/или исполнителя. Использовать API MusicBrainz (потребуется работа с JSON и HTTP-запросами). Рекомендуется использовать библиотеку `MetaBrainz.MusicBrainz` (NuGet). Она упростит работу с API.
   - Дополнять/исправлять данные в каталоге на основе информации из MusicBrainz.
3. **Воспроизведение аудио:**
   - Добавить воспроизведение трека. При выборе трека выводить, что трек играет, после его окончания сообщать, что трек закончился. Добавить возможность досрочно завершить прослушивание.

Дополнительная задача

1. **Расширенная интеграция с MusicBrainz:**
   - **Поиск по MBID:** Добавить возможность поиска треков по их уникальному идентификатору MusicBrainz (MBID). Хранить это значение для каждого трека.
   - **Получение информации об исполнителе:** Реализовать получение дополнительной информации об исполнителе (страна, дата основания/распада, участники группы).
   - **Получение информации об альбоме:** Реализовать получение дополнительной информации об альбоме (треклист, лейбл, дата релиза в разных странах).
   - **Ссылки:** Добавить возможность отображения ссылок на страницу исполнителя/альбома/трека на MusicBrainz.
2. **Импорт/Экспорт:**
   - Добавить импорт метаданных из файлов CSV.
   - Добавить экспорт метаданных в форматы CSV.
3. **Автоматическое сканирование папки:**
   - Реализовать функцию автоматического сканирования указанной пользователем папки на наличие музыкальных файлов (например, MP3, FLAC, Ogg Vorbis).
   - Извлекать метаданные из файлов **и** использовать MusicBrainz API для уточнения и дополнения информации (как в основной задаче). Это позволит получить более точные и полные данные. Для работы с тегами аудиофайлов можно использовать библиотеку `TagLibSharp`.
   - Обрабатывать случаи, когда MusicBrainz не находит информацию о треке: предлагать пользователю ввести данные вручную или пропустить трек.
   - Реализовать неблокирующий интерфейс, с применением async/await.

~ *Вариант №10* ~

- Базовая часть
- A-side
- B-side

Базовая часть

Базовая часть

Разработать консольное приложение «Менеджер коллекции видеоигр». Приложение должно:

1. **Хранить данные об играх.**

   Формат файла не регламентирован. Например, каждая строка может представлять собой отдельную игру и иметь следующий вид:

   ```
   [Название игры] [Платформа] [Жанр] [Год выпуска] [Статус прохождения]
   ```

   - **Список статусов прохождения:** «Не начата», «В процессе», «Пройдена», «Заброшена», «100% Completion» (можно добавить свои). Важно предусмотреть **несколько** статусов.
   - **Список жанров:** Action, RPG, Strategy, Simulator, Adventure, Puzzle, Fighting, Racing, Sports, и т.д. (предоставить пользователю выбор из **заранее определённого** списка, но с возможностью добавления «Другое»).
   - **Платформа:** PC, PS5, Xbox Series X, Nintendo Switch, Mobile, и т.д. (также предоставить выбор).

   Пример файла:

   ```
   [The Witcher 3: Wild Hunt] [PC, PS5, Xbox Series X] [RPG] [2015] [Пройдена]
   [Cyberpunk 2077] [PC, PS5, Xbox Series X] [RPG] [2020] [В процессе]
   [Hades] [PC, Nintendo Switch] [Roguelike] [2020] [100% Completion]
   [Stardew Valley] [PC, PS4, Xbox One, Nintendo Switch, Mobile] [Simulator] [2016] [Заброшена]
   ```

2. **Путь к файлу:**
   - Путь к файлу коллекции должен запрашиваться у пользователя.
   - Приложение должно проверять корректность введенного пути и обрабатывать ошибки (файл не найден, ошибка чтения).
3. **Предоставлять пользователю возможность:**
   - **Просмотра всех игр** (в виде списка: Название, Платформа, Жанр, Год, Статус).
   - **Добавления новой игры.** Запрашивать у пользователя название, платформу (множественный выбор!), жанр, год выпуска, статус прохождения.
   - **Редактирования информации об игре.** Пользователь выбирает игру и может изменить любое поле.
   - **Удаления игры.** Пользователь выбирает игру для удаления.
4. **Выводить информацию об играх на консоль в формате:** (Сделать вывод в виде таблицы).

   ```
   Каталог игр:
   ------------------------------------------------------
   Название: The Witcher 3: Wild Hunt
   Платформы: PC, PS5, Xbox Series X
   Жанр: RPG
   Год выпуска: 2015
   Статус: Пройдена
   ------------------------------------------------------
   Название: Cyberpunk 2077
   Платформы: PC, PS5, Xbox Series X
   Жанр: RPG
   Год выпуска: 2020
   Статус: В процессе
   ------------------------------------------------------
   ...
   ```

5. **Сохранять изменения каталога обратно в файл при завершении работы приложения.**

[К варианту]

A-side

Основная часть

1. **Фильтрация и сортировка игр:**
   - Фильтрация по платформе, жанру, году выпуска (диапазон), статусу прохождения.
   - Сортировка (по возрастанию/убыванию): по названию, году выпуска, статусу.
   - **Возможность комбинировать фильтры и сортировки.**
2. **Статистика:**
   - Общее количество игр в коллекции.
   - Количество игр по каждой платформе.
   - Количество игр по каждому жанру.
   - Количество игр по статусу прохождения.
   - Самая старая и самая новая игра в коллекции.
3. **Поиск игр:**
   - Поиск по названию игры (частичное совпадение, регистронезависимый).

Дополнительная часть

1. **Рейтинг игр:**
   - Добавить поле «Личный рейтинг» (например, от 1 до 10, или звездная система).
   - Возможность фильтрации и сортировки по рейтингу.
   - Вывод среднего рейтинга по жанрам.
2. **Заметки к играм:**
   - Добавить поле «Заметки» для каждой игры (текстовое поле).
   - Пользователь может добавлять любые комментарии, впечатления, советы по прохождению и т.д.
3. **Экспорт/Импорт:**
   - Реализовать экспорт и импорт данных в/из CSV.

[К варианту]

B-side

Основная задача

1. **Визуализация с помощью `Spectre.Console`:**
   - **Таблица:** Отображение списка игр в виде интерактивной таблицы с возможностью прокрутки, фильтрации и сортировки прямо в таблице.
   - **Дерево (Tree):** Отобразить иерархию «Жанр -> Платформа -> Игра».
   - **Календарь:** Отображение календаря с выделением годов выпуска игр (интенсивность цвета года соответствует количеству игр, выпущенных в этот год).
   - **Bar Chart:** Диаграмма распределения игр по платформам.
2. **Интеграция с IGDB (Internet Game Database):**
   - Реализовать автоматическое получение информации об игре (описание, обложка, разработчик, издатель, ссылки) по названию игры, используя API IGDB (https://api-docs.igdb.com/).
   - **Потребуется API key (бесплатный).**
   - Предлагать пользователю подтвердить найденную игру перед добавлением информации в каталог.
3. **Обложки Игр:**
   - Добавить в приложение возможность сохранять локально обложки игр и отображать их в терминале используя Spectre.Console.

Дополнительная задача

1. **Расширенная интеграция с IGDB:**
   - **Поиск по ID:** добавить поиск игры по IGDB ID.
   - **Связанные игры:** Получать информацию о дополнениях (DLC), сиквелах, приквелах к играм.
   - **Похожие игры:** Реализовать простую систему рекомендаций, предлагая похожие игры на основе жанра, разработчика, и т.д. (используя данные из IGDB).
2. **Управление списками желаемого/пройденного:**
   - Позволить пользователю вести отдельные списки «Хочу поиграть» и «Пройденные игры», помимо основного каталога. Реализовать перемещение игр между списками.
3. **Резервное копирование и восстановление:**
   - Добавить функцию резервного копирования и восстановления данных из файла.

~ *Примечания* ~

А можно свой формат файла для хранения файлов?

Да, конечно. Хоть SQLite используйте для хранения данных.

Если у вас не первый вариант - там нужно использовать указанный в условии формат.

Что делать, если файл уже существует?

Необходимо запросить подтверждение пользователя на перезапись файла.

Требования к фильтрации и сортировкам

Нужно реализовать возможность комбинировать фильтры и сортировки.

Например, установить набор фильтров:
- Сортировка №1
- Фильтрация №1
- Фильтрация №2
- Сортировка №2

И затем предоставить пользователю возможность удалить любой из пунктов. Например, первую фильтрацию, чтобы получилось:
- Сортировка №1
- Фильтрация №2
- Сортировка №2

Все действия применяются последовательно, от первого до последнего, в порядке добавления.