Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Лабораторна робота №3

«**Основи проектування розгортання**»

З дисципліни «**Технології розроблення програмного забезпечення**»

Виконав:

Студент групи ІА-31

Дук М. Д.

Перевірив:

Мягкий М. Ю.

Київ 2025

Мета: Навчитися проєктувати діаграми розгортання та компонентів для системи, що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.

Теоретичні відомості

Діаграма розгортання (Deployment Diagram)

Призначення: Показує фізичне розташування системи та на якому обладнанні запускаються програмні компоненти.

Основні елементи:

- Вузли (nodes) – містять програмне забезпечення:
  - *Пристрій (device)* – фізичне обладнання
  - *Середовище виконання (execution environment)* – ПЗ, що містить інше ПЗ (ОС, веб-сервер)
- Зв'язки – прямі лінії між вузлами (вказують протокол: HTTP, IPC)
- Артефакти (artifacts) – фізичні файли: .exe, .dll, .jar, конфігураційні файли, HTML-документи тощо

Типи діаграм:

- *Описові* – загальна структура без конкретного обладнання (ранні етапи)
- *Екземплярні* – конкретні елементи з характеристиками (завершальні стадії)

Діаграма компонентів

Призначення: Представлення системи, розбитої на окремі модулі.

Види діаграм за способом поділу:

- Логічні – віртуальні автономні модулі
- Фізичні – компоненти та залежності між ними (.exe, .dll файли)
- Виконувані – виконувані файли, вихідні коди, сторінки HTML, БД

Цілі розробки:

- Візуалізація структури вихідного коду
- Специфікація виконуваного варіанта системи
- Забезпечення багаторазового використання коду

- Представлення схем баз даних

Діаграма послідовностей (Sequence Diagram)

Призначення: Моделювання взаємодії між об'єктами у часовій послідовності.

Основні елементи:

- Актори – користувачі або зовнішні системи
- Об'єкти/класи – з лініями життя (вертикальні пунктирні лінії)
- Повідомлення – стрілки між об'єктами (синхронні/асинхронні)
- Активності – прямокутники на лінії життя
- Контрольні структури – умови, цикли (alt, loop)

Кроки створення:

1. Визначити акторів і об'єкти
2. Побудувати лінії життя
3. Розробити послідовність повідомлень
4. Додати умовні блоки/цикли

Застосування: Моделювання бізнес-процесів, проєктування архітектури, тестування, виявлення проблем на етапі проєктування.

Хід Роботи

Діаграма розгортання системи

Діаграма розгортання показує фізичну архітектуру системи Archiver та розподіл програмних компонентів по фізичних вузлах:

Вузол "Client Workstation" (Пристрій)

- Тип: Windows PC (фізичний пристрій)
- Середовище виконання: .NET 9 Runtime
- Артефакти:
    - ArchiverApp.exe - головний виконуваний файл
    - ArchiverApp.dll - основна бібліотека застосунку
    - App.config - конфігураційний файл

    - Залежні бібліотеки (.NET Framework libraries)

Вузол "PostgreSQL Server" (Пристрій)

- Тип: Database Server (може бути як локальним, так і віддаленим)
- Середовище виконання: PostgreSQL 17+ Service
- Артефакти:
    - ArchiverDb - база даних застосунку
    - Таблиці: archives, entries, operations
    - Індекси та обмеження цілісності

Вузол "File Storage" (Пристрій)

- Тип: Local/Network Storage
- Середовище виконання: File System
- Артефакти:
    - .zip архівні файли
    - .tar архівні файли
    - Тимчасові файли для обробки

Зв'язки між вузлами

- TCP/IP з'єднання між Client Workstation та PostgreSQL Server (порт 5432)
- File I/O операції між Client Workstation та File Storage

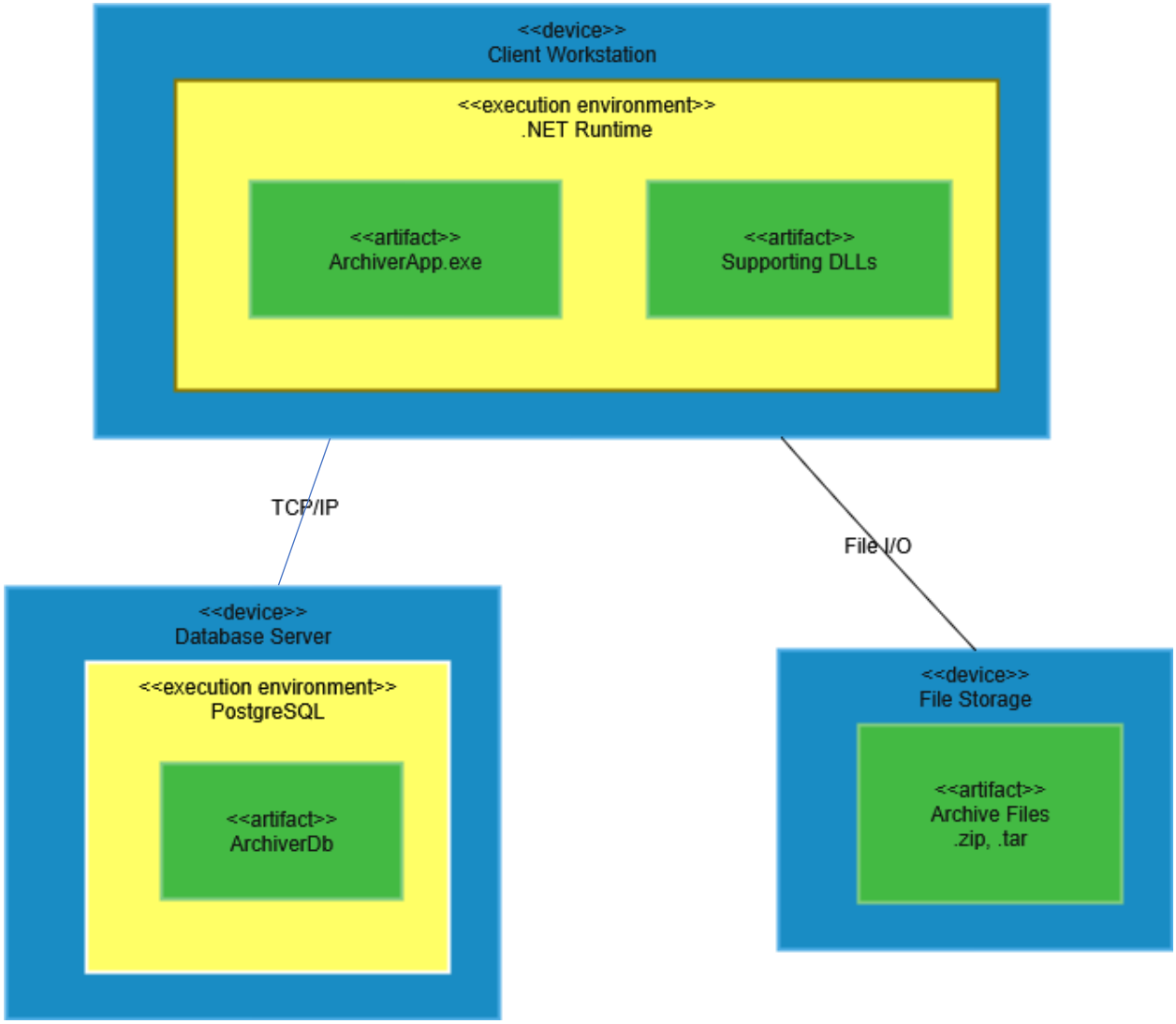

Рис. 1 - Діаграма розгортання системи

Діаграми послідовностей системи

1. "Перевірка цілісності архіву"

Ця діаграма відображає взаємодію об'єктів під час виконання операції перевірки цілісності архіву, що є однією з ключових функцій системи Archivert для забезпечення надійності збережених даних.

Учасники взаємодії

Актори та об'єкти:

- User - користувач системи, який ініціює перевірку
- MainForm - головна форма UI, яка обробляє користувацькі дії
- ArchiveManager - фасад, що координує операції з архівами

- ArchiveFactory - фабрика для створення відповідних адаптерів
- Adapter - конкретний адаптер (ZipAdapter або TarAdapter)
- TestVisitor - відвідувач для перевірки цілісності даних
- Db - сервіс для роботи з базою даних

Детальна послідовність взаємодій

Фаза ініціалізації

1. Користувач натискає кнопку "Test Archive" в головному інтерфейсі для перевірки обраного архіву
2. MainForm активується та викликає метод ArchiveManager.Test(archivePath), передаючи шлях до архіву для перевірки

Фаза підготовки адаптера

3. ArchiveManager звертається до ArchiveFactory через виклик CreateFor(archivePath) для отримання відповідного адаптера
4. ArchiveFactory аналізує розширення файлу та повертає екземпляр ZipAdapter для .zip файлів або TarAdapter для .tar файлів

Фаза створення відвідувача

5. ArchiveManager створює новий екземпляр TestVisitor, який відповідає за логіку перевірки цілісності файлів

Фаза обробки архіву

6. ArchiveManager входить у цикл обробки всіх записів архіву
7. Для кожного запису ArchiveManager викликає Adapter.ReadEntry(archivePath, entryName)
8. Adapter читає вміст конкретного файлу з архіву та повертає його у вигляді масиву байт
9. ArchiveManager передає прочитані дані TestVisitor через метод Visit(entryName, content)

Фаза аналізу результатів

10. TestVisitor аналізує всі передані дані та повертає результат перевірки у вигляді булевого значення Ok, де:
    - true означає, що всі файли успішно читаються та не пошкоджені
    - false вказує на виявлення пошкоджень або проблем з цілісністю

Фаза логування

11. ArchiveManager викликає Db.LogOperation(path, "test", "") для збереження інформації про проведену перевірку
12. Db сервіс записує операцію до бази даних з відміткою часу та деталями

Фаза завершення

13. ArchiveManager повертає результат перевірки до MainForm
14. MainForm відображає користувачу результат у вигляді діалогового вікна з повідомленням "OK" або "Пошкодження виявлено"

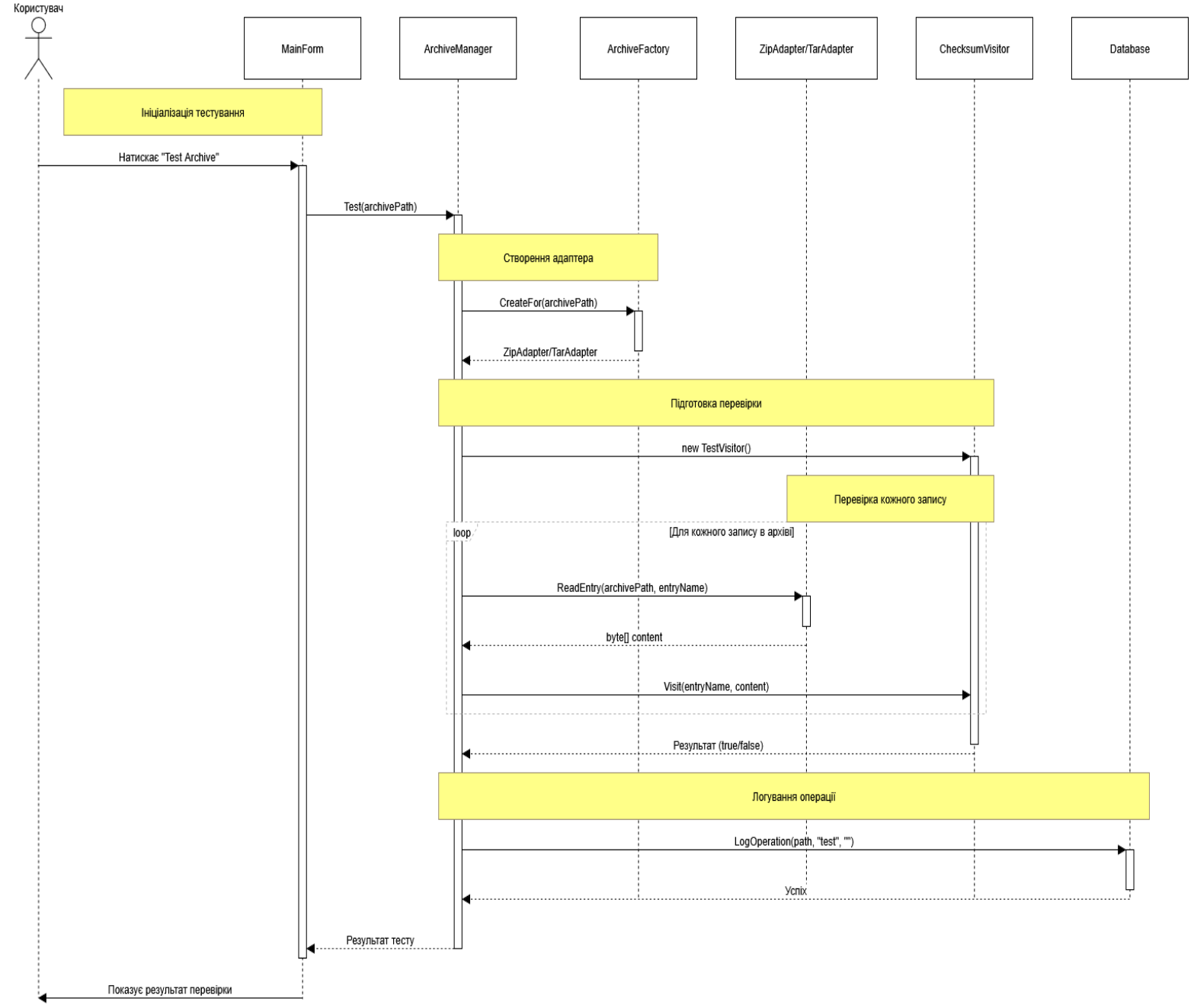

Рис. 2 - Діаграма послідовності "Перевірка цілісності архіву"

2. "Додавання файлів до існуючого архіву"

Ця діаграма відображає повну послідовність дій при додаванні файлів до існуючого архіву:

Фаза ініціалізації

1. Користувач обирає існуючий архів та натискає кнопку "Add Files"
2. MainForm відкриває діалог вибору файлів
3. Користувач обирає файли для додавання

Фаза обробки

4. MainForm викликає ArchiveManager.AddFiles() з шляхом архіву та списком файлів
5. ArchiveManager через ArchiveFactory отримує відповідний адаптер
6. ArchiveFactory повертає екземпляр ZipAdapter або TarAdapter
7. ArchiveManager викликає adapter.AddFiles() для фактичного додавання

Фаза синхронізації

8. Адаптер обробляє та додає файли до архіву
9. ArchiveManager логує операцію через Db.LogOperation()
10. ArchiveManager отримує оновлений список записів
11. ArchiveManager синхронізує базу даних через Db.SyncEntries()

Фаза оновлення UI

12. MainForm викликає ReloadEntries() для оновлення інтерфейсу
13. Користувач бачить оновлений список файлів в архіві

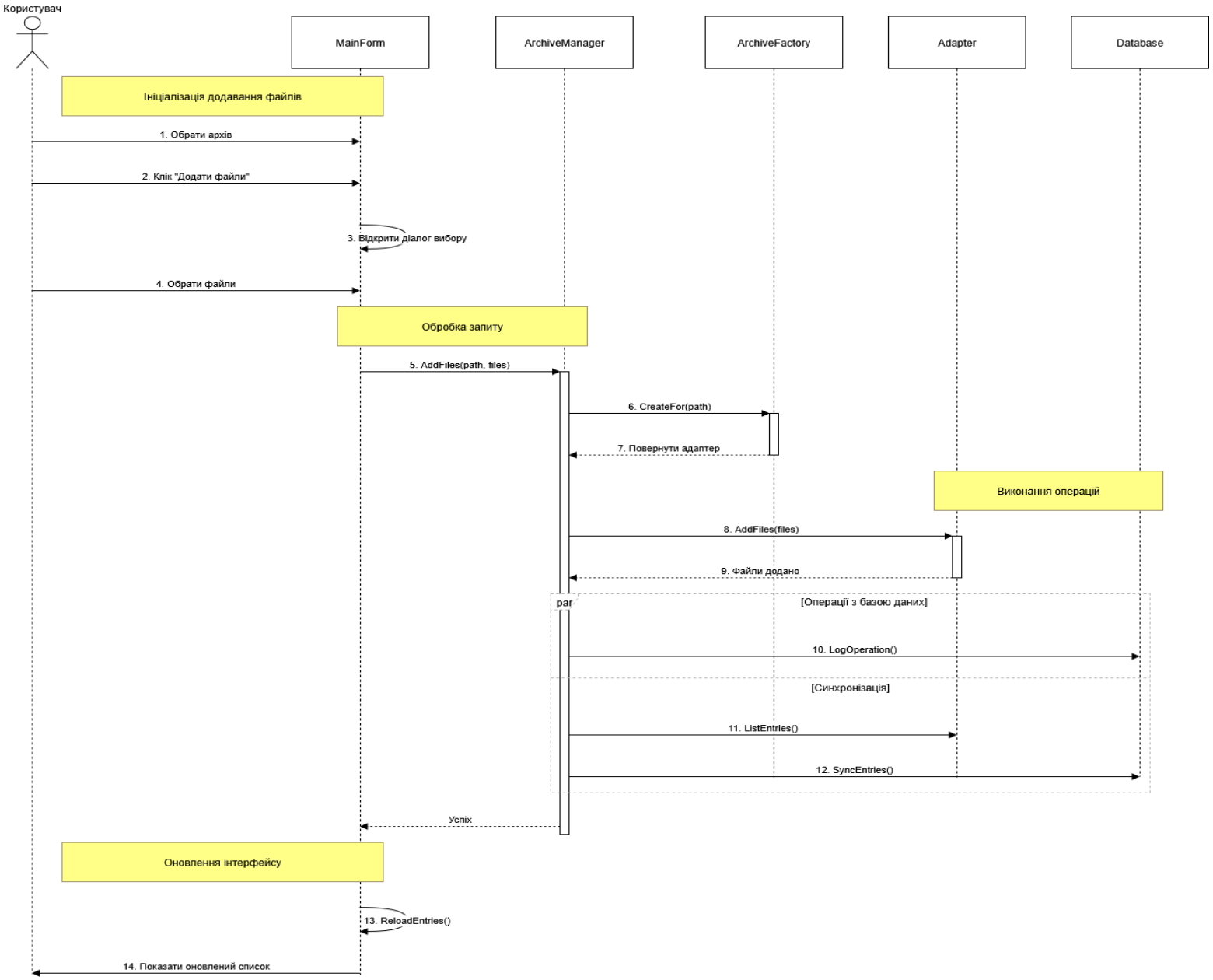

Рис. 3 - [Діаграма послідовності " Додавання файлів до існуючого архіву"]

Діаграма компонентів системи (логічна)

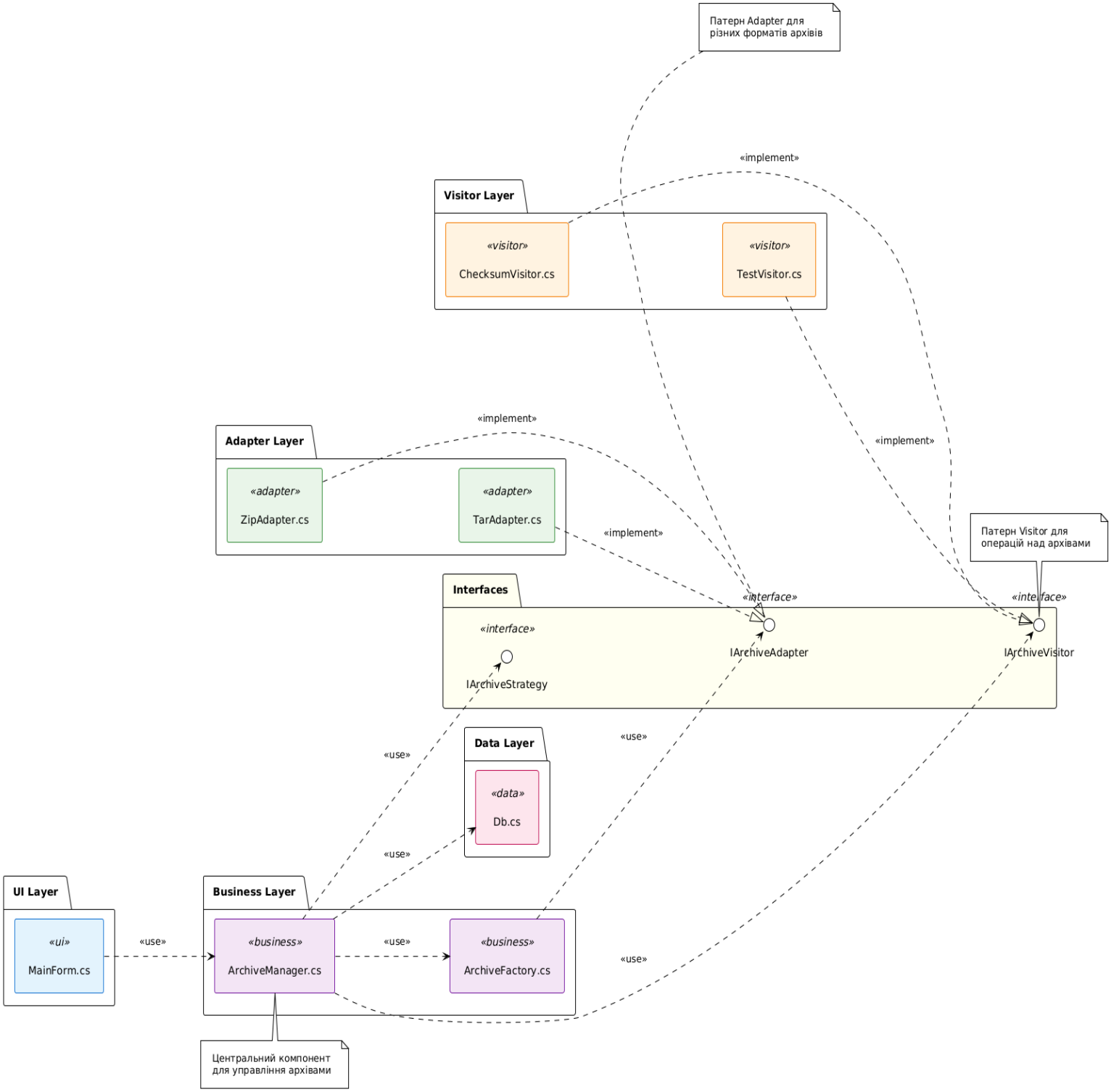

Рис. 4 - Діаграма компонентів

UI Layer (Шар представлення)

MainForm.cs виступає точкою входу для користувацької взаємодії. Цей компонент відповідає за відображення інтерфейсу та делегує всі бізнес-

операції до шару бізнес-логіки, дотримуючись принципу єдиної відповідальності.

Business Layer (Бізнес-логіка)

Серце системи, що містить два ключові компоненти:

ArchiveManager.cs — центральний компонент системи, який координує всі операції з архівами. Він взаємодіє з усіма іншими шарами через інтерфейси, що забезпечує слабку зв'язаність. Менеджер використовує патерн Strategy через інтерфейс IArchiveStrategy для вибору алгоритмів роботи, патерн Visitor через IArchiveVisitor для виконання операцій над архівами, та звертається до бази даних через компонент Db.

ArchiveFactory.cs — реалізує патерн Factory для створення адаптерів архівів. Фабрика приховує деталі інстанціювання конкретних адаптерів та надає уніфікований інтерфейс для їх отримання.

Interfaces (Шар інтерфейсів)

Визначає контракти для взаємодії між компонентами:

IArchiveStrategy — визначає стратегії обробки архівів, дозволяючи динамічно змінювати алгоритми роботи.

IArchiveAdapter — забезпечує уніфікований інтерфейс для роботи з різними форматами архівів (ZIP, TAR), приховуючи специфіку кожного формату.

IArchiveVisitor — дозволяє додавати нові операції над архівами без зміни їх структури.

Adapter Layer (Шар адаптерів)

Реалізує патерн Adapter для підтримки різних форматів архівів:

ZipAdapter.cs — адаптує функціонал роботи з ZIP-архівами до загального інтерфейсу системи.

TarAdapter.cs — адаптує функціонал роботи з TAR-архівами до загального інтерфейсу системи.

Такий підхід дозволяє легко додавати підтримку нових форматів архівів без зміни існуючого коду.

Visitor Layer (Шар відвідувачів)

Реалізує патерн Visitor для виконання різноманітних операцій над архівами:

ChecksumVisitor.cs — обчислює контрольні суми файлів у архіві для перевірки цілісності даних.

TestVisitor.cs — виконує тестування архівів на предмет коректності структури та можливості розпакування.

Data Layer (Шар даних)

Db.cs — забезпечує доступ до бази даних для збереження метаінформації про архіви, історії операцій та налаштувань системи.

Розширення програмної частини системи

В ході виконання лабораторної роботи, система була розширена додатковими UI формами ArchiveHistoryForm та SettingsForm, що в сумі дає 3 візуальні форми. Також був реалізований повний цикл роботи з даними від вводу на формі до збереження їх в БД і подальшій виборці з БД (розширено Db.cs) та відображенням на UI, про які йшлося вище:

```
using ArchiverApp.Data;

namespace ArchiverApp.Forms
{
    2 usages  PriMerro23
    public partial class ArchiveHistoryForm : Form
    {
        private string _archivePath;

        1 usage  PriMerro23
        public ArchiveHistoryForm(string archivePath){...}

        1 usage  PriMerro23
        private void SetArchiveName(){...}

        2 usages  PriMerro23
        private void LoadOperationHistory(){...}

        1 usage  PriMerro23
        private string GetOperationDisplayName(string action){...}

        1 usage  PriMerro23
        private void btnRefresh_Click(object sender, EventArgs e){...}

        1 usage  PriMerro23
        private void btnClose_Click(object sender, EventArgs e){...}
    }
}
```

ArchiveHistoryForm.cs

```
namespace ArchiverApp.Forms
{
    2 usages  PriMerro23
    partial class ArchiveHistoryForm
    {
        /// Required designer variable. ...
        private System.ComponentModel.IContainer components = null;

        /// Clean up any resources being used. ...
        PriMerro23
        protected override void Dispose(bool disposing){...}

        #region Windows Form Designer generated code

        /// Required method for Designer support - do not modify ...
        1 usage  PriMerro23
        private void InitializeComponent()
        {
            this.lblArchiveName = new System.Windows.Forms.Label();
            this.lstOperations = new System.Windows.Forms.ListView();
            this.colDateTime = new System.Windows.Forms.ColumnHeader();
            this.colOperation = new System.Windows.Forms.ColumnHeader();
            this.colDetails = new System.Windows.Forms.ColumnHeader();
            this.btnRefresh = new System.Windows.Forms.Button();
            this.btnClose = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            // lblArchiveName
            //
            this.lblArchiveName.AutoSize = true;
            this.lblArchiveName.Font = new System.Drawing.Font( familyName: "Microsoft Sans Serif",
            this.lblArchiveName.Location = new System.Drawing.Point(12, 12);
            this.lblArchiveName.Name = "lblArchiveName";
```

```
this.lstOperations.Columns.AddRange(new System.Windows.Forms.ColumnHeader[] {
this.colDateTime,
this.colOperation,
this.colDetails});
this.lstOperations.FullRowSelect = true;
this.lstOperations.GridLines = true;
this.lstOperations.HideSelection = false;
this.lstOperations.Location = new System.Drawing.Point(12, 45);
this.lstOperations.Name = "lstOperations";
this.lstOperations.Size = new System.Drawing.Size( width: 560,  height: 280);
this.lstOperations.TabIndex = 1;
this.lstOperations.UseCompatibleStateImageBehavior = false;
this.lstOperations.View = System.Windows.Forms.View.Details;
//
// colDateTime
//
this.colDateTime.Text = "Дата/Час";
this.colDateTime.Width = 150;
//
// colOperation
//
this.colOperation.Text = "Операція";
this.colOperation.Width = 100;
//
// colDetails
//
this.colDetails.Text = "Деталі";
this.colDetails.Width = 300;
//
// btnRefresh
//
this.btnRefresh.Location = new System.Drawing.Point(400, 335);
this.btnRefresh.Name = "btnRefresh";
this.btnRefresh.Size = new System.Drawing.Size( width: 80,  height: 30);
this.btnRefresh.TabIndex = 2;
this.btnRefresh.Text = "Оновити";
this.btnRefresh.UseVisualStyleBackColor = true;
this.btnRefresh.Click += new System.EventHandler(this.btnRefresh_Click);
```

```
            //
            this.btnClose.DialogResult = System.Windows.Forms.DialogResult.OK;
            this.btnClose.Location = new System.Drawing.Point(490, 335);
            this.btnClose.Name = "btnClose";
            this.btnClose.Size = new System.Drawing.Size( width: 80,  height: 30);
            this.btnClose.TabIndex = 3;
            this.btnClose.Text = "Закрити";
            this.btnClose.UseVisualStyleBackColor = true;
            this.btnClose.Click += new System.EventHandler(this.btnClose_Click);
            //
            // ArchiveHistoryForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF( width: 6F,  height: 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size( width: 584,  height: 381);
            this.Controls.Add(this.btnClose);
            this.Controls.Add(this.btnRefresh);
            this.Controls.Add(this.lstOperations);
            this.Controls.Add(this.lblArchiveName);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "ArchiveHistoryForm";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
            this.Text = "Історія операцій архіву";
            this.ResumeLayout(false);
            this.PerformLayout();
        }

        #endregion

        private System.Windows.Forms.Label lblArchiveName;
        private System.Windows.Forms.ListView lstOperations;
        private System.Windows.Forms.ColumnHeader colDateTime;
        private System.Windows.Forms.ColumnHeader colOperation;
        private System.Windows.Forms.ColumnHeader colDetails;
        private System.Windows.Forms.Button btnRefresh;
        private System.Windows.Forms.Button btnClose;
```

ArchiveHistoryForm.Designer.cs

```
namespace ArchiverApp.Forms
{
    2 usages  PriMerro23
    public partial class SettingsForm : Form
    {
        1 usage  PriMerro23
        public SettingsForm(){...}

        1 usage  PriMerro23
        private void LoadSettings()
        {
            try{...}
            catch (Exception ex){...}
        }

        1 usage  PriMerro23
        private void btnSave_Click(object sender, EventArgs e)
        {
            try{...}
            catch (Exception ex)
            {
                MessageBox.Show( text: $"Помилка збереження налаштувань: {ex.Message}",  caption: "Помилка",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        3 usages  PriMerro23
        private void UpdateAppSetting(Configuration config, string key, string value)
        {
            if (config.AppSettings.Settings[key] != null){...}
            else
            {
                config.AppSettings.Settings.Add(key, value);
            }
        }

        1 usage  PriMerro23
        private void btnTestConnection_Click(object sender, EventArgs e){...}
```

SettingsForm.cs

```
partial class SettingsForm
{
    /// Required designer variable. ...
    private System.ComponentModel.IContainer components = null;

    /// Clean up any resources being used. ...
    PriMerro23
    protected override void Dispose(bool disposing){...}

    #region Windows Form Designer generated code

    /// Required method for Designer support - do not modify ...
    1 usage  PriMerro23
    private void InitializeComponent()
    {
        this.groupBoxDatabase = new System.Windows.Forms.GroupBox();
        this.btnTestConnection = new System.Windows.Forms.Button();
        this.txtConnectionString = new System.Windows.Forms.TextBox();
        this.lblConnection = new System.Windows.Forms.Label();
        this.groupBoxArchive = new System.Windows.Forms.GroupBox();
        this.lblCompressionInfo = new System.Windows.Forms.Label();
        this.numericSplitSize = new System.Windows.Forms.NumericUpDown();
        this.lblSplitSize = new System.Windows.Forms.Label();
        this.chkAutoBackup = new System.Windows.Forms.CheckBox();
        this.numericCompressionLevel = new System.Windows.Forms.NumericUpDown();
        this.lblCompression = new System.Windows.Forms.Label();
        this.btnSave = new System.Windows.Forms.Button();
        this.btnCancel = new System.Windows.Forms.Button();
        this.groupBoxDatabase.SuspendLayout();
        this.groupBoxArchive.SuspendLayout();
        ((System.ComponentModel.ISupportInitialize)(this.numericSplitSize)).BeginInit();
        ((System.ComponentModel.ISupportInitialize)(this.numericCompressionLevel)).BeginInit();
        this.SuspendLayout();
        //
        // groupBoxDatabase
        //
        this.groupBoxDatabase.Controls.Add(this.btnTestConnection);
        this.groupBoxDatabase.Controls.Add(this.txtConnectionString);
        this.groupBoxDatabase.Controls.Add(this.lblConnection);
        this.groupBoxDatabase.Location = new System.Drawing.Point(12, 12);
        this.groupBoxDatabase.Name = "groupBoxDatabase";
        this.groupBoxDatabase.Size = new System.Drawing.Size( width: 460,  height: 85);
```

```
this.groupBoxDatabase.Text = "База даних";
//
// btnTestConnection
//
this.btnTestConnection.Location = new System.Drawing.Point(370, 40);
this.btnTestConnection.Name = "btnTestConnection";
this.btnTestConnection.Size = new System.Drawing.Size( width: 75,  height: 23);
this.btnTestConnection.TabIndex = 2;
this.btnTestConnection.Text = "Тест";
this.btnTestConnection.UseVisualStyleBackColor = true;
this.btnTestConnection.Click += new System.EventHandler(this.btnTestConnection_Click);
//
// txtConnectionString
//
this.txtConnectionString.Location = new System.Drawing.Point(15, 42);
this.txtConnectionString.Name = "txtConnectionString";
this.txtConnectionString.Size = new System.Drawing.Size( width: 340,  height: 20);
this.txtConnectionString.TabIndex = 1;
//
// lblConnection
//
this.lblConnection.AutoSize = true;
this.lblConnection.Location = new System.Drawing.Point(12, 22);
this.lblConnection.Name = "lblConnection";
this.lblConnection.Size = new System.Drawing.Size( width: 161,  height: 13);
this.lblConnection.TabIndex = 0;
this.lblConnection.Text = "Рядок підключення до БД:";
//
// groupBoxArchive
//
this.groupBoxArchive.Controls.Add(this.lblCompressionInfo);
this.groupBoxArchive.Controls.Add(this.numericSplitSize);
this.groupBoxArchive.Controls.Add(this.lblSplitSize);
this.groupBoxArchive.Controls.Add(this.chkAutoBackup);
this.groupBoxArchive.Controls.Add(this.numericCompressionLevel);
this.groupBoxArchive.Controls.Add(this.lblCompression);
this.groupBoxArchive.Location = new System.Drawing.Point(12, 110);
this.groupBoxArchive.Name = "groupBoxArchive";
this.groupBoxArchive.Size = new System.Drawing.Size( width: 460,  height: 140);
this.groupBoxArchive.TabIndex = 1;
```

```
this.lblCompressionInfo.Text = "Рівень 6 (збалансоване)";
//
// numericSplitSize
//
this.numericSplitSize.Location = new System.Drawing.Point(15, 108);
this.numericSplitSize.Maximum = new decimal( bits: new int[] {
1000,
0,
0,
0});
this.numericSplitSize.Minimum = new decimal( bits: new int[] {
1,
0,
0,
0});
this.numericSplitSize.Name = "numericSplitSize";
this.numericSplitSize.Size = new System.Drawing.Size( width: 100,  height: 20);
this.numericSplitSize.TabIndex = 4;
this.numericSplitSize.Value = new decimal( bits: new int[] {
5,
0,
0,
0});
//
// lblSplitSize
//
this.lblSplitSize.AutoSize = true;
this.lblSplitSize.Location = new System.Drawing.Point(12, 90);
this.lblSplitSize.Name = "lblSplitSize";
this.lblSplitSize.Size = new System.Drawing.Size( width: 207,  height: 13);
this.lblSplitSize.TabIndex = 3;
this.lblSplitSize.Text = "Розмір частини при розділенні (МБ):";
//
// chkAutoBackup
//
this.chkAutoBackup.AutoSize = true;
this.chkAutoBackup.Location = new System.Drawing.Point(15, 68);
this.chkAutoBackup.Name = "chkAutoBackup";
this.chkAutoBackup.Size = new System.Drawing.Size( width: 207,  height: 17);
this.chkAutoBackup.TabIndex = 2;
```

```
private void InitializeComponent()
    this.numericCompressionLevel.Location = new System.Drawing.Point(15, 43);
    this.numericCompressionLevel.Maximum = new decimal( bits: new int[] {
    9,
    0,
    0,
    0});
    this.numericCompressionLevel.Name = "numericCompressionLevel";
    this.numericCompressionLevel.Size = new System.Drawing.Size( width: 100,  height: 20);
    this.numericCompressionLevel.TabIndex = 1;
    this.numericCompressionLevel.Value = new decimal( bits: new int[] {
    6,
    0,
    0,
    0});
    this.numericCompressionLevel.ValueChanged += new System.EventHandler(this.numericCompressionLevel_ValueChanged);
    //
    // lblCompression
    //
    this.lblCompression.AutoSize = true;
    this.lblCompression.Location = new System.Drawing.Point(12, 25);
    this.lblCompression.Name = "lblCompression";
    this.lblCompression.Size = new System.Drawing.Size( width: 225,  height: 13);
    this.lblCompression.TabIndex = 0;
    this.lblCompression.Text = "Рівень стиснення (за замовчуванням):";
    //
    // btnSave
    //
    this.btnSave.Location = new System.Drawing.Point(310, 270);
    this.btnSave.Name = "btnSave";
    this.btnSave.Size = new System.Drawing.Size( width: 80,  height: 30);
    this.btnSave.TabIndex = 2;
    this.btnSave.Text = "Зберегти";
    this.btnSave.UseVisualStyleBackColor = true;
    this.btnSave.Click += new System.EventHandler(this.btnSave_Click);
    //
    // btnCancel
    //
    this.btnCancel.DialogResult = System.Windows.Forms.DialogResult.Cancel;
    this.btnCancel.Location = new System.Drawing.Point(400, 270);
    this.btnCancel.Name = "btnCancel";
```

```
partial class SettingsForm
    private void InitializeComponent()
        this.AcceptButton = this.btnSave;
        this.AutoScaleDimensions = new System.Drawing.SizeF( width: 6F,  height: 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.CancelButton = this.btnCancel;
        this.ClientSize = new System.Drawing.Size( width: 494,  height: 312);
        this.Controls.Add(this.btnCancel);
        this.Controls.Add(this.btnSave);
        this.Controls.Add(this.groupBoxArchive);
        this.Controls.Add(this.groupBoxDatabase);
        this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;
        this.MaximizeBox = false;
        this.MinimizeBox = false;
        this.Name = "SettingsForm";
        this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
        this.Text = "Налаштування";
        this.groupBoxDatabase.ResumeLayout(false);
        this.groupBoxDatabase.PerformLayout();
        this.groupBoxArchive.ResumeLayout(false);
        this.groupBoxArchive.PerformLayout();
        ((System.ComponentModel.ISupportInitialize)(this.numericSplitSize)).EndInit();
        ((System.ComponentModel.ISupportInitialize)(this.numericCompressionLevel)).EndInit();
        this.ResumeLayout(false);
    }

    #endregion

    private System.Windows.Forms.GroupBox groupBoxDatabase;
    private System.Windows.Forms.Button btnTestConnection;
    private System.Windows.Forms.TextBox txtConnectionString;
    private System.Windows.Forms.Label lblConnection;
    private System.Windows.Forms.GroupBox groupBoxArchive;
    private System.Windows.Forms.Label lblCompressionInfo;
    private System.Windows.Forms.NumericUpDown numericSplitSize;
    private System.Windows.Forms.Label lblSplitSize;
    private System.Windows.Forms.CheckBox chkAutoBackup;
    private System.Windows.Forms.NumericUpDown numericCompressionLevel;
    private System.Windows.Forms.Label lblCompression;
    private System.Windows.Forms.Button btnSave;
    private System.Windows.Forms.Button btnCancel;
}
```

SettingsForm.Designer.cs

```
namespace ArchiverApp.Data
{
    3 usages  PriMerro23
    public class OperationRecord{...}
    12 usages  PriMerro23
    public static class Db
    {
        1 usage  PriMerro23
        public static void Initialize(){...}

        1 usage  PriMerro23
        public static void EnsureArchive(string path){...}

        3 usages  PriMerro23
        public static void SyncEntries(string path, string[] names){...}

        6 usages  PriMerro23
        public static void LogOperation(string path, string action, string details){...}

        1 usage  PriMerro23
        public static List<OperationRecord> GetOperationHistory(string archivePath){...}

        PriMerro23
        public static List<string> GetAllArchives(){...}


        6 usages  PriMerro23
        private static string GetConn() => ConfigurationManager.ConnectionStrings["ArchiverDb"].ConnectionString;
    }
}
```

Db.cs

Система тепер реалізує повний цикл:

1. Введення даних через UI форми (MainForm, SettingsForm)
2. Обробка через бізнес-логіку (ArchiveManager, Adapters)
3. Збереження в PostgreSQL базі даних (Db класи)
4. Вибірка та відображення в UI (ArchiveHistoryForm)

Висновок: В рамках лабораторної роботи, було успішно спроектовано діаграми розгортання та компонентів для системи, а також створено діаграми послідовностей для певних сценаріїв використання. Система була розширена додатковими UI формами та повністю реалізує архітектуру з повним циклом роботи з даними від введення до збереження в БД та відображення в інтерфейсі користувача.

Питання до лабораторної роботи

1. Що собою становить діаграма розгортання?

Діаграма розгортання представляє фізичне розташування системи, показуючи, на якому фізичному обладнанні запускається та чи інша складова програмного забезпечення. Головними елементами діаграми є вузли, пов'язані інформаційними шляхами.

2. Які бувають види вузлів на діаграмі розгортання?
   Вузли бувають двох типів:
   Пристрій (device) – це фізичне обладнання: комп'ютер або пристрій, пов'язаний із системою
   Середовище виконання (execution environment) – це програмне забезпечення, яке саме може включати інше програмне забезпечення, наприклад операційну систему або процес-контейнер (наприклад, вебсервер)

3. Які бувають зв’язки на діаграмі розгортання?
   Атрибути множинності (для показання, наприклад, підключення 2х і більше клієнтів до одного сервера)
   Назва – як правило, містить спосіб зв'язку між двома вузлами (назва протоколу, наприклад HTTP, IPC, або технологія взаємодії, наприклад .NET Remoting, WCF)
4. Які елементи присутні на діаграмі компонентів?
   Компоненти – окремі модулі системи (можуть бути представлені як логічні, фізичні файли .exe, .dll, або виконувані модулі)
   Залежності між компонентами – показують взаємозв'язки між модулями
   Блоки/групи компонентів – наприклад, серверні компоненти, клієнтські компоненти, middleware
5. Що становлять собою зв’язки на діаграмі компонентів?
   Залежності на діаграмі компонентів показують, що класи з одного компонента використовують класи з іншого компонента. Такі зв'язки допомагають зрозуміти, які компоненти повинні бути зібрані в інсталяційний пакет, а також показують, зміни в якому компоненті будуть впливати на інші компоненти.
6. Які бувають види діаграм взаємодії?
   Залежно від способу поділу на модулі розрізняють три види діаграм компонентів:
   Логічні – система уявляється як набір автономних модулів
   Фізичні – показують компоненти та залежності між ними (фізичні файли)
   Виконувані – кожен компонент являє собою деякий файл (виконувані файли, вихідні коди, HTML-сторінки тощо)

7. Для чого призначена діаграма послідовностей?
   Діаграма послідовностей призначена для моделювання взаємодії між об'єктами системи у певній послідовності часу. Вона відображає, як об'єкти

обмінюються повідомленнями, показуючи порядок і логіку виконання операцій. Діаграми є корисними для моделювання бізнес-процесів, проєктування архітектури систем і тестування.

8. Які ключові елементи можуть бути на діаграмі послідовностей?
   Актори (Actors) – користувачі чи інші системи, які взаємодіють із системою Об'єкти або класи – представлені прямокутниками з іменами
   Лінії життя – вертикальні пунктирні лінії, що представляють життєвий цикл об'єкта
   Повідомлення – лінії зі стрілками, які показують передачу повідомлень чи виклик методів (синхронні або асинхронні) Активності – прямокутники на лінії життя, що вказують періоди виконання дій
   Контрольні структури – блоки для відображення умов, циклів або альтернативних сценаріїв (alt, loop)
9. Як діаграми послідовностей пов’язані з діаграмами варіантів використання?
   Діаграми послідовностей зазвичай деталізують сценарії, описані в діаграмах варіантів використання, показуючи конкретну послідовність взаємодій для реалізації певного варіанту використання.
10. Як діаграми послідовностей пов’язані з діаграмами класів?
    Діаграми послідовностей використовують об'єкти класів, визначених на діаграмах класів, та показують, як ці об'єкти взаємодіють між собою через виклик методів, які також описані в діаграмах класів.