Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №4

з дисципліни
«Компоненти програмної інженерії. Якість та тестування програмного забезпечення»

на тему
«Провести інтеграційне тестування модульної системи»

Виконав:

студент групи ІП-93

Домінський Валентин Олексійович

номер залікової книжки: 9311

номер у списку: 9

Перевірив:

Бабарикін Ігор Владиславович

Київ 2021

Зміст

Мета:

Провести інтеграційне тестування модульної системи

Завдання:

N п/п	N_z mod 6	Input	Output
1	0, 3	IIG.PasswordHashingUtils (.dll) IIG.BinaryFlag (project)	File (.dll) Database (.dll + .cs + .bak)
2	1, 4	IIG.BinaryFlag (.dll) IIG.PasswordHashingUtils (project)	File (.dll) Database (.dll + .cs + .bak)
3	2	IIG.FileWorker (.dll) IIG.BinaryFlag (project)	Database (.dll + .cs + .bak)
4	5	IIG.FileWorker (.dll) IIG.PasswordHashingUtils (project)	Database (.dll + .cs + .bak)

Варіант = 9311 mod 6 = 5, отже Мій варіант:

4	5	IIG.FileWorker (.dll) IIG.PasswordHashingUtils (project)	Database (.dll + .cs + .bak)

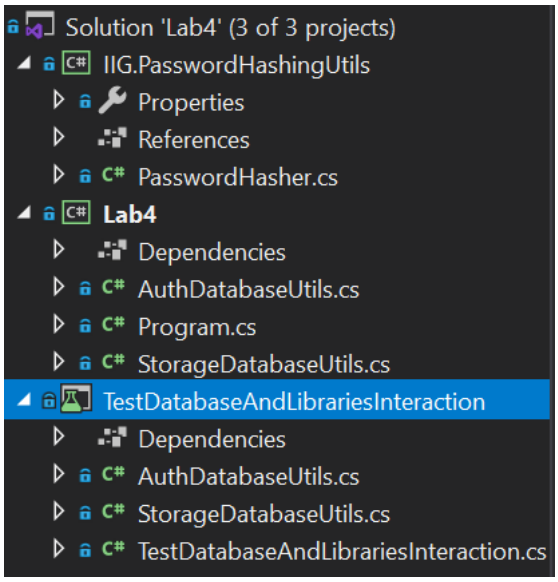
Отже, мій вибір для лабораторної:

- .NET 5
- Бібліотека для тестування xUnit
- Бібліотеки IIG.FileWorker (.dll) та IIG.PasswordHashingUtils (project)

Хід роботи:

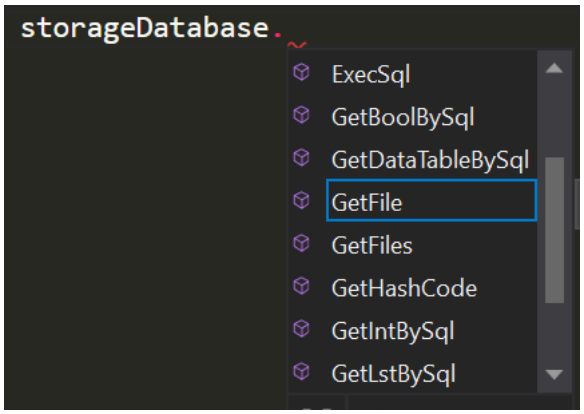
Початок роботи:

Я створив проект “Lab4” на .NET 5, додав xUnit Test Project “TestDatabaseAndLibrariesInteraction” та файли для роботи з базами даних

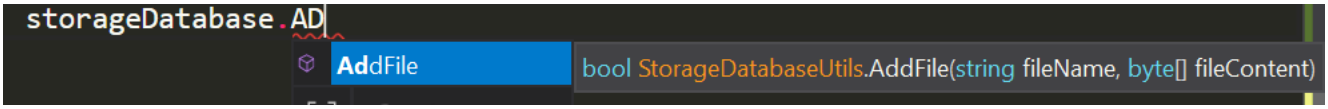


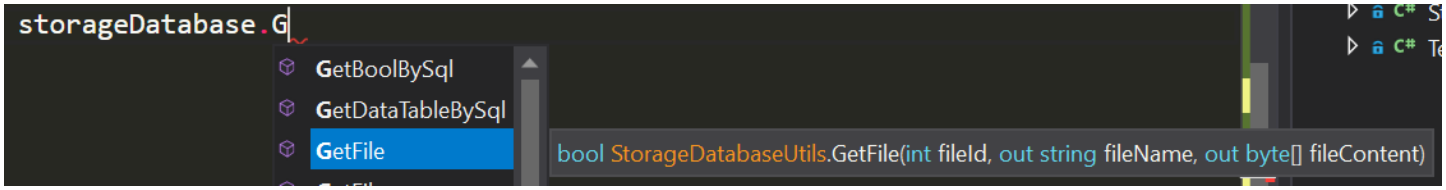
Для початку Нам треба обрати техніку для тестування. Найбільш підходящою, на Мій погляд, є Великий Вибух (“Big Bang” Integration) – всі, або практично всі, розроблені модулі збираються разом у вигляді закінченої системи або її основної частини, а після цього проводиться інтеграційне тестування.

Коли Ми підключили бази даних, можна глянути на методи *Назва Бази Даних*Utils, які пропонує Нам Visual Studio. Почати вирішив з бібліотеки FileWorker та БД IIG.CoSWE.StorageDB.



Оскільки Ми тестуємо лише ВЗАЄМОДІЮ, то Мій вибір пав на функції додавання файлів та отримання їх вмісту, тобто AddFile та GetFile



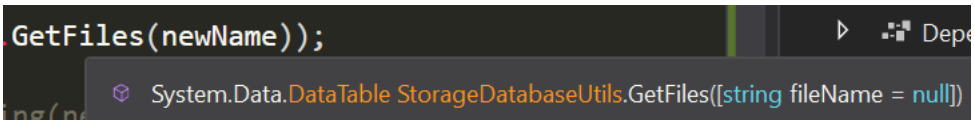


Тут Я почав експериментувати з першим методом, внаслідок чого в БД з'явилося багато файлів кожен з яких мав параметр, який Мною не задавався – FileID. І спочатку Я думав, що це доволі зручно, адже можна діставати вміст файлів за допомогою цього ідентифікатору. При подальших тестах, Я вирішив видалити всі дані з Бази Даних, щоб можна було звертатися до першого елементу і таким чином писати тести. На жаль, тут Я помітив одну проблему – FileID не прийняв значення за замовчанням. Навпаки – він ріс з кожним доданим файлом. У певний момент Моя БД мала такий вигляд:

	FileID	FileName	FileContent	UploadDateTime
1	21	SomeCoolName.txt	0x536F6D6520537472696E67	2021-05-15 21:24:12.330
2	22	SomeCoolName.txt	0x536F6D6520537472696E67	2021-05-15 21:24:12.330

Єдині два елементи мають значення не, наприклад, (0 та 1) або ж (1 та 2), а (21 й 22). Отже тепер, при кожному запуску тестів, Мені треба буде міняти значення field у тестах...

«Ну добре, але ж ще є метод GetFiles, який приймає лише назву».



Та, на відміну від GetFile, даний метод повертає не масив байтів, а DataTable.

На лекції Я дізнався, що можна дістати з БД максимальне значення параметру FileID, завдяки чому написання тестів продовжилося:

```
int? fileID = storageDatabase.GetIntBySql("SELECT MAX(FileID) FROM Files");
```

Тестування:

FileWorker

Тести, які стосуються FileWorker мають таку структуру:

- Задати значення, які будуть використовуватися.
- Додати файл до БД
- Отримати його FileID
- Взяти значення файлу з БД
- Перевірити їх ідентичність

Були перевірені такі кейси:

- Звичайні/повсякденні стрічки, які Ми зазвичай і зустрічаємо в якості назви файлу та його вмісту – значення повертаються успішно

```
/// <summary> Test FileWorker with regular values Should return same strings
[Fact]
| 0 references | Valentyn-Dominskyi, 3 hours ago | 1 author, 2 changes
public void FileWorker_RegularLetters_ReturnsSameStrings()
{
    // Arrange
    string expectedText = "Some Text";
    byte[] expectedTextInBytes = Encoding.UTF8.GetBytes(expectedText);

    string expectedName = "SomeCoolName.txt";

    // Act
    storageDatabase.AddFile(expectedName, expectedTextInBytes);

    int? fileID = storageDatabase.GetIntBySql
        ("SELECT MAX(FileID) FROM Files");

    storageDatabase.GetFile((int)fileID, out string actualName,
        out byte[] actualTextInBytes);

    string actualText = Encoding.UTF8.GetString(actualTextInBytes);

    storageDatabase.DeleteFile((int)fileID);

    // Assert
    Assert.Equal(actualName, expectedName);
    Assert.Equal(actualText, expectedText);
    Assert.Equal(actualTextInBytes, expectedTextInBytes);
}
```

- Порожній вміст файлу - значення повертаються успішно

```
[Fact]
| 0 references | Valentyn-Dominskyi, 3 hours ago | 1 author, 2 changes
public void FileWorker_EmptyText_ReturnsSameStrings()
{
    // Arrange
    string expectedText = "";
    byte[] expectedTextInBytes = Encoding.UTF8.GetBytes(expectedText);

    string expectedName = "SomeCoolName.txt";

    // Act
    storageDatabase.AddFile(expectedName, expectedTextInBytes);

    int? fileID = storageDatabase.GetIntBySql
        ("SELECT MAX(FileID) FROM Files");

    storageDatabase.GetFile((int)fileID, out string actualName,
        out byte[] actualTextInBytes);

    string actualText = Encoding.UTF8.GetString(actualTextInBytes);

    storageDatabase.DeleteFile((int)fileID);

    // Assert
    Assert.Equal(actualName, expectedName);
    Assert.Equal(actualText, expectedText);
    Assert.Equal(actualTextInBytes, expectedTextInBytes);
}
```

- Емодзі як назва/вміст - значення повертаються успішно

```
/// <summary> Test FileWorker with emojis Should return same strings
[Fact]
| 0 references | Valentyn-Dominskyi, 3 hours ago | 1 author, 3 changes
public void FileWorker_Emojis_ReturnsSameStrings()
{
    // Arrange
    string expectedText = "🎨🎨🎨";
    byte[] expectedTextInBytes = Encoding.UTF8.GetBytes(expectedText);

    string expectedName = "⚡⚡⚡.txt";

    // Act
    storageDatabase.AddFile(expectedName, expectedTextInBytes);

    int? fileID = storageDatabase.GetIntBySql("SELECT MAX(FileID) FROM Files");

    storageDatabase.GetFile((int)fileID, out string actualName,
        out byte[] actualTextInBytes);

    string actualText = Encoding.UTF8.GetString(actualTextInBytes);

    storageDatabase.DeleteFile((int)fileID);

    // Assert
    Assert.Equal(actualName, expectedName);
    Assert.Equal(actualText, expectedText);
    Assert.Equal(actualTextInBytes, expectedTextInBytes);
}
```

- Ієрогліфи як назва/вміст - значення повертаються успішно

```
[Fact]
| 0 references | Valentyn-Dominskyi, 3 hours ago | 1 author, 2 changes
public void FileWorker_Hieroglyphs_ReturnsSameStrings()
{
    // Arrange
    string expectedText = "汉字";
    byte[] expectedTextInBytes = Encoding.UTF8.GetBytes(expectedText);

    string expectedName = "漢字.txt";

    // Act
    storageDatabase.AddFile(expectedName, expectedTextInBytes);

    int? fileID = storageDatabase.GetIntBySql
        ("SELECT MAX(FileID) FROM Files");

    storageDatabase.GetFile((int)fileID, out string actualName,
        out byte[] actualTextInBytes);

    string actualText = Encoding.UTF8.GetString(actualTextInBytes);

    storageDatabase.DeleteFile((int)fileID);

    // Assert
    Assert.Equal(actualName, expectedName);
    Assert.Equal(actualText, expectedText);
    Assert.Equal(actualTextInBytes, expectedTextInBytes);
}
```

- Усі порожні стрічки - значення не повертаються успішно

```
[Fact]
| 0 references | Valentyn-Dominskyi, 3 hours ago | 1 author, 2 changes
public void FileWorker_EmptyStrings_ReturnsException()
{
    // Arrange
    string expectedText = "";
    byte[] expectedTextInBytes = Encoding.UTF8.GetBytes(expectedText);

    string expectedName = "";

    // Act
    storageDatabase.AddFile(expectedName, expectedTextInBytes);

    int? fileID = storageDatabase.GetIntBySql
        ("SELECT MAX(FileID) FROM Files");

    // Assert
    Assert.Throws<InvalidOperationException>(() =>
    storageDatabase.GetFile((int)fileID, out string actualName,
        out byte[] actualTextInBytes));
}
```

- Порожня назва - значення не повертаються успішно

```
[Fact]
| 0 references | Valentyn-Dominskyi, 3 hours ago | 1 author, 2 changes
public void FileWorker_EmptyName_ReturnsException()
{
    // Arrange
    string expectedText = "Some Text";
    byte[] expectedTextInBytes = Encoding.UTF8.GetBytes(expectedText);

    string expectedName = "";

    // Act
    storageDatabase.AddFile(expectedName, expectedTextInBytes);

    int? fileID = storageDatabase.GetIntBySql
        ("SELECT MAX(FileID) FROM Files");

    // Assert
    Assert.Throws<InvalidOperationException>(() =>
    storageDatabase.GetFile((int)fileID, out string actualName,
        out byte[] actualTextInBytes));
}
```

- Null назва - значення не повертаються успішно

```
[Fact]
| 0 references | Valentyn-Dominskyi, 3 hours ago | 1 author, 2 changes
public void FileWorker_NullName_ReturnsException()
{
    // Arrange
    string expectedText = "Some Text";
    byte[] expectedTextInBytes = Encoding.UTF8.GetBytes(expectedText);

    string expectedName = null;

    // Act
    storageDatabase.AddFile(expectedName, expectedTextInBytes);

    int? fileID = storageDatabase.GetIntBySql
        ("SELECT MAX(FileID) FROM Files");

    // Assert
    Assert.Throws<InvalidOperationException>(() =>
    storageDatabase.GetFile((int)fileID, out string actualName,
        out byte[] actualTextInBytes));
}
```

- Null масив байтів - значення не повертаються успішно

```
[Fact]
| 0 references | Valentyn-Dominskyi, 3 hours ago | 1 author, 2 changes
public void FileWorker_NullTextInBytes_ReturnsException()
{
    // Arrange
    byte[] expectedTextInBytes = null;

    string expectedName = "SomeCoolName.txt";

    // Act
    storageDatabase.AddFile(expectedName, expectedTextInBytes);

    int? fileID = storageDatabase.GetIntBySql
        ("SELECT MAX(FileID) FROM Files");

    // Assert
    Assert.Throws<InvalidOperationException>(() =>
    storageDatabase.GetFile((int)fileID, out string actualName,
        out byte[] actualTextInBytes));
}
```

PasswordHasher - AddCredentials

Тести, які стосуються PasswordHasher, а точніше AddCredentials мають таку структуру:

- Задати значення, які будуть використовуватися.
- Отримати хеш
- Додати файл до БД
- Перевірити їх ідентичність за допомогою CheckCredentials

Були перевірені такі кейси:

- Звичайні/повсякденні стрічки, які Ми зазвичай і зустрічаємо в якості логіну та паролю– значення збігаються

```
[Fact]
| 0 references | Valentyn-Dominskyi, 3 hours ago | 1 author, 1 change
public void PassHasher_AddCredentials_RegularLetters_ReturnsSameStrings()
{
    // Arrange
    string expectedLogin = "SomeCoolLogin";
    string expectedPassword = "SomeCoolPassword";

    // Act
    string expectedHashPassword = PasswordHasher.
        GetHash(expectedPassword);

    authDatabase.AddCredentials(expectedLogin, expectedHashPassword);

    bool areCredentialsTheSame = authDatabase.
        CheckCredentials(expectedLogin, expectedHashPassword);

    authDatabase.DeleteCredentials(expectedLogin, expectedHashPassword);

    // Assert
    Assert.True(areCredentialsTheSame);
}
```

- Порожній пароль - значення збігаються

```
[Fact]
| 0 references | Valentyn-Dominskyi, 3 hours ago | 1 author, 1 change
public void PassHasher_AddCredentials_EmptyPassword_ReturnsSameStrings()
{
    // Arrange
    string expectedLogin = "SomeCoolLogin";
    string expectedPassword = "";

    // Act
    string expectedHashPassword = PasswordHasher.
        GetHash(expectedPassword);

    authDatabase.AddCredentials(expectedLogin, expectedHashPassword);

    bool areCredentialsTheSame = authDatabase.
        CheckCredentials(expectedLogin, expectedHashPassword);

    authDatabase.DeleteCredentials(expectedLogin, expectedHashPassword);

    // Assert
    Assert.True(areCredentialsTheSame);
}
```

- Порожній пароль та одна літера в логіні - значення збігаються

```
| 0 references | Valentyn-Dominskyi, 3 hours ago | 1 author, 1 change
public void PassHasher_AddCredentials_EmptyPasswordAndOneLetterlogin_ReturnsSameStrings()
{
    // Arrange
    string expectedLogin = "S";
    string expectedPassword = "";

    // Act
    string expectedHashPassword = PasswordHasher.
        GetHash(expectedPassword);

    authDatabase.AddCredentials(expectedLogin, expectedHashPassword);

    bool areCredentialsTheSame = authDatabase.
        CheckCredentials(expectedLogin, expectedHashPassword);

    authDatabase.DeleteCredentials(expectedLogin, expectedHashPassword);

    // Assert
    Assert.True(areCredentialsTheSame);
}
```

- Емодзі як логін та пароль - значення збігаються

```
| 0 references | Valentyn-Dominskyi, 3 hours ago | 1 author, 1 change
public void PassHasher_AddCredentials_Emojis_ReturnsSameStrings()
{
    // Arrange
    string expectedLogin = "🍪🍪🍪";
    string expectedPassword = "⚡⚡⚡";

    // Act
    string expectedHashPassword = PasswordHasher.
        GetHash(expectedPassword);

    authDatabase.AddCredentials(expectedLogin, expectedHashPassword);

    bool areCredentialsTheSame = authDatabase.
        CheckCredentials(expectedLogin, expectedHashPassword);

    authDatabase.DeleteCredentials(expectedLogin, expectedHashPassword);

    // Assert
    Assert.True(areCredentialsTheSame);
}
```

- Ієрогліфи як логін та пароль значення збігаються

```
| 0 references | Valentyn-Dominskyi, 3 hours ago | 1 author, 1 change
public void PassHasher_AddCredentials_Hieroglyphs_ReturnsSameStrings()
{
    // Arrange
    string expectedLogin = "汉字";
    string expectedPassword = "漢字";

    // Act
    string expectedHashPassword = PasswordHasher.GetHash(expectedPassword);

    authDatabase.AddCredentials(expectedLogin, expectedHashPassword);

    bool areCredentialsTheSame = authDatabase.CheckCredentials(expectedLogin, expectedHashPassword);

    authDatabase.DeleteCredentials(expectedLogin, expectedHashPassword);

    // Assert
    Assert.True(areCredentialsTheSame);
}
```

PasswordHasher - UpdateCredentials

Тести, які стосуються PasswordHasher, а точніше UpdateCredentials мають таку структуру:

- Задати значення, які будуть використовуватися.
- Отримати хеш
- Додати файл до БД
- Оновити файл у БД
- Перевірити їх ідентичність за допомогою CheckCredentials

Були перевірені такі кейси:

- Звичайні/повсякденні стрічки, які Ми зазвичай і зустрічаємо в якості логіну та паролю– значення збігаються

```
public void PassHasher_UpdateCredentials_RegularLetters_ReturnsSameStrings()
{
    // Arrange
    string firstLogin = "SomeCoolLogin";
    string firstPassword = "SomeCoolPassword";

    string newLogin = "NewSomeCoolLogin";
    string newPassword = "NewSomeCoolPassword";

    // Act
    string firstHashPassword = PasswordHasher.
        GetHash(firstPassword);

    string newHashPassword = PasswordHasher.
        GetHash(newPassword);

    authDatabase.AddCredentials(firstLogin, firstHashPassword);

    authDatabase.UpdateCredentials(firstLogin, firstHashPassword,
        newLogin, newHashPassword);

    bool areCredentialsTheSame = authDatabase.
        CheckCredentials(newLogin, newHashPassword);

    authDatabase.DeleteCredentials(newLogin, newHashPassword);

    // Assert
    Assert.True(areCredentialsTheSame);
}
```

- Порожній пароль - значення збігаються

```
public void PassHasher_UpdateCredentials_EmptyPassword_ReturnsSameStrings()
{
    // Arrange
    string firstLogin = "SomeCoolLogin";
    string firstPassword = "";

    string newLogin = "NewSomeCoolLogin";
    string newPassword = "";

    // Act
    string firstHashPassword = PasswordHasher.
        GetHash(firstPassword);

    string newHashPassword = PasswordHasher.
        GetHash(newPassword);

    authDatabase.AddCredentials(firstLogin, firstHashPassword);

    authDatabase.UpdateCredentials(firstLogin, firstHashPassword,
        newLogin, newHashPassword);

    bool areCredentialsTheSame = authDatabase.
        CheckCredentials(newLogin, newHashPassword);

    authDatabase.DeleteCredentials(newLogin, newHashPassword);

    // Assert
    Assert.True(areCredentialsTheSame);
}
```

- Порожній пароль та одна літера в логіні - значення збігаються

```
| 0 references | Valentyn-Dominskyi, 45 minutes ago | 1 author, 1 change
public void PassHasher_UpdateCredentials_EmptyPasswordAndOneLetterLogin_ReturnsSameStrings()
{
    // Arrange
    string firstLogin = "S";
    string firstPassword = "";

    string newLogin = "N";
    string newPassword = "";

    // Act
    string firstHashPassword = PasswordHasher.
        GetHash(firstPassword);

    string newHashPassword = PasswordHasher.
        GetHash(newPassword);

    authDatabase.AddCredentials(firstLogin, firstHashPassword);

    authDatabase.UpdateCredentials(firstLogin, firstHashPassword,
        newLogin, newHashPassword);

    bool areCredentialsTheSame = authDatabase.
        CheckCredentials(newLogin, newHashPassword);

    authDatabase.DeleteCredentials(newLogin, newHashPassword);

    // Assert
    Assert.True(areCredentialsTheSame);
}
```

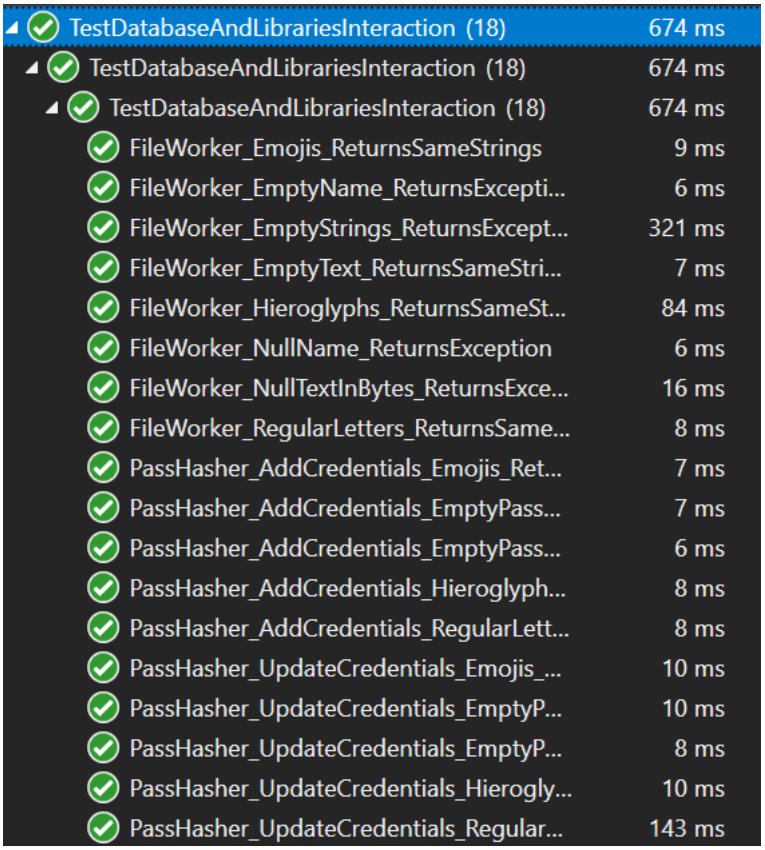
- Емодзі як логін та пароль - значення збігаються

```
public void PassHasher_UpdateCredentials_Emojis_ReturnsSameStrings()
{
    // Arrange
    string firstLogin = "🍪🍪🍪";
    string firstPassword = "⚡⚡⚡";

    string newLogin = "🔥🔥🔥";
    string newPassword = "🐛🐛🐛";

    // Act
    string firstHashPassword = PasswordHasher.
        GetHash(firstPassword);

    string newHashPassword = PasswordHasher.
        GetHash(newPassword);

    authDatabase.AddCredentials(firstLogin, firstHashPassword);

    authDatabase.UpdateCredentials(firstLogin, firstHashPassword,
        newLogin, newHashPassword);

    bool areCredentialsTheSame = authDatabase.
        CheckCredentials(newLogin, newHashPassword);

    authDatabase.DeleteCredentials(newLogin, newHashPassword);

    // Assert
    Assert.True(areCredentialsTheSame);
}
```

- Ієрогліфи як логін та пароль - значення збігаються

```
public void PassHasher_UpdateCredentials_Hieroglyphs_ReturnsSameStrings()
{
    // Arrange
    string firstLogin = "汉字";
    string firstPassword = "漢字";

    string newLogin = "한자";
    string newPassword = "漢字";

    // Act
    string firstHashPassword = PasswordHasher.
        GetHash(firstPassword);

    string newHashPassword = PasswordHasher.
        GetHash(newPassword);

    authDatabase.AddCredentials(firstLogin, firstHashPassword);

    authDatabase.UpdateCredentials(firstLogin, firstHashPassword,
        newLogin, newHashPassword);

    bool areCredentialsTheSame = authDatabase.
        CheckCredentials(newLogin, newHashPassword);

    authDatabase.DeleteCredentials(newLogin, newHashPassword);

    // Assert
    Assert.True(areCredentialsTheSame);
}
```

Результати тестування



Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
44	11,49%	339	88,51%

Сирцеві коди:

TestDatabaseAndLibrariesInteraction

```
using System;
using Xunit;
using IIG.CoSFE.DatabaseUtils;
using IIG.PasswordHashingUtils;
using System.Text;

namespace TestDatabaseAndLibrariesInteraction
{
  public class TestDatabaseAndLibrariesInteraction
  {
    private const string Server = @"VSIG-MACHINE";
    private const string AuthDatabase = @"IIG.CoSWE.AuthDB";
    private const string StorageDatabase = @"IIG.CoSWE.StorageDB";
    private const bool IsTrusted = true;
    private const string Login = @"coswe";
    private const string Password = @"L}EjpfCgru9X@GLj";
    private const int ConnectionTimeout = 75;

    static readonly StorageDatabaseUtils storageDatabase =
      new(Server, StorageDatabase, IsTrusted,
        Login, Password, ConnectionTimeout);

    static readonly AuthDatabaseUtils authDatabase =
      new(Server, AuthDatabase,IsTrusted,
        Login, Password, ConnectionTimeout);

    /*
    Naming:
    1. The name of the project being tested.
    (optional) 2. The name of the method being tested
    3. The scenario under which it's being tested.
    4. The expected behavior when the scenario is invoked.
    */

    #region Storage DB

    /// <summary>
    /// Test FileWorker with regular values
    /// Should return same strings
    /// </summary>
    [Fact]
    public void FileWorker_RegularLetters_ReturnsSameStrings()
    {
      // Arrange
      string expectedText = "Some Text";
      byte[] expectedTextInBytes = Encoding.UTF8.GetBytes(expectedText);

      string expectedName = "SomeCoolName.txt";

      // Act
      storageDatabase.AddFile(expectedName, expectedTextInBytes);

      int? fileID = storageDatabase.GetIntBySql
        ("SELECT MAX(FileID) FROM Files");

      storageDatabase.GetFile((int)fileID, out string actualName,
        out byte[] actualTextInBytes);
```

```
    string actualText = Encoding.UTF8.GetString(actualTextInBytes);

    storageDatabase.DeleteFile((int)fileID);

    // Assert
    Assert.Equal(actualName, expectedName);
    Assert.Equal(actualText, expectedText);
    Assert.Equal(actualTextInBytes, expectedTextInBytes);
}

/// <summary>
/// Test FileWorker with empty values
/// Should return same strings
/// </summary>
[Fact]
public void FileWorker_EmptyText_ReturnsSameStrings()
{
    // Arrange
    string expectedText = "";
    byte[] expectedTextInBytes = Encoding.UTF8.GetBytes(expectedText);

    string expectedName = "SomeCoolName.txt";

    // Act
    storageDatabase.AddFile(expectedName, expectedTextInBytes);

    int? fileID = storageDatabase.GetIntBySql
      ("SELECT MAX(FileID) FROM Files");

    storageDatabase.GetFile((int)fileID, out string actualName,
      out byte[] actualTextInBytes);

    string actualText = Encoding.UTF8.GetString(actualTextInBytes);

    storageDatabase.DeleteFile((int)fileID);

    // Assert
    Assert.Equal(actualName, expectedName);
    Assert.Equal(actualText, expectedText);
    Assert.Equal(actualTextInBytes, expectedTextInBytes);
}

/// <summary>
/// Test FileWorker with emojis
/// Should return same strings
/// </summary>
[Fact]
public void FileWorker_Emojis_ReturnsSameStrings()
{
    // Arrange
    string expectedText = "🌩🌩🌩";
    byte[] expectedTextInBytes = Encoding.UTF8.GetBytes(expectedText);

    string expectedName = "⚡⚡⚡.txt";

    // Act
    storageDatabase.AddFile(expectedName, expectedTextInBytes);

    int? fileID = storageDatabase.GetIntBySql("SELECT MAX(FileID) FROM Files");

    storageDatabase.GetFile((int)fileID, out string actualName,
```

```
        out byte[] actualTextInBytes);

    string actualText = Encoding.UTF8.GetString(actualTextInBytes);

    storageDatabase.DeleteFile((int)fileID);

    // Assert
    Assert.Equal(actualName, expectedName);
    Assert.Equal(actualText, expectedText);
    Assert.Equal(actualTextInBytes, expectedTextInBytes);
  }

  /// <summary>
  /// Test FileWorker with hieroglyphs
  /// Should return same strings
  /// </summary>
  [Fact]
  public void FileWorker_Hieroglyphs_ReturnsSameStrings()
  {
    // Arrange
    string expectedText = "汉字";
    byte[] expectedTextInBytes = Encoding.UTF8.GetBytes(expectedText);

    string expectedName = "漢字.txt";

    // Act
    storageDatabase.AddFile(expectedName, expectedTextInBytes);

    int? fileID = storageDatabase.GetIntBySql
      ("SELECT MAX(FileID) FROM Files");

    storageDatabase.GetFile((int)fileID, out string actualName,
      out byte[] actualTextInBytes);

    string actualText = Encoding.UTF8.GetString(actualTextInBytes);

    storageDatabase.DeleteFile((int)fileID);

    // Assert
    Assert.Equal(actualName, expectedName);
    Assert.Equal(actualText, expectedText);
    Assert.Equal(actualTextInBytes, expectedTextInBytes);
  }

  /// <summary>
  /// Test FileWorker with empty strings
  /// Should return exception
  /// </summary>
  [Fact]
  public void FileWorker_EmptyStrings_ReturnsException()
  {
    // Arrange
    string expectedText = "";
    byte[] expectedTextInBytes = Encoding.UTF8.GetBytes(expectedText);

    string expectedName = "";

    // Act
    storageDatabase.AddFile(expectedName, expectedTextInBytes);

    int? fileID = storageDatabase.GetIntBySql
      ("SELECT MAX(FileID) FROM Files");
```

```
    // Assert
    Assert.Throws<InvalidOperationException>(() =>
    storageDatabase.GetFile((int)fileID, out string actualName,
        out byte[] actualTextInBytes));
}

/// <summary>
/// Test FileWorker with empty name
/// Should return exception
/// </summary>
[Fact]
public void FileWorker_EmptyName_ReturnsException()
{
    // Arrange
    string expectedText = "Some Text";
    byte[] expectedTextInBytes = Encoding.UTF8.GetBytes(expectedText);

    string expectedName = "";

    // Act
    storageDatabase.AddFile(expectedName, expectedTextInBytes);

    int? fileID = storageDatabase.GetIntBySql
        ("SELECT MAX(FileID) FROM Files");

    // Assert
    Assert.Throws<InvalidOperationException>(() =>
    storageDatabase.GetFile((int)fileID, out string actualName,
        out byte[] actualTextInBytes));
}

/// <summary>
/// Test FileWorker with null name
/// Should return exception
/// </summary>
[Fact]
public void FileWorker_NullName_ReturnsException()
{
    // Arrange
    string expectedText = "Some Text";
    byte[] expectedTextInBytes = Encoding.UTF8.GetBytes(expectedText);

    string expectedName = null;

    // Act
    storageDatabase.AddFile(expectedName, expectedTextInBytes);

    int? fileID = storageDatabase.GetIntBySql
        ("SELECT MAX(FileID) FROM Files");

    // Assert
    Assert.Throws<InvalidOperationException>(() =>
    storageDatabase.GetFile((int)fileID, out string actualName,
        out byte[] actualTextInBytes));
}

/// <summary>
/// Test FileWorker with null bytes
/// Should return exception
/// </summary>
[Fact]
```

```
public void FileWorker_NullTextInBytes_ReturnsException()
{
  // Arrange
  byte[] expectedTextInBytes = null;

  string expectedName = "SomeCoolName.txt";

  // Act
  storageDatabase.AddFile(expectedName, expectedTextInBytes);

  int? fileID = storageDatabase.GetIntBySql
    ("SELECT MAX(FileID) FROM Files");

  // Assert
  Assert.Throws<InvalidOperationException>(() =>
  storageDatabase.GetFile((int)fileID, out string actualName,
    out byte[] actualTextInBytes));
}

#endregion Storage DB

#region Auth DB

#region AddCredentials

/// <summary>
/// Test PasswordHasher AddCredentials with regular values
/// Should return same strings
/// </summary>
[Fact]
public void PassHasher_AddCredentials_RegularLetters_ReturnsSameStrings()
{
  // Arrange
  string expectedLogin = "SomeCoolLogin";
  string expectedPassword = "SomeCoolPassword";

  // Act
  string expectedHashPassword = PasswordHasher.
    GetHash(expectedPassword);

  authDatabase.AddCredentials(expectedLogin, expectedHashPassword);

  bool areCredentialsTheSame = authDatabase.
    CheckCredentials(expectedLogin, expectedHashPassword);

  authDatabase.DeleteCredentials(expectedLogin, expectedHashPassword);

  // Assert
  Assert.True(areCredentialsTheSame);
}

/// <summary>
/// Test PasswordHasher AddCredentials with empty password
/// Should return same strings
/// </summary>
[Fact]
public void PassHasher_AddCredentials_EmptyPassword_ReturnsSameStrings()
{
  // Arrange
  string expectedLogin = "SomeCoolLogin";
  string expectedPassword = "";
```

```
    // Act
    string expectedHashPassword = PasswordHasher.
      GetHash(expectedPassword);

    authDatabase.AddCredentials(expectedLogin, expectedHashPassword);

    bool areCredentialsTheSame = authDatabase.
      CheckCredentials(expectedLogin, expectedHashPassword);

    authDatabase.DeleteCredentials(expectedLogin, expectedHashPassword);

    // Assert
    Assert.True(areCredentialsTheSame);
}

/// <summary>
/// Test PasswordHasher AddCredentials with empty password
/// and one letter in a login
/// Should return same strings
/// </summary>
[Fact]
public void PassHasher_AddCredentials_EmptyPasswordAndOneLetterlogin_ReturnsSameStrings()
{
    // Arrange
    string expectedLogin = "S";
    string expectedPassword = "";

    // Act
    string expectedHashPassword = PasswordHasher.
      GetHash(expectedPassword);

    authDatabase.AddCredentials(expectedLogin, expectedHashPassword);

    bool areCredentialsTheSame = authDatabase.
      CheckCredentials(expectedLogin, expectedHashPassword);

    authDatabase.DeleteCredentials(expectedLogin, expectedHashPassword);

    // Assert
    Assert.True(areCredentialsTheSame);
}

/// <summary>
/// Test PasswordHasher AddCredentials with emojis
/// Should return same strings
/// </summary>
[Fact]
public void PassHasher_AddCredentials_Emojis_ReturnsSameStrings()
{
    // Arrange
    string expectedLogin = "🥀🥀🥀";
    string expectedPassword = "⚡⚡⚡";

    // Act
    string expectedHashPassword = PasswordHasher.
      GetHash(expectedPassword);

    authDatabase.AddCredentials(expectedLogin, expectedHashPassword);

    bool areCredentialsTheSame = authDatabase.
      CheckCredentials(expectedLogin, expectedHashPassword);
```

```
        authDatabase.DeleteCredentials(expectedLogin, expectedHashPassword);

        // Assert
        Assert.True(areCredentialsTheSame);
    }

    /// <summary>
    /// Test PasswordHasher AddCredentials with hieroglyphs
    /// Should return same strings
    /// </summary>
    [Fact]
    public void PassHasher_AddCredentials_Hieroglyphs_ReturnsSameStrings()
    {
        // Arrange
        string expectedLogin = "汉字";
        string expectedPassword = "漢字";

        // Act
        string expectedHashPassword = PasswordHasher.GetHash(expectedPassword);

        authDatabase.AddCredentials(expectedLogin, expectedHashPassword);

        bool areCredentialsTheSame = authDatabase.CheckCredentials(expectedLogin,
expectedHashPassword);

        authDatabase.DeleteCredentials(expectedLogin, expectedHashPassword);

        // Assert
        Assert.True(areCredentialsTheSame);
    }

    #endregion AddCredentials

    #region UpdateCredentials

    /// <summary>
    /// Test PasswordHasher UpdateCredentials with regular values
    /// Should return same strings
    /// </summary>
    [Fact]
    public void PassHasher_UpdateCredentials_RegularLetters_ReturnsSameStrings()
    {
        // Arrange
        string firstLogin = "SomeCoolLogin";
        string firstPassword = "SomeCoolPassword";

        string newLogin = "NewSomeCoolLogin";
        string newPassword = "NewSomeCoolPassword";

        // Act
        string firstHashPassword = PasswordHasher.
            GetHash(firstPassword);

        string newHashPassword = PasswordHasher.
            GetHash(newPassword);

        authDatabase.AddCredentials(firstLogin, firstHashPassword);

        authDatabase.UpdateCredentials(firstLogin, firstHashPassword,
            newLogin, newHashPassword);

        bool areCredentialsTheSame = authDatabase.
```

```
      CheckCredentials(newLogin, newHashPassword);

    authDatabase.DeleteCredentials(newLogin, newHashPassword);

    // Assert
    Assert.True(areCredentialsTheSame);
}

/// <summary>
/// Test PasswordHasher UpdateCredentials with empty password
/// Should return same strings
/// </summary>
[Fact]
public void PassHasher_UpdateCredentials_EmptyPassword_ReturnsSameStrings()
{
    // Arrange
    string firstLogin = "SomeCoolLogin";
    string firstPassword = "";

    string newLogin = "NewSomeCoolLogin";
    string newPassword = "";

    // Act
    string firstHashPassword = PasswordHasher.
      GetHash(firstPassword);

    string newHashPassword = PasswordHasher.
      GetHash(newPassword);

    authDatabase.AddCredentials(firstLogin, firstHashPassword);

    authDatabase.UpdateCredentials(firstLogin, firstHashPassword,
      newLogin, newHashPassword);

    bool areCredentialsTheSame = authDatabase.
      CheckCredentials(newLogin, newHashPassword);

    authDatabase.DeleteCredentials(newLogin, newHashPassword);

    // Assert
    Assert.True(areCredentialsTheSame);
}

/// <summary>
/// Test PasswordHasher UpdateCredentials
/// with empty password and one letter login
/// Should return same strings
/// </summary>
[Fact]
public void PassHasher_UpdateCredentials_EmptyPasswordAndOneLetterLogin_ReturnsSameStrings()
{
    // Arrange
    string firstLogin = "S";
    string firstPassword = "";

    string newLogin = "N";
    string newPassword = "";

    // Act
    string firstHashPassword = PasswordHasher.
      GetHash(firstPassword);
```

```
    string newHashPassword = PasswordHasher.
      GetHash(newPassword);

    authDatabase.AddCredentials(firstLogin, firstHashPassword);

    authDatabase.UpdateCredentials(firstLogin, firstHashPassword,
      newLogin, newHashPassword);

    bool areCredentialsTheSame = authDatabase.
      CheckCredentials(newLogin, newHashPassword);

    authDatabase.DeleteCredentials(newLogin, newHashPassword);

    // Assert
    Assert.True(areCredentialsTheSame);
  }

  /// <summary>
  /// Test PasswordHasher UpdateCredentials with emojis
  /// Should return same strings
  /// </summary>
  [Fact]
  public void PassHasher_UpdateCredentials_Emojis_ReturnsSameStrings()
  {
    // Arrange
    string firstLogin = "🤣🤣🤣";
    string firstPassword = "⚡⚡⚡";

    string newLogin = "🔥🔥🔥";
    string newPassword = "🦵🦵🦵";

    // Act
    string firstHashPassword = PasswordHasher.
      GetHash(firstPassword);

    string newHashPassword = PasswordHasher.
      GetHash(newPassword);

    authDatabase.AddCredentials(firstLogin, firstHashPassword);

    authDatabase.UpdateCredentials(firstLogin, firstHashPassword,
      newLogin, newHashPassword);

    bool areCredentialsTheSame = authDatabase.
      CheckCredentials(newLogin, newHashPassword);

    authDatabase.DeleteCredentials(newLogin, newHashPassword);

    // Assert
    Assert.True(areCredentialsTheSame);
  }

  /// <summary>
  /// Test PasswordHasher UpdateCredentials with hieroglyphs
  /// Should return same strings
  /// </summary>
  [Fact]
  public void PassHasher_UpdateCredentials_Hieroglyphs_ReturnsSameStrings()
  {
    // Arrange
    string firstLogin = "汉字";
    string firstPassword = "漢字";
```

```
        string newLogin = "한자";
        string newPassword = "漢字";

        // Act
        string firstHashPassword = PasswordHasher.
          GetHash(firstPassword);

        string newHashPassword = PasswordHasher.
          GetHash(newPassword);

        authDatabase.AddCredentials(firstLogin, firstHashPassword);

        authDatabase.UpdateCredentials(firstLogin, firstHashPassword,
          newLogin, newHashPassword);

        bool areCredentialsTheSame = authDatabase.
          CheckCredentials(newLogin, newHashPassword);

        authDatabase.DeleteCredentials(newLogin, newHashPassword);

        // Assert
        Assert.True(areCredentialsTheSame);
      }

      #endregion UpdateCredentials

      #endregion Auth DB
    }
  }
```

Висновки:

Виконавши цю лабораторну роботу Я познайомився з **інтеграційним тестуванням** у цілому, а також використав таку техніку, як **“Big Bang”**- при тестуванні даним видом всі, або практично всі, розроблені модулі збираються разом у вигляді закінченої системи або її основної частини, а після цього проводиться інтеграційне тестування

Джерела:

- Github - https://github.com/VsIG-official/Components-Of-Software-Engineering
- TestDatabaseAndLibrariesInteraction- https://github.com/VsIG-official/Components-Of-Software-Engineering/blob/master/Labs/Lab4/TestDatabaseAndLibrariesInteraction/TestDatabaseAndLibrariesInteraction.cs
- Директорія 4-ої лабораторної роботи - https://github.com/VsIG-official/Components-Of-Software-Engineering/tree/master/Labs/Lab4
- Офіційна документація - https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices
- Лекція по темі Класифікація тестування - https://docs.google.com/presentation/d/17vnw-vsFCvkHBBKFdMV3Hkob7X8B3Je_/edit#slide=id.p17