

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

**«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА И ГОСУДАРСТВЕННОЙ СЛУЖБЫ ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

КОЛЛЕДЖ МНОГОУРОВНЕВОГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ОТЧЕТ

По учебной практике УП.01.01 Разработка программных модулей

специальность 09.02.07 Информационные системы и программирование

Выполнили студент:

Зайцев Г.В

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

Гусятинер Л. Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(оценка)

Москва, 2023

Содержание

[Раздел 1. Проектирование консольного приложения 1](#__RefHeading___Toc12469_3557482293)

[1.1 Техническое задание 1](#__RefHeading___Toc12471_3557482293)

[1.1.1. Описание бизнес-процесса 1](#__RefHeading___Toc12473_3557482293)

[1.1.2. Структура базы данных 1](#__RefHeading___Toc12475_3557482293)

[1.1.3. Запросы к базе данных: 2](#__RefHeading___Toc12477_3557482293)

[1.2. Построение диаграммы вариантов использования. 3](#__RefHeading___Toc12479_3557482293)

[1.3 Разработка сценария проекта 4](#__RefHeading___Toc12481_3557482293)

[Раздел 2. Разработка консольного приложения 5](#__RefHeading___Toc12483_3557482293)

[2.1. Состав проекта 5](#__RefHeading___Toc12485_3557482293)

[2.2 Разработка главного модуля 5](#__RefHeading___Toc12487_3557482293)

[2.3 Разработка входящих модулей 10](#__RefHeading___Toc12489_3557482293)

[2.4 Тестирование и отладка 14](#__RefHeading___Toc12491_3557482293)

[2.5 Дневник 15](#__RefHeading___Toc12493_3557482293)

[Приложение 1. Руководство программиста 18](#__RefHeading___Toc12497_3557482293)

[Структура программы 18](#__RefHeading___Toc12501_3557482293)

[Приложение 1. Руководство программиста. Продолжение 19](#__RefHeading___Toc12507_3557482293)

[Приложение 2. Руководство пользователя 20](#__RefHeading___Toc12523_3557482293)

[Приложение 2. Руководство программиста. Продолжение 22](#__RefHeading___Toc12533_3557482293)

[Приложение 3. Листинги 23](#__RefHeading___Toc12541_3557482293)

[Листинг кода: 23](#__RefHeading___Toc12543_3557482293)

# Раздел 1. Проектирование консольного приложения

## Техническое задание

Разработать программу для «Института МОСИНЖ проект», которая будет использоваться для управления задачами и подзадачами. Программа должна обеспечивать возможность создания, редактирования, удаления задач и подзадач, а также просмотра списка всех задач и подзадач.

Программа должна иметь простой и понятный интерфейс, который позволит пользователю легко ориентироваться в функциях программы.

При запуске программы должно отображаться консольное меню с возможностью добавления, редактирования и удаления всех задач и подзадач

Необходимо предусмотреть возможность экспорта данных о задачах и подзадачах в различных форматах (например, в виде таблицы Excel или текстового файла).

### Описание бизнес-процесса

Отделы организации имеют разные задачи за выполнением которых следит Зам.Директора и Директор

Бизнес процесс по поручению задач в организации является важным элементом эффективного функционирования и достижения поставленных целей. Данный процесс включает в себя следующие шаги:

1. Получение поручения задачи: Директор или ответственный сотрудник передает задачу подчиненному или коллеге
2. Выполнение задачи сотрудником
   1. Сотрудник разбивает задачу на подзачи и отправляет количество подзадач и примерный срок выполнения проекта и приступает к её выполнению
   2. Сотрудник делает отчет о выполненных подзадачах начальнику
3. Мониторинг и контроль: в процессе выполнения задачи начальник/зам отслеживает ее прогресс и контролирует соответствие результатов поставленным требованиям и срокам выполнения. Прогрессом выполнения задачи считается выполнение пункта подзадачи
4. Отчетность: по завершении задачи сотрудник составляет отчет, в котором отображены выполненные работы, достигнутые результаты и время, затраченное на выполнение задачи. Этот отчет передается начальнику или ответственному лицу.

### 1.1.2. Структура базы данных

1. Таблица "Задачи.txt":

- id

- Наименование задачи

- ФИО сотрудника ответственный за задачу

- Статус (в процентах)

- Отчетность

- Количество подзадач

- Срок

- Время начала задачи

- Время выполнения

2. Таблица “Подзадачи.txt”

-id\_задачи

-Количество\_подзадач

-Наименование\_подзадачи

-Время\_выполнения

### 1.1.3. Запросы к базе данных:

1. Получить данные о сотруднике и его задачах и подзадачах

2. Получить подзадачи задачи сотрудника

***Пример входных файлов:***

«Задачи.txt»

ID  Название задачи  ФИО сотрудника Статус Отчетность Количество подзадач  Срок Время начала задачи Время выполнения

1;Подготовка отчета;Иванов;50;Еженедельный отчет;3;30;2023-12-10;2023-12-14  
2;Разработка функционала;Петров;75;Спринт 2;5;60;2023-12-12;2023-12-18  
3;Тестирование приложения;Сидоров;100;Финальное тестирование;2;31;2023-12-20;2023-12-24

«Подзадачи.txt»

id\_задачи Количество\_подзадач Наименование\_подзадачи Время\_выполнения

1;1;Сбор информации;2023-12-11  
1;2;Анализ данных;2023-12-12  
1;3;Формирование отчета;2023-12-13  
2;1;Проектирование;2023-12-13  
2;2;Кодирование;2023-12-15  
2;3;Тестирование;2023-12-18  
2;4;Оптимизация;2023-12-19  
3;1;Тестирование функционала;2023-12-21  
3;2;Проверка безопасности;2023-12-23

***Запросы:***

*Запрос 1: Получить данные о клиенте и его заказах*

Входной файл 1. «Задачи.txt»

Входной файл 2. «Подзадачи.txt»

Выходной файл

**Отчет**

Наименование задачи,Сотрудник,Статус,Время выполения задачи

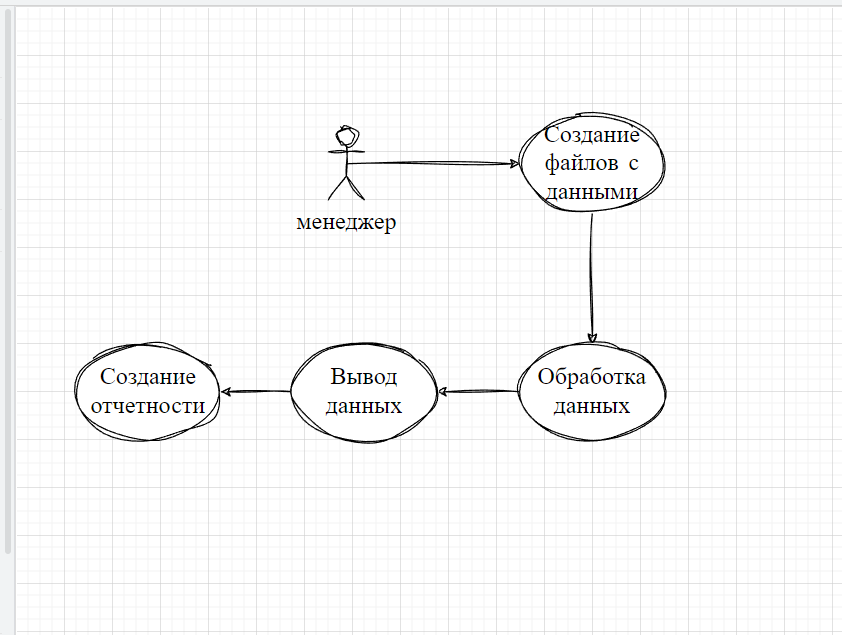
Тестирование приложения,Сидоров,Выполнено,2023-12-24

## 1.2. Построение диаграммы вариантов использования.

Варианты использования

Диаграмма вариантов использования приведена на Рисунке 1.

Рисунок 1. Диаграмма вариантов использования



## 1.3 Разработка сценария проекта

В данном разделе приведен сценарий использования программы пользователем (Рисунок 2).

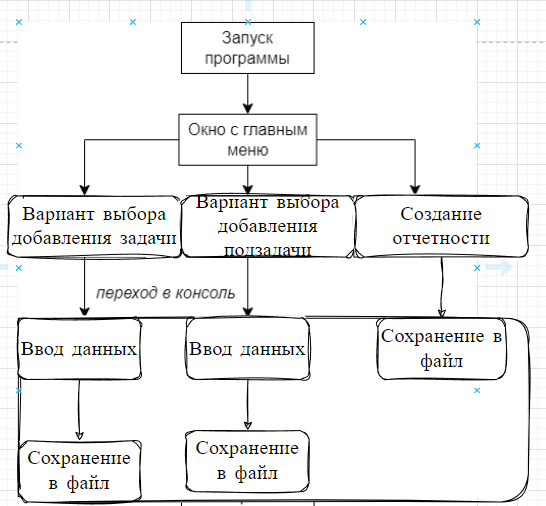


Рисунок 2. Сценарий проекта

# Раздел 2. Разработка консольного приложения

## 2.1. Состав проекта

Класс TaskManager   
Класс **Task**

Класс **Subtask**

## 2.2 Разработка главного модуля

Главный модуль состоит из класса TaskManager. Он выполняется при запуске программы. Его также можно назвать главным меню.

Листинг 1. Главный модуль **TaskManager:**

class TaskManager:  
 def \_\_init\_\_(self):  
 self.tasks\_file\_path = 'Задачи.txt'  
 self.subtasks\_file\_path = 'Подзадачи.txt'  
 self.last\_id = 0  
 self.tasks = self.load\_tasks\_from\_file()  
 self.subtasks = self.load\_subtasks\_from\_file()  
  
 def load\_tasks\_from\_file(self):  
 tasks = []  
 with open(self.tasks\_file\_path, 'r', encoding='utf-8') as file:  
 for line in file:  
 task\_data = line.strip().split(';')  
 task = Task(\*task\_data)  
 tasks.append(task.to\_dict())  
 return tasks  
  
 def load\_subtasks\_from\_file(self):  
 subtasks = []  
 with open(self.subtasks\_file\_path, 'r', encoding='utf-8') as file:  
 for line in file:  
 subtask\_data = line.strip().split(';')  
 subtask = Subtask(\*subtask\_data)  
 subtasks.append(subtask.to\_dict())  
 return subtasks  
  
 def save\_tasks\_to\_file(self):  
 with open(self.tasks\_file\_path, 'w', encoding='utf-8') as file:  
 for task in self.tasks:  
 data = []  
 for key in task:  
 data.append(str(task[key]))  
 file.write(';'.join(data) + '\n')  
  
 def save\_subtasks\_to\_file(self):  
 with open(self.subtasks\_file\_path, 'w', encoding='utf-8') as file:  
 for subtask in self.subtasks:  
 data = []  
 for key in subtask:  
 data.append(str(subtask[key]))  
 file.write(';'.join(data) + '\n')  
  
 def generate\_id(self):  
 #генерация уникального ID для новой задачи  
 self.last\_id += 1  
 return str(self.last\_id)  
  
 def show\_tasks\_menu(self):  
 #отображение меню задач  
 while True:  
 print("\n=== Задачи ===")  
 print("1. Добавить задачу")  
 print("2. Назад")  
  
 choice = input("Выберите действие: ")  
 if choice == '1':  
 self.add\_task()  
 elif choice == '2':  
 break  
 else:  
 print("Некорректный выбор. Пожалуйста, повторите.")  
  
 def add\_task(self):  
 #добавление новой задачи с вводом пользователем  
 task\_name = input("Введите наименование задачи: ")  
 responsible = input("Введите сотрудника, ответственного за задачу: ")  
 status = int(input("Введите статус задачи в процентах: "))  
 report = input("Введите отчетность по задаче: ")  
 subtasks\_count = int(input("Введите количество подзадач задачи: "))  
 data\_srok = int(input("Введите срок выполнения задачи (в Днях, число): "))  
 start\_time = input("Введите время начала задачи (формат: ГГГГ-ММ-ДД): ")  
 completion\_time = input("Введите время выполнения задачи (формат: ГГГГ-ММ-ДД): ")  
  
 new\_task = {  
 'id': self.generate\_id(),  
 'Наименование\_задачи': task\_name,  
 'Сотрудник': responsible,  
 'Статус': status,  
 'Отчетность': report,  
 'Количество\_подзадач': subtasks\_count,  
 'Срок': data\_srok,  
 'Время\_начала\_задачи': start\_time,  
 'Время\_выполнения': completion\_time  
 }  
  
 self.tasks.append(new\_task)  
 self.save\_tasks\_to\_file()  
 self.add\_subtasks(new\_task)  
 print("Задача добавлена успешно.")  
  
 def add\_subtasks(self, task):  
 #добавление подзадач для задачи с вводом пользователем  
 subtasks\_list = []  
 for i in range(task['Количество\_подзадач']):  
 subtask\_name = input(f"Введите наименование подзадачи {i + 1}: ")  
 completion\_time = input(f"Введите время выполнения подзадачи {i + 1}: ")  
 subtasks\_list.append({  
 'id\_задачи': task['id'],  
 'Количество\_подзадач': i + 1,  
 'Наименование\_подзадачи': subtask\_name,  
 'Время\_выполнения': completion\_time  
 })  
 self.subtasks.extend(subtasks\_list)  
 self.save\_subtasks\_to\_file()  
  
 def show\_main\_menu(self):  
 #отображение главного меню  
 while True:  
 print("\n=== Главное меню ===")  
 print("1. Задачи")  
 print("2. Подзадачи")  
 print("3. Сделать отчет")  
 print("4. Выход")  
  
 choice = input("Выберите действие: ")  
 if choice == '1':  
 self.show\_tasks\_menu()  
 elif choice == '2':  
 self.show\_subtasks\_menu()  
 elif choice == '3':  
 self.generate\_report()  
 elif choice == '4':  
 break  
 else:  
 print("Некорректный выбор. Пожалуйста, повторите.")  
  
 def show\_subtasks\_menu(self):  
 # Отображение меню подзадач  
 while True:  
 print("\n=== Подзадачи ===")  
 print("1. Добавить подзадачу к задаче")  
 print("2. Назад")  
  
 choice = input("Выберите действие: ")  
 if choice == '1':  
 self.add\_subtask()  
 elif choice == '2':  
 break  
 else:  
 print("Некорректный выбор. Пожалуйста, повторите.")  
  
 def add\_subtask(self):  
 #добавление подзадачи к задаче с вводом ID задачи пользователем  
 task\_id = input("Введите ID задачи, к которой добавить подзадачу: ")  
 task = next((t for t in self.tasks if t['id'] == task\_id), None)  
  
 if task:  
 self.add\_subtasks(task)  
 print("Подзадачи добавлены успешно.")  
 else:  
 print("Задача с указанным ID не найдена.")  
  
 def generate\_report(self):  
 #генерация отчета по выполненным задачам и сохранение в файл  
 completed\_tasks = [task for task in self.tasks if task['Статус'] == 100]  
  
 with open('Отчет.txt', 'w', encoding='utf-8') as report\_file:  
 report\_file.write("Наименование задачи,Сотрудник,Статус,Время выполения задачи\n")  
 for task in completed\_tasks:  
 status = 'Выполнено' if task['Статус'] == 100 else 'Не выполнено'  
 report\_file.write(  
 f"{task['Наименование\_задачи']},{task['Сотрудник']},{status},{task['Время\_выполнения']}\n"  
 )

Листинг 2. Модуль **Task:**

**class** Task:  
 def \_\_init\_\_(self, task\_id, task\_name='', responsible='', status=0, report='', subtasks\_count=0, deadline='', start\_time='', completion\_time=''):  
 self.id = task\_id  
 self.task\_name = task\_name  
 self.responsible = responsible  
 self.status = int(status)  
 self.report = report  
 self.subtasks\_count = int(subtasks\_count)  
 self.deadline = deadline  
 self.start\_time = start\_time  
 self.completion\_time = completion\_time  
  
 def to\_dict(self):  
 return {  
 'id': self.id,  
 'Наименование\_задачи': self.task\_name,  
 'Сотрудник': self.responsible,  
 'Статус': self.status,  
 'Отчетность': self.report,  
 'Количество\_подзадач': self.subtasks\_count,  
 'Срок': self.deadline,  
 'Время\_начала\_задачи': self.start\_time,  
 'Время\_выполнения': self.completion\_time  
 }

**Листинг 3. Модуль Subtask:**

**class** Subtask:  
 def \_\_init\_\_(self, task\_id, subtask\_count=0, subtask\_name='', completion\_time=''):  
 self.task\_id = task\_id  
 self.subtask\_count = int(subtask\_count)  
 self.subtask\_name = subtask\_name  
 self.completion\_time = completion\_time  
  
 def to\_dict(self):  
 return {  
 'id\_задачи': self.task\_id,  
 'Количество\_подзадач': self.subtask\_count,  
 'Наименование\_подзадачи': self.subtask\_name,  
 'Время\_выполнения': self.completion\_time  
 }

## 2.3 Разработка входящих модулей

Всего разработано 3 модуля:

* Модуль main.py, Класс **TaskManager** , **Task, Subtask**

Описание модулей:

Класс **TaskManager** r

Этот класс управляет задачами и подзадачами. Он включает в себя методы для загрузки данных из файлов, сохранения данных в файлы, добавления задач и подзадач, а также отображения меню для взаимодействия с пользователем.

- client\_id (int): идентификатор клиента.

- name (str): имя клиента.

- city (str): город, в котором находится клиент.

- rating (int): рейтинг клиента.

- manager\_id (int): идентификатор менеджера клиента.

Рассмотрим функцию для получения информации о менеджере клиента на основе идентификатора клиента и её блок-схему:

Листинг 2. Функция для получения информации о менеджере клиента:

def load\_tasks\_from\_file(self):  
 tasks = []  
 with open(self.tasks\_file\_path, 'r', encoding='utf-8') as file:  
 for line in file:  
 task\_data = line.strip().split(';')  
 task = Task(\*task\_data)  
 tasks.append(task.to\_dict())  
 return tasks  
  
def load\_subtasks\_from\_file(self):  
 subtasks = []  
 with open(self.subtasks\_file\_path, 'r', encoding='utf-8') as file:  
 for line in file:  
 subtask\_data = line.strip().split(';')  
 subtask = Subtask(\*subtask\_data)  
 subtasks.append(subtask.to\_dict())  
 return subtasks  
  
def save\_tasks\_to\_file(self):  
 with open(self.tasks\_file\_path, 'w', encoding='utf-8') as file:  
 for task in self.tasks:  
 data = []  
 for key in task:  
 data.append(str(task[key]))  
 file.write(';'.join(data) + '\n')  
  
def save\_subtasks\_to\_file(self):  
 with open(self.subtasks\_file\_path, 'w', encoding='utf-8') as file:  
 for subtask in self.subtasks:  
 data = []  
 for key in subtask:  
 data.append(str(subtask[key]))  
 file.write(';'.join(data) + '\n')

Изображение выглядит как диаграмма, текст, зарисовка, План

Автоматически созданное описание

Рисунок 4. Блок-схема

Класс **Task**

**Этот класс представляет задачи. У него есть атрибуты, соответствующие характеристикам задачи, такие как** id**,** task\_name**,** responsible**, и т.д. Также есть метод** to\_dict()**, который преобразует объект задачи в словарь.**

**class** Task:  
 def \_\_init\_\_(self, task\_id, task\_name='', responsible='', status=0, report='', subtasks\_count=0, deadline='', start\_time='', completion\_time=''):  
 self.id = task\_id  
 self.task\_name = task\_name  
 self.responsible = responsible  
 self.status = int(status)  
 self.report = report  
 self.subtasks\_count = int(subtasks\_count)  
 self.deadline = deadline  
 self.start\_time = start\_time  
 self.completion\_time = completion\_time  
  
 def to\_dict(self):  
 return {  
 'id': self.id,  
 'Наименование\_задачи': self.task\_name,  
 'Сотрудник': self.responsible,  
 'Статус': self.status,  
 'Отчетность': self.report,  
 'Количество\_подзадач': self.subtasks\_count,  
 'Срок': self.deadline,  
 'Время\_начала\_задачи': self.start\_time,  
 'Время\_выполнения': self.completion\_time  
 }

Класс **Subtask**

**Subtask: Этот класс представляет подзадачи. Он имеет атрибуты, такие как** task\_id**,** subtask\_count**,** subtask\_name**, и** completion\_time**. Как и у класса** Task**, есть метод** to\_dict()**, который преобразует объект подзадачи в словарь.**

**class** Subtask:  
 def \_\_init\_\_(self, task\_id, subtask\_count=0, subtask\_name='', completion\_time=''):  
 self.task\_id = task\_id  
 self.subtask\_count = int(subtask\_count)  
 self.subtask\_name = subtask\_name  
 self.completion\_time = completion\_time  
  
 def to\_dict(self):  
 return {  
 'id\_задачи': self.task\_id,  
 'Количество\_подзадач': self.subtask\_count,  
 'Наименование\_подзадачи': self.subtask\_name,  
 'Время\_выполнения': self.completion\_time  
 }

## 2.4 Тестирование и отладка

В ходе написания проекта при попытке запустить программу были получены ошибки (Рисунок 5):

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 5. Ошибки

При проверке кода были исправлены найденные ошибки, в результате при запуске программы ошибок не было (Рисунок 6):

Изображение выглядит как текст, Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Рисунок 6. Успешная сборка

## 2.5 Дневник

|  |  |  |
| --- | --- | --- |
| Дата | Содержание работ | Отметка о выполнении |
| 01.09 | 1. Вводный инструктаж |  |
| 01.09 | 2. Формирование алгоритма решения вычислительной задачи |  |
| 01.09 | 3. Формирование алгоритма решения задачи по обработке данных |  |
| 04.09 | 4. Изучение средств визуализации алгоритмов |  |
| 09..09 | 5. Оформление математического алгоритма при помощи программных средств |  |
| 11.09 | 6. Разработка простой спецификации учебной задачи: Введение. Общее описание |  |
| 16.09 | 7. Разработка простой спецификации учебной задачи: Функции системы |  |
| 22.09 | 8. Разработка простой спецификации учебной задачи: Требования к данным |  |
| 22.09 | 9. Разработка простой спецификации учебной задачи: Требования к внешним интерфейсам |  |
| 22.09 | 10. Разработка простой спецификации учебной задачи: Атрибуты качества |  |
| 29.09 | 11. Оформление спецификации программного продукта. Оформление введения и раздела "Общее описание" |  |
| 29.09 | 12. Оформление раздела "Функции системы" |  |
| 29.09 | 13. Оформление раздела "Требования к внешним данным" |  |
| 04.10 | 14. Оформление раздела "Требования к внешним интерфейсам" |  |
| 04.10 | 15. Оформление раздела "Атрибуты качества" |  |
| 11.10 | 16. Разработка классов главного модуля |  |
| 11.10 | 17. Разработка методов классов и функций главного модуля |  |
| 18.10 | 18. Разработка классов подсистемы вычислений |  |
| 18.10 | 19. Разработка классов подсистемы связи с внешними данными |  |
| 25.10 | 20. Разработка классов подсистемы отчётов |  |
| 25.10 | 21. Разработать тестовые наборы |  |
| 01.11 | 22. Провести тестирование главного модуля |  |
| 01.11 | 23. Провести тестирование модулей |  |
| 08.11 | 24. Провести тестирование программного продукта |  |
| 08.11 | 25. По результатам тестирования провести рефакторинг программного продукта |  |
| 22.11 | Разработать руководство программиста: Обращение к программе |  |
| 29.11 | 27.Разработать руководство программиста: Характеристика программы |  |
| 29.11 | 28. Разработать руководство программиста: Обращение к программе |  |
| 06.12 | 29. Разработать руководство программиста: Входные и выходные данные |  |
| 06.12 | 30. Разработать руководство программиста: Сообщения |  |
| 13.12 | 31. Разработать руководство пользователя: Введение. Назначение и условия применения |  |
| 13.12 | 32. Разработать руководство пользователя: Подготовка к работе |  |

Приложение 1. Руководство программиста

Руководство программиста: TaskManager

TaskManager - это консольная программа, написанная на языке программирования Python, предназначенная для управления задачами и подзадачами. Программа использует простой текстовый формат для хранения данных и обеспечивает базовые операции добавления, редактирования, удаления задач и подзадач.

### Структура программы

Программа состоит из трех основных классов:

* **TaskManager:**
  + Основной класс, отвечающий за управление задачами и подзадачами.
  + Методы для загрузки/сохранения данных из/в файлы.
  + Методы для добавления задач и подзадач, а также отображения пользовательского интерфейса.
* **Task:**
  + Класс, представляющий задачу.
  + Содержит атрибуты, описывающие задачу (например, id, наименование, ответственный и др.).
  + Метод to\_dict() преобразует объект задачи в словарь.
* **Subtask:**
  + Класс, представляющий подзадачу.
  + Атрибуты описывают подзадачу (например, id задачи, наименование, время выполнения).
  + Метод to\_dict() преобразует объект подзадачи в словарь.

Загрузка/Сохранение данных

* Методы load\_tasks\_from\_file() и load\_subtasks\_from\_file() в классе TaskManager загружают данные о задачах и подзадачах из файлов.
* Методы save\_tasks\_to\_file() и save\_subtasks\_to\_file() сохраняют данные о задачах и подзадачах в соответствующие файлы.

Добавление задач и подзадач

* Метод add\_task() в классе TaskManager добавляет новую задачу, получая данные от пользователя.

Приложение 1. Руководство программиста. Продолжение

* Метод add\_subtasks(task) в классе TaskManager добавляет подзадачи к задаче, также получая данные от пользователя.

Генерация отчета

* Метод generate\_report() в классе TaskManager генерирует отчет по выполненным задачам и сохраняет его в файл "Отчет.txt".

Интерфейс пользователя

* Методы show\_main\_menu(), show\_tasks\_menu(), и show\_subtasks\_menu() отвечают за отображение меню и взаимодействие с пользователем.
* Методы add\_task(), add\_subtask(), и generate\_report() выполняют соответствующие действия в ответ на выбор пользователя.

Дополнительные возможности

* Программа предоставляет возможность экспорта данных в текстовые файлы для удобства сохранения и последующего использования.

Зависимости

* Программа не требует установки дополнительных библиотек или зависимостей.

Запуск программы

* Запустите программу, выполнив команду python main.py в командной строке.

Расширение функциональности

* Для расширения функциональности программы, вы можете добавить новые методы, атрибуты или внести изменения в существующие классы.

Заключение

TaskManager предоставляет базовый функционал для управления задачами и подзадачами, и его структура позволяет легко добавлять новые возможности в будущем. Пользуйтесь этим руководством, чтобы более глубоко понять структуру и логику работы программы.

Приложение 2. Руководство пользователя

Руководство пользователя: TaskManager

TaskManager - это простая консольная программа для управления задачами и подзадачами. Эта программа позволяет вам создавать, редактировать и удалять задачи и подзадачи, а также просматривать список всех задач и подзадач. В дополнение к этому, вы можете экспортировать данные о задачах и подзадачах в различные форматы, такие как текстовый файл.

Основные действия

При запуске программы вы увидите главное меню, предлагающее следующие опции:

* **Задачи:**
  + Добавление новой задачи.
  + Просмотр и редактирование существующих задач.
* **Подзадачи:**
  + Добавление новой подзадачи к существующей задаче.
* **Сделать отчет:**
  + Генерация отчета о выполненных задачах и сохранение его в файл.
* **Выход:**
  + Завершение работы программы.

Меню задач

В меню задач у вас есть следующие опции:

* **Добавить задачу:**
  + Введите необходимую информацию о новой задаче.
* **Назад:**
  + Возврат в главное меню.

Меню подзадач

В меню подзадач у вас есть следующие опции:

* **Добавить подзадачу к задаче:**
  + Введите ID задачи, к которой хотите добавить подзадачу.
* **Назад:**
  + Возврат в главное меню.

Приложение 2. Руководство программиста. Продолжение

1. Генерация отчета
2. В этом разделе вы можете создать отчет о выполненных задачах. Отчет будет сохранен в файл "Отчет.txt".
3. Экспорт данных
4. Ваши данные о задачах и подзадачах сохраняются в текстовых файлах. Вы можете найти задачи в файле "Задачи.txt" и подзадачи в файле "Подзадачи.txt". Также, отчет о выполненных задачах сохраняется в файле "Отчет.txt".
5. Завершение работы
6. При завершении работы программы, ваши данные сохраняются, и вы можете продолжить работу с ними при следующем запуске.
7. Теперь вы готовы использовать TaskManager для эффективного управления вашими задачами и подзадачами!

Приложение 3. Листинги

Листинг кода:

class Task:  
 def \_\_init\_\_(self, task\_id, task\_name='', responsible='', status=0, report='', subtasks\_count=0, deadline='', start\_time='', completion\_time=''):  
 self.id = task\_id  
 self.task\_name = task\_name  
 self.responsible = responsible  
 self.status = int(status)  
 self.report = report  
 self.subtasks\_count = int(subtasks\_count)  
 self.deadline = deadline  
 self.start\_time = start\_time  
 self.completion\_time = completion\_time  
  
 def to\_dict(self):  
 return {  
 'id': self.id,  
 'Наименование\_задачи': self.task\_name,  
 'Сотрудник': self.responsible,  
 'Статус': self.status,  
 'Отчетность': self.report,  
 'Количество\_подзадач': self.subtasks\_count,  
 'Срок': self.deadline,  
 'Время\_начала\_задачи': self.start\_time,  
 'Время\_выполнения': self.completion\_time  
 }  
  
class Subtask:  
 def \_\_init\_\_(self, task\_id, subtask\_count=0, subtask\_name='', completion\_time=''):  
 self.task\_id = task\_id  
 self.subtask\_count = int(subtask\_count)  
 self.subtask\_name = subtask\_name  
 self.completion\_time = completion\_time  
  
 def to\_dict(self):  
 return {  
 'id\_задачи': self.task\_id,  
 'Количество\_подзадач': self.subtask\_count,  
 'Наименование\_подзадачи': self.subtask\_name,  
 'Время\_выполнения': self.completion\_time  
 }  
  
  
class TaskManager:  
 def \_\_init\_\_(self):  
 self.tasks\_file\_path = 'Задачи.txt'  
 self.subtasks\_file\_path = 'Подзадачи.txt'  
 self.last\_id = 0  
 self.tasks = self.load\_tasks\_from\_file()  
 self.subtasks = self.load\_subtasks\_from\_file()  
  
 def load\_tasks\_from\_file(self):  
 tasks = []  
 with open(self.tasks\_file\_path, 'r', encoding='utf-8') as file:  
 for line in file:  
 task\_data = line.strip().split(';')  
 task = Task(\*task\_data)  
 tasks.append(task.to\_dict())  
 return tasks  
  
 def load\_subtasks\_from\_file(self):  
 subtasks = []  
 with open(self.subtasks\_file\_path, 'r', encoding='utf-8') as file:  
 for line in file:  
 subtask\_data = line.strip().split(';')  
 subtask = Subtask(\*subtask\_data)  
 subtasks.append(subtask.to\_dict())  
 return subtasks  
  
 def save\_tasks\_to\_file(self):  
 with open(self.tasks\_file\_path, 'w', encoding='utf-8') as file:  
 for task in self.tasks:  
 data = []  
 for key in task:  
 data.append(str(task[key]))  
 file.write(';'.join(data) + '\n')  
  
 def save\_subtasks\_to\_file(self):  
 with open(self.subtasks\_file\_path, 'w', encoding='utf-8') as file:  
 for subtask in self.subtasks:  
 data = []  
 for key in subtask:  
 data.append(str(subtask[key]))  
 file.write(';'.join(data) + '\n')  
  
 def generate\_id(self):  
 #генерация уникального ID для новой задачи  
 self.last\_id += 1  
 return str(self.last\_id)  
  
 def show\_tasks\_menu(self):  
 #отображение меню задач  
 while True:  
 print("\n=== Задачи ===")  
 print("1. Добавить задачу")  
 print("2. Назад")  
  
 choice = input("Выберите действие: ")  
 if choice == '1':  
 self.add\_task()  
 elif choice == '2':  
 break  
 else:  
 print("Некорректный выбор. Пожалуйста, повторите.")  
  
 def add\_task(self):  
 #добавление новой задачи с вводом пользователем  
 task\_name = input("Введите наименование задачи: ")  
 responsible = input("Введите сотрудника, ответственного за задачу: ")  
 status = int(input("Введите статус задачи в процентах: "))  
 report = input("Введите отчетность по задаче: ")  
 subtasks\_count = int(input("Введите количество подзадач задачи: "))  
 data\_srok = int(input("Введите срок выполнения задачи (в Днях, число): "))  
 start\_time = input("Введите время начала задачи (формат: ГГГГ-ММ-ДД): ")  
 completion\_time = input("Введите время выполнения задачи (формат: ГГГГ-ММ-ДД): ")  
  
 new\_task = {  
 'id': self.generate\_id(),  
 'Наименование\_задачи': task\_name,  
 'Сотрудник': responsible,  
 'Статус': status,  
 'Отчетность': report,  
 'Количество\_подзадач': subtasks\_count,  
 'Срок': data\_srok,  
 'Время\_начала\_задачи': start\_time,  
 'Время\_выполнения': completion\_time  
 }  
  
 self.tasks.append(new\_task)  
 self.save\_tasks\_to\_file()  
 self.add\_subtasks(new\_task)  
 print("Задача добавлена успешно.")  
  
 def add\_subtasks(self, task):  
 #добавление подзадач для задачи с вводом пользователем  
 subtasks\_list = []  
 for i in range(task['Количество\_подзадач']):  
 subtask\_name = input(f"Введите наименование подзадачи {i + 1}: ")  
 completion\_time = input(f"Введите время выполнения подзадачи {i + 1}: ")  
 subtasks\_list.append({  
 'id\_задачи': task['id'],  
 'Количество\_подзадач': i + 1,  
 'Наименование\_подзадачи': subtask\_name,  
 'Время\_выполнения': completion\_time  
 })  
 self.subtasks.extend(subtasks\_list)  
 self.save\_subtasks\_to\_file()  
  
 def show\_main\_menu(self):  
 #отображение главного меню  
 while True:  
 print("\n=== Главное меню ===")  
 print("1. Задачи")  
 print("2. Подзадачи")  
 print("3. Сделать отчет")  
 print("4. Выход")  
  
 choice = input("Выберите действие: ")  
 if choice == '1':  
 self.show\_tasks\_menu()  
 elif choice == '2':  
 self.show\_subtasks\_menu()  
 elif choice == '3':  
 self.generate\_report()  
 elif choice == '4':  
 break  
 else:  
 print("Некорректный выбор. Пожалуйста, повторите.")  
  
 def show\_subtasks\_menu(self):  
 # Отображение меню подзадач  
 while True:  
 print("\n=== Подзадачи ===")  
 print("1. Добавить подзадачу к задаче")  
 print("2. Назад")  
  
 choice = input("Выберите действие: ")  
 if choice == '1':  
 self.add\_subtask()  
 elif choice == '2':  
 break  
 else:  
 print("Некорректный выбор. Пожалуйста, повторите.")  
  
 def add\_subtask(self):  
 #добавление подзадачи к задаче с вводом ID задачи пользователем  
 task\_id = input("Введите ID задачи, к которой добавить подзадачу: ")  
 task = next((t for t in self.tasks if t['id'] == task\_id), None)  
  
 if task:  
 self.add\_subtasks(task)  
 print("Подзадачи добавлены успешно.")  
 else:  
 print("Задача с указанным ID не найдена.")  
  
 def generate\_report(self):  
 #генерация отчета по выполненным задачам и сохранение в файл  
 completed\_tasks = [task for task in self.tasks if task['Статус'] == 100]  
  
 with open('Отчет.txt', 'w', encoding='utf-8') as report\_file:  
 report\_file.write("Наименование задачи,Сотрудник,Статус,Время выполения задачи\n")  
 for task in completed\_tasks:  
 status = 'Выполнено' if task['Статус'] == 100 else 'Не выполнено'  
 report\_file.write(  
 f"{task['Наименование\_задачи']},{task['Сотрудник']},{status},{task['Время\_выполнения']}\n"  
 )  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 task\_manager = TaskManager()  
 task\_manager.show\_main\_menu()