EXQT
Готовые ответы — на EXQT
Доступ к ответам приобретается отдельно.
Найти ответы ↗

Методические рекомендации

1. Жизненный цикл программного обеспечения (Software Development Life Cycle, SDLC)

1.1. Понятие и цель

Жизненный цикл программного обеспечения — это совокупность процессов, охватывающих все стадии существования программного продукта: от идеи до вывода из эксплуатации.

Цель — систематизация разработки, обеспечение управляемости, прогнозируемости сроков и качества продукта.

1.2. Основные этапы SDLC

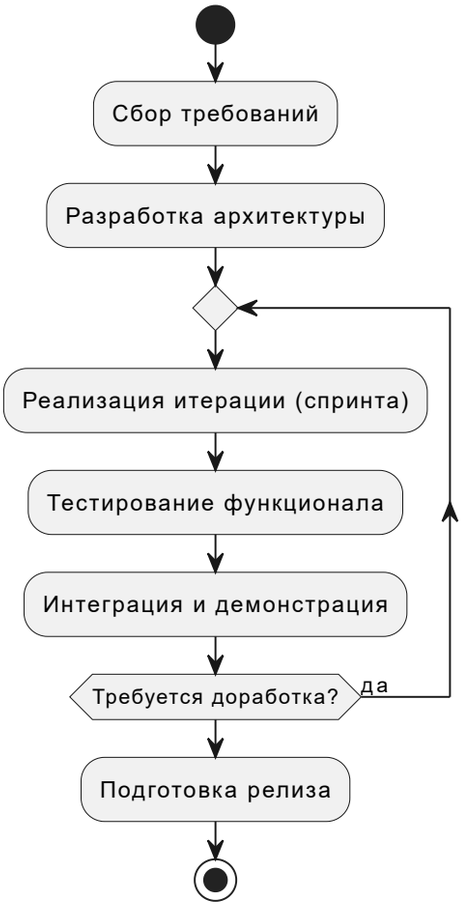
Этап	Цель	Основные результаты
1. Анализ требований	Выявление потребностей пользователей и постановка задач	Техническое задание (ТЗ), спецификации требований
2. Проектирование	Разработка архитектуры и структуры системы	Диаграммы, модели данных, описание модулей
3. Реализация	Программирование и сборка системы	Исходный код, бинарные файлы
4. Тестирование	Проверка корректности и соответствия ТЗ	Тест-кейсы, отчёт о тестировании
5. Внедрение и сопровождение	Развёртывание, обучение, обновления	Руководство пользователя, эксплуатационные документы

1.3. Классические и современные модели жизненного цикла

- Каскадная (Waterfall) — строгая последовательность этапов, без возврата.
- Итеративная / Инкрементальная — создание продукта по частям (итерациям).
- Спиральная — цикл разработки с анализом рисков и уточнением требований.
- Agile / Scrum / Kanban — гибкие методологии, короткие спринты, постоянная обратная связь.

1.4. Пример визуализации итеративного подхода

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru



▶ Показать исходный код PlantUML

1.5. Принципы модульной разработки

- Разделение системы на независимые модули с минимальными связями.
- Каждый модуль выполняет одну логическую функцию (Single Responsibility Principle).
- Повторное использование и тестируемость компонентов.
- Изоляция ошибок и лёгкость модификации.

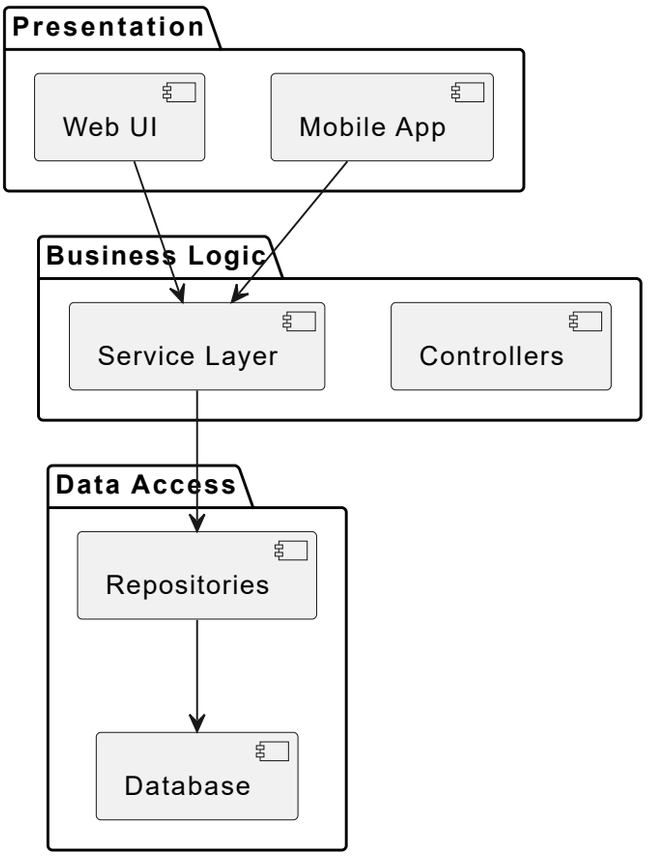
2. Принципы проектирования информационных систем

2.1. Многоуровневая архитектура

Современные приложения строятся по принципу трёхуровневой архитектуры:

1. **Уровень данных (Data Layer)** — базы данных, хранилища, файловые системы.
2. **Бизнес-логика (Business Layer)** — обработка данных, расчёты, правила.
3. **Представление (Presentation Layer)** — интерфейс пользователя, API, отчёты.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru



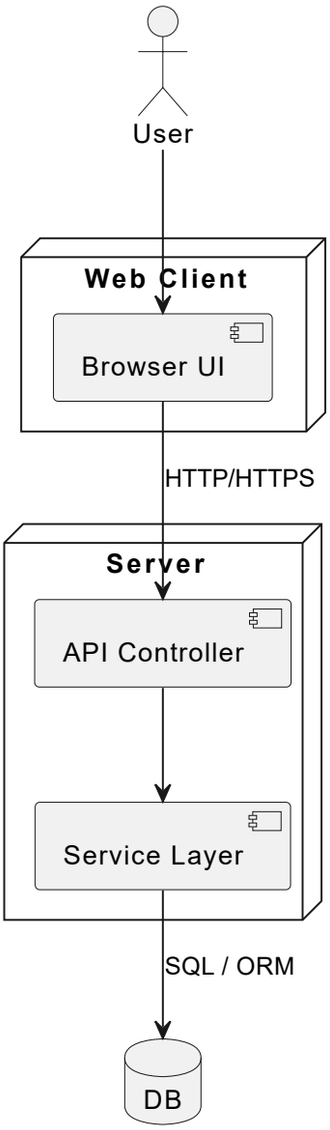
▶ Показать исходный код PlantUML

2.2. Архитектура клиент–серверных приложений

- Клиент инициирует запросы (например, через браузер или мобильное приложение).
- Сервер обрабатывает запросы и возвращает ответы в виде JSON, XML и др.
- Компоненты могут быть распределены по различным машинам.

PlantUML (component):

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru



▶ Показать исходный код PlantUML

2.3. Интерфейсы взаимодействия модулей

- Все модули должны обмениваться данными через интерфейсы и контракты.
- При изменении одного модуля остальные остаются работоспособными.
- Пример: `IDataModule`, `ILogicModule`, `IReportModule`.

3. Объектно-ориентированное программирование (ООП)

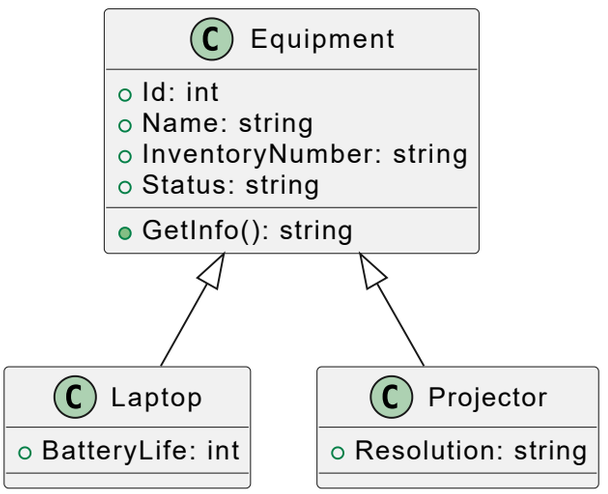
3.1. Основные принципы

Принцип	Суть	Пример
Класс	Шаблон, описывающий структуру данных и поведение	`class Equipment { }`
Объект	Конкретный экземпляр класса	`Equipment item = new Equipment();`

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

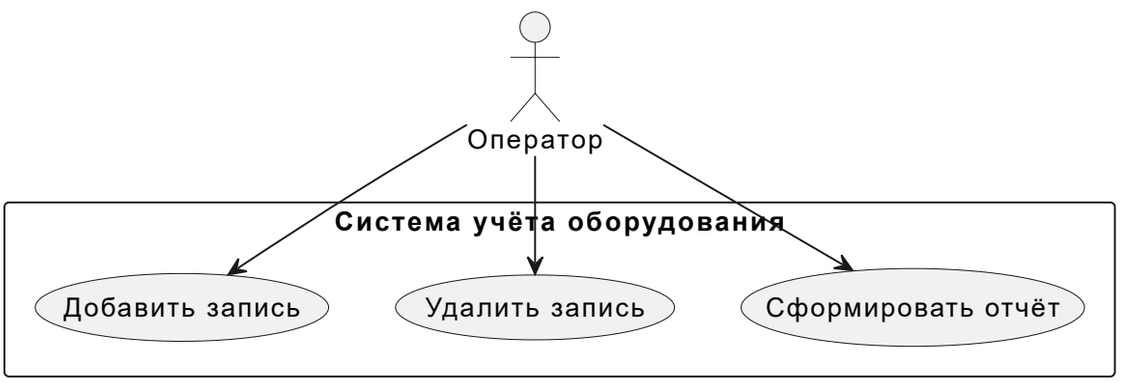
Принцип	Суть	Пример
Инкапсуляция	Сокрытие деталей реализации	`private string status;`
Наследование	Повторное использование кода	`class Laptop : Equipment { }`
Полиморфизм	Разные реализации одного интерфейса	`IReport.Generate()`

3.2. Диаграмма классов (пример)



▶ Показать исходный код PlantUML

3.3. Диаграмма прецедентов (use case)



▶ Показать исходный код PlantUML

4. Моделирование и документирование систем

4.1. Инструменты моделирования

- PlantUML — текстовое описание UML-диаграмм.
- Draw.io, Lucidchart, Miro, StarUML — визуальные редакторы.
- Mermaid.js — встроенная визуализация для HTML/Markdown.

4.2. Документирование

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Документация проекта должна включать:

- техническое задание (ТЗ);
- описание архитектуры;
- схемы UML и ER-диаграммы;
- таблицы данных и форматы обмена;
- инструкции по запуску и эксплуатации.

5. Работа с данными и файлами

5.1. Форматы данных

Формат	Особенности	Пример
JSON	лёгкий, читаемый, используется для API	`{ "id":1, "name":"Laptop" }`
XML	строгий, совместимый со многими системами	`<device><id>1</id></device>`
CSV	табличные данные	`id,name,price`

5.2. Пример сериализации в C#

```
var obj = new Equipment { Id = 1, Name = "Ноутбук", Status = "Активен" };
string json = JsonSerializer.Serialize(obj);
File.WriteAllText("data.json", json);
```

6. Инструментальные средства разработки

Категория	Примеры
IDE / Редакторы	Visual Studio, VS Code, PyCharm
Контроль версий	Git, GitHub, GitLab
Менеджеры зависимостей	NuGet, pip, npm
Системы сборки	MSBuild, Make, Gradle
Тестирование API	Postman, Swagger
CI/CD	GitHub Actions, GitLab CI, Jenkins

7. Методы тестирования

7.1. Классификация тестов

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

- Модульное (Unit) — проверка отдельных функций или классов.
- Интеграционное — взаимодействие модулей.
- Функциональное — проверка пользовательских сценариев.
- Регрессионное — повторное тестирование после изменений.
- Нагрузочное — проверка производительности.

7.2. Пример таблицы тест-кейсов

ID	Тест	Вход	Ожидаемый результат
TC-01	Добавление оборудования	Name="Ноутбук"	Успешное добавление
TC-02	Удаление несуществующей записи	ID=999	Ошибка 404
TC-03	Сохранение JSON	Коллекция 10 объектов	Файл создан

8. Оценка качества и надёжности ПО

8.1. Критерии по ISO/IEC 25010

Категория	Критерий	Пример оценки
Функциональность	Полнота функций	Реализованы все требования ТЗ
Надёжность	Обработка ошибок	Исключения перехватываются корректно
Удобство	UX, навигация	Интерфейс интуитивно понятен
Производительность	Время отклика	< 300 мс при 1000 запросах
Безопасность	Доступ, валидация	Реализованы права ролей
Поддерживаемость	Структура кода	Комментарии, единый стиль

8.2. Пример шкалы оценки качества

Балл	Уровень	Пояснение
0–1	Низкий	Частые ошибки, нет тестов
2–3	Средний	Работает, но не оптимально
4–5	Высокий	Надёжная и стабильная система

9. Экономическая эффективность информационной системы

9.1. Основные формулы

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

ROI (окупаемость инвестиций)

$$\text{ROI} = \frac{\text{Доход} - \text{Затраты}}{\text{Затраты}} \times 100\%$$

NPV (чистая текущая стоимость)

$$\text{NPV} = \sum_{t=1}^{n} \frac{CF_t}{(1+r)^t} - I_0\,,$$

где CF_t — денежный поток в период t, r — ставка дисконтирования, I_0 — первоначальные инвестиции.

PP (Payback Period) — срок окупаемости

Для равномерных потоков: $\text{PP} = \frac{I_0}{CF}$.
Для неравномерных — минимальный T, при котором кумулятивный дисконтированный поток $\sum_{t=1}^{T} \frac{CF_t}{(1+r)^t} \geq I_0$.

9.2. Пример расчёта ROI и NPV

Инвестиции	500 000 ₽
Доход (год 1)	220 000 ₽
Доход (год 2)	300 000 ₽
Доход (год 3)	350 000 ₽
Ставка r	10%

NPV:

$$\text{NPV} = \frac{220\,000}{1.1} + \frac{300\,000}{1.1^2} + \frac{350\,000}{1.1^3} - 500\,000 \approx 204\,000 + 247\,000 + 263\,000 - 500\,000 = 214\,000\,.$$

$\text{NPV} > 0 \Rightarrow$ проект экономически целесообразен.

ROI:

$$\text{ROI} = \frac{870\,000 - 500\,000}{500\,000} \times 100\% = 74\%\,.$$

Срок окупаемости: проект окупается примерно к середине второго года.

10. Документирование программных продуктов

10.1. Состав документации

1. Техническое задание (ТЗ)
2. Руководство пользователя
3. Руководство программиста

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

4. Отчёт о тестировании
5. Экономическое обоснование
6. Паспорт ИС и инструкция по эксплуатации

10.2. Пример структуры ТЗ

1. Назначение и цель системы
2. Требования к функциональности
3. Нефункциональные требования (скорость, надёжность, интерфейс)
4. Архитектура и структура данных
5. Форматы входных/выходных файлов
6. Методы тестирования и приёмки
7. Требования к документированию и сопровождению

10.3. Пример фрагмента руководства пользователя

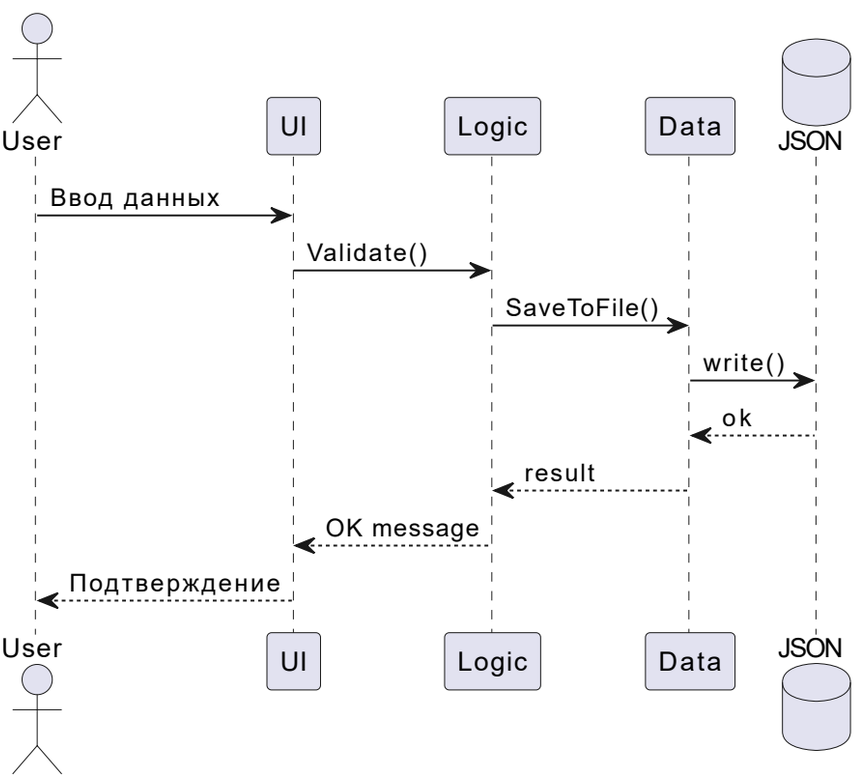
Название: Система учёта оборудования

Назначение: учёт, выдача и возврат техники.

Запуск: открыть LabInventory.exe → выбрать «Просмотр» → кнопка «Добавить».

Сохранение данных: JSON-файл создаётся автоматически при выходе.

11. Пример единой UML-схемы взаимодействия



▶ Показать исходный код PlantUML

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Выполнение задания

Описания задания

1. Спроектировать и реализовать многомодульное приложение, демонстрирующее навыки:
    - управления процессом разработки;
    - программирования отдельных модулей;
    - тестирования и документирования информационной системы.
2. Выбрать предметную область.
3. Определить цели и функции будущего программного продукта, включая:
    - назначение системы и ожидаемый результат её использования;
    - основные задачи и функциональные возможности;
    - категории пользователей и сценарии работы.
4. Разработать архитектуру приложения с разделением на логические модули:
    - модуль хранения данных (файлы JSON или встроенная база данных);
    - модуль бизнес-логики (обработка, поиск, фильтрация, расчёты);
    - модуль интерфейса или отчётности (ввод, отображение, экспорт данных).
5. Создать UML-диаграммы, отражающие структуру и взаимодействие компонентов:
    - диаграмму классов;
    - диаграмму взаимодействия (последовательности или компонентов).
6. Реализовать программный код в выбранной среде разработки:
    - Visual Studio
    - использовать формат обмена данными JSON или встроенную СУБД;
    - обеспечить корректное взаимодействие модулей и обработку ошибок.

0. Предварительные требования

- Установлен .NET SDK 6+ (`dotnet --version`).
- Редактор: Visual Studio 2022 / VS Code.
- Базовые знания C#, ООП, JSON, UML.
- Git (рекомендовано): `git init`, первый коммит после шага 1.

---

1. Создать проект и базовую структуру

1.1. Инициализация проекта

```
dotnet new console -n IntegrationApp
cd IntegrationApp
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

```
mkdir Models Modules Docs
```

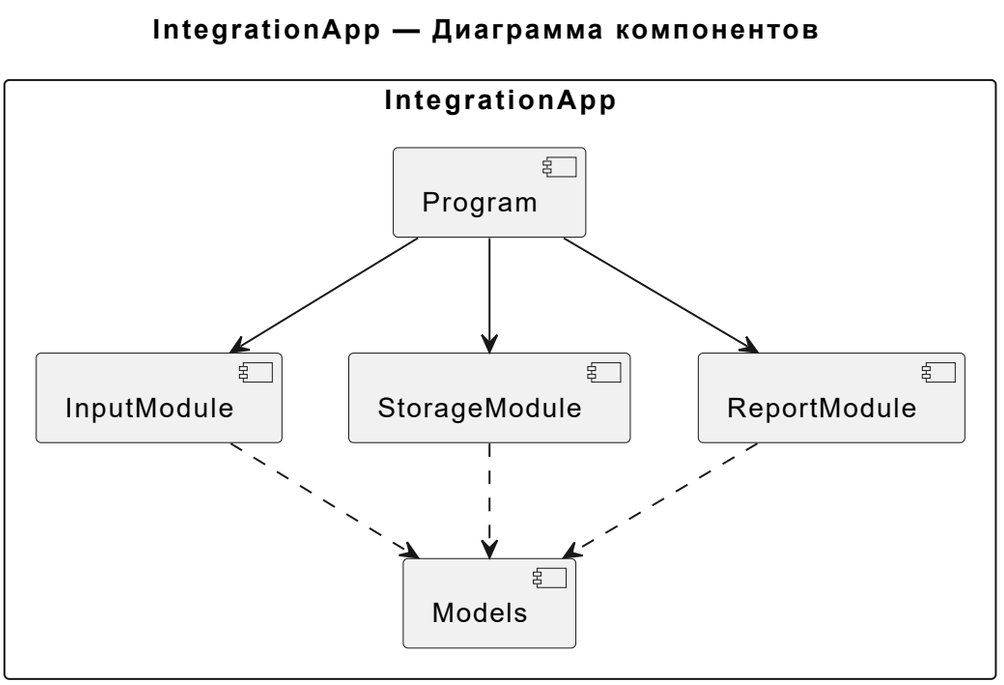
1.2. Структура папок (целевой вид)

```
IntegrationApp/
├── Program.cs
├── IntegrationApp.csproj
├── Models/
│   └── Domain.cs
├── Modules/
│   ├── InputModule.cs
│   ├── StorageModule.cs
│   └── ReportModule.cs
└── Docs/
    ├── components.puml
    └── classes.puml
```

> **Контрольная точка:** проект компилируется `dotnet build`.

2. Спроектировать архитектуру (UML → код)

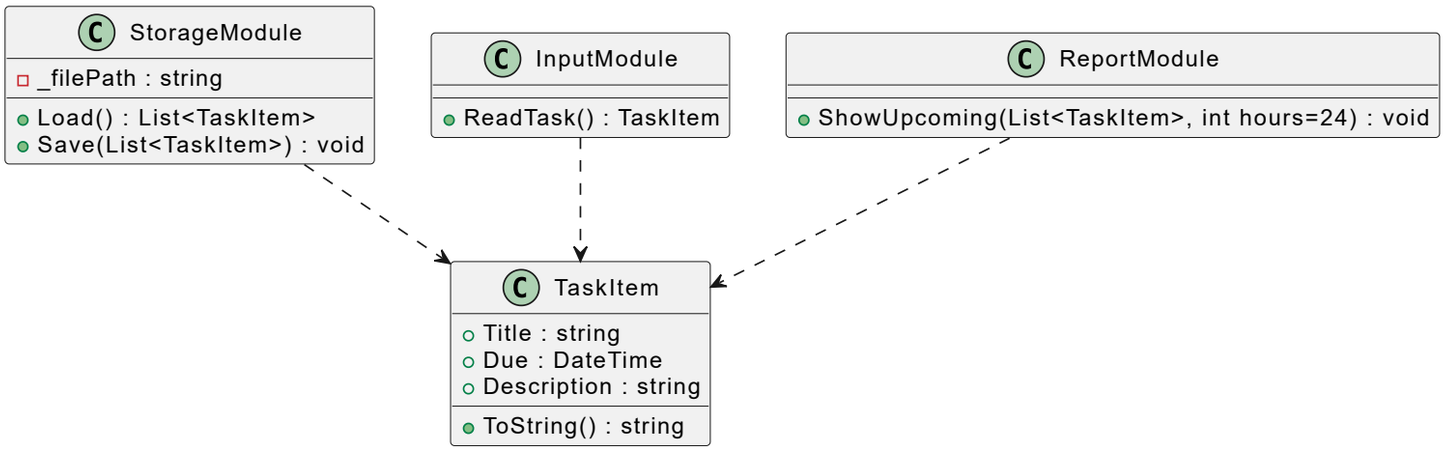
2.1. Диаграмма компонентов (PlantUML)



▶ Показать исходный код PlantUML

2.2. Диаграмма классов (PlantUML)

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru



▶ Показать исходный код PlantUML

> Контрольная точка: схемы согласованы с ТЗ: есть минимум три логических модуля (ввод, хранение JSON, отчёт/уведомления).

---

3. Описать доменную модель (Models)

`Models/Domain.cs`

```
namespace IntegrationApp.Models;

public class TaskItem
{
    public string Title { get; set; } = string.Empty;
    public DateTime Due { get; set; }
    public string Description { get; set; } = string.Empty;

    public override string ToString() => $"{Title} — {Due:g} — {Description}";
}
```

> Контрольная точка: класс компилируется, отражает сущность предметной области.

---

4. Реализовать модуль хранения JSON (StorageModule)

`Modules/StorageModule.cs`

```
using System.Text.Json;
using IntegrationApp.Models;

namespace IntegrationApp.Modules;
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
public class StorageModule
{
    private readonly string _filePath;
    private readonly JsonSerializerOptions _opts = new() { WriteIndented = true };

    public StorageModule(string filePath = "tasks.json")
        => _filePath = filePath;

    public List<TaskItem> Load()
    {
        if (!File.Exists(_filePath)) return new List<TaskItem>();
        var json = File.ReadAllText(_filePath);
        return JsonSerializer.Deserialize<List<TaskItem>>(json) ?? new List<TaskItem
    }

    public void Save(List<TaskItem> tasks)
    {
        var json = JsonSerializer.Serialize(tasks, _opts);
        File.WriteAllText(_filePath, json);
    }
}
```

> Контрольная точка: можно вручную создать `tasks.json`, проверить чтение/запись.

5. Реализовать модуль ввода с валидацией (InputModule)

`Modules/InputModule.cs`

```
using IntegrationApp.Models;

namespace IntegrationApp.Modules;

public static class InputModule
{
    public static TaskItem ReadTask()
    {
        Console.Write("Название: ");
        var title = (Console.ReadLine() ?? "").Trim();

        DateTime due;
        while (true)
        {
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
            Console.Write("Срок (yyyy-MM-dd HH:mm): ");
            if (DateTime.TryParse(Console.ReadLine(), out due)) break;
            Console.WriteLine("Некорректная дата/время. Повторите.");
        }

        Console.Write("Описание: ");
        var desc = (Console.ReadLine() ?? "").Trim();

        return new TaskItem { Title = title, Due = due, Description = desc };
    }
}
```

> Контрольная точка: ввести задачу, получить корректный объект `TaskItem`.

6. Реализовать модуль отчётов/уведомлений (ReportModule)

`Modules/ReportModule.cs`

```
using IntegrationApp.Models;

namespace IntegrationApp.Modules;

public static class ReportModule
{
    public static void ShowUpcoming(List<TaskItem> tasks, int hours = 24)
    {
        var now = DateTime.Now;
        var upcoming = tasks
            .Where(t => t.Due >= now && t.Due <= now.AddHours(hours))
            .OrderBy(t => t.Due)
            .ToList();

        if (!upcoming.Any())
        {
            Console.WriteLine($"Нет задач в ближайшие {hours} часов.");
            return;
        }

        Console.WriteLine($"Ближайшие задачи (<= {hours} ч):");
        foreach (var t in upcoming) Console.WriteLine("- " + t);
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

```
    }
}
```

> Контрольная точка: при наличии задач в интервале виден корректный список.

7. Собрать взаимодействие в Program.cs (интеграция модулей)

Program.cs

```
using IntegrationApp.Models;
using IntegrationApp.Modules;

Console.OutputEncoding = System.Text.Encoding.UTF8;

var storage = new StorageModule();
var tasks = storage.Load();

bool exit = false;
while (!exit)
{
    Console.WriteLine("\n=== IntegrationApp ===");
    Console.WriteLine("1. Добавить задачу");
    Console.WriteLine("2. Показать ближайшие задачи (24 ч)");
    Console.WriteLine("3. Сохранить и выйти");
    Console.Write("Выбор: ");
    var choice = Console.ReadLine();

    try
    {
        switch (choice)
        {
            case "1":
                var t = InputModule.ReadTask();
                tasks.Add(t);
                Console.WriteLine("Задача добавлена.");
                break;
            case "2":
                ReportModule.ShowUpcoming(tasks, 24);
                break;
            case "3":
                storage.Save(tasks);
                Console.WriteLine("Сохранено. Выход...");
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru

```
                exit = true;
                break;
            default:
                Console.WriteLine("Неизвестная команда.");
                break;
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Ошибка: " + ex.Message);
    }
}
```

> Контрольная точка: приложение запускается `dotnet run`, добавляет задачи, сохраняет `tasks.json`.

---

8. Добавить обработку ошибок и логические проверки

- В `StorageModule.Save/Load` уже есть базовая защита.
- При вводе дат — повторный запрос.
- Добавьте доп. проверки: пустое название, дата в прошлом и т. п.

```
if (string.IsNullOrWhiteSpace(title)) throw new ArgumentException("Название не может
if (due < DateTime.Now) Console.WriteLine("Внимание: срок в прошлом.");
```

> Контрольная точка: негативные сценарии не «роняют» программу.

---

9. Простые интеграционные тесты (без фреймворка)

Создайте вспомогательный метод для автопроверки связок «ввод→хранение→отчёт»:

`Program.cs (временный блок)`

```
static void IntegrationSmokeTest()
{
    var storage = new StorageModule("test.json");
    var list = new List<TaskItem>
    {
        new() { Title="Тест", Due = DateTime.Now.AddHours(1), Description="Smoke" }
    };
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
    storage.Save(list);
    var read = storage.Load();
    ReportModule.ShowUpcoming(read, 2);
    File.Delete("test.json");
}
// Вызов один раз вручную:
// IntegrationSmokeTest();
```

> Контрольная точка: тест подтверждает корректную связку модулей.
> (При желании используйте MSTest/xUnit, но для практики достаточно «дымового» теста.)

---

10. Документация (отчёт и схемы)

10.1. Что включить в отчёт

1. Описание архитектуры: роли модулей, принципы модульности, пространства имён.
2. UML-схемы: компоненты, классы (см. шаг 2).
3. Ключевые фрагменты кода (по модулям) с краткими пояснениями.
4. Тестирование: сценарии, результаты, скриншоты запуска.
5. Выводы и перспективы развития (валидация, GUI, БД и т. п.).

10.2. Пример описания интерфейсов взаимодействия

- `Program → InputModule: TaskItem ReadTask()`
- `Program → StorageModule: List<TaskItem> Load(), void Save(List<TaskItem>)`
- `Program → ReportModule: void ShowUpcoming(List<TaskItem>, int)`

Пример тестового проекта: github.com/OlgaKraven/PP.PM.05-testProject.git

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер disynergy.ru