Продолжаем шпаргалку! Допишем недостающие элементы, включая:

* Полный WPF-проект под C# с аналогичными классами
* Логика интерфейса
* Подключение базы данных
* Расчёт продукции
* Идеи по оформлению и финальной сдаче

**6. Реализация в WPF (на C# под Visual Studio)**

**📁 Структура проекта**

WpfMaterialsApp/

├── App.xaml

├── App.xaml.cs

├── MainWindow.xaml / .cs // Главное окно со списком материалов

├── EditMaterialWindow.xaml / .cs // Форма добавления/редактирования

├── UsageWindow.xaml / .cs // Где используется материал

├── Models/

│ ├── Material.cs

│ ├── MaterialType.cs

│ ├── Product.cs

│ ├── ProductType.cs

│ ├── MaterialProduct.cs

├── Services/

│ ├── DatabaseContext.cs

│ ├── CalculatorService.cs

├── Resources/ // Иконки, логотип, стиль

│ ├── logo.png

│ ├── icon.ico

└── warehouse.db // SQLite база

**📦 Классы модели (Models/...)**

public class Material {

public int Id { get; set; }

public string Name { get; set; }

public int TypeId { get; set; }

public decimal Price { get; set; }

public string Unit { get; set; }

public decimal QuantityInPack { get; set; }

public decimal QuantityInStock { get; set; }

public decimal MinQuantity { get; set; }

}

public class MaterialType {

public int Id { get; set; }

public string Name { get; set; }

public double LossPercentage { get; set; }

}

public class Product {

public int Id { get; set; }

public string Name { get; set; }

public string Article { get; set; }

public int TypeId { get; set; }

public decimal MinPartnerPrice { get; set; }

}

public class ProductType {

public int Id { get; set; }

public string Name { get; set; }

public double Coefficient { get; set; }

}

public class MaterialProduct {

public int MaterialId { get; set; }

public int ProductId { get; set; }

public decimal RequiredQuantity { get; set; }

}

**🔧 Services/DatabaseContext.cs**

public class DatabaseContext

{

private SQLiteConnection connection;

public DatabaseContext()

{

connection = new SQLiteConnection("Data Source=warehouse.db");

connection.Open();

}

public List<Material> GetAllMaterials()

{

var result = new List<Material>();

var command = new SQLiteCommand("SELECT \* FROM Materials", connection);

var reader = command.ExecuteReader();

while (reader.Read())

{

result.Add(new Material {

Id = Convert.ToInt32(reader["id"]),

Name = reader["name"].ToString(),

TypeId = Convert.ToInt32(reader["type\_id"]),

Price = Convert.ToDecimal(reader["price"]),

Unit = reader["unit"].ToString(),

QuantityInPack = Convert.ToDecimal(reader["quantity\_in\_pack"]),

QuantityInStock = Convert.ToDecimal(reader["quantity\_in\_stock"]),

MinQuantity = Convert.ToDecimal(reader["min\_quantity"])

});

}

return result;

}

public void Close() => connection.Close();

}

**⚙ Services/CalculatorService.cs**

public static class CalculatorService

{

public static decimal CalculateRequiredQuantity(int materialId, SQLiteConnection conn)

{

var command = new SQLiteCommand("SELECT SUM(required\_quantity) FROM MaterialProducts WHERE material\_id = @id", conn);

command.Parameters.AddWithValue("@id", materialId);

var result = command.ExecuteScalar();

return result == DBNull.Value ? 0 : Convert.ToDecimal(result);

}

public static int CalculateProductAmount(int productTypeId, int materialTypeId, int rawQuantity, double param1, double param2, SQLiteConnection conn)

{

var prodCmd = new SQLiteCommand("SELECT coefficient FROM ProductTypes WHERE id = @id", conn);

prodCmd.Parameters.AddWithValue("@id", productTypeId);

var coeff = Convert.ToDouble(prodCmd.ExecuteScalar() ?? -1);

var matCmd = new SQLiteCommand("SELECT loss\_percentage FROM MaterialTypes WHERE id = @id", conn);

matCmd.Parameters.AddWithValue("@id", materialTypeId);

var loss = Convert.ToDouble(matCmd.ExecuteScalar() ?? -1);

if (coeff <= 0 || loss < 0) return -1;

var requiredPerUnit = param1 \* param2 \* coeff;

var usable = rawQuantity \* (1 - loss / 100);

return (int)(usable / requiredPerUnit);

}

}

**🖼 Пример окна MainWindow.xaml.cs**

private void LoadMaterials()

{

var materials = db.GetAllMaterials();

materialDataGrid.ItemsSource = materials;

}

private void ShowUsage\_Click(object sender, RoutedEventArgs e)

{

var selected = (Material)materialDataGrid.SelectedItem;

if (selected == null) return;

var window = new UsageWindow(selected.Id);

window.ShowDialog();

}

**💅 Оформление (Приложение 2)**

| **Элемент** | **Значение** |
| --- | --- |
| **Шрифт** | Constantia |
| **Цвет фона** | #FFFFFF |
| **Цвет панели** | #BFD6F6 |
| **Цвет текста** | #405C73 |
| **Иконка** | Resources/icon.ico |
| **Логотип** | Resources/logo.png (в верхней части главного окна) |

**📤 Что нужно сдать:**

| **Файл** | **Содержание** |
| --- | --- |
| warehouse.db | База данных SQLite |
| DatabaseScript.sql | SQL-скрипт |
| WpfMaterialsApp/ | Исходный код |
| WpfMaterialsApp.exe | Скомпилированное приложение |
| ERDiagram.pdf | ER-диаграмма из SQLiteStudio или SSMS |
| README.txt | Как запускать проект и что где лежит |

**✅ Рекомендации перед экзаменом:**

1. Создай базу через SQLiteStudio (или MS SQL + SSMS).
2. Убедись, что работает расчёт требуемого количества.
3. При изменении данных — обновляй список.
4. Отображай сообщения об ошибках через MessageBox.Show(...).
5. Сделай резервные копии проекта перед финальной сдачей.
6. Прогони проект хотя бы один раз полностью перед экзаменом.

Если хочешь — я могу собрать весь WPF-проект в виде архива ZIP и отправить тебе.  
Пиши: **«Сделай архив проекта WPF»**, и я его подготовлю под твою структуру.

Ты справишься. Осталось только грамотно всё показать 👏