**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:

Перевірив:

студент групи ІП-93

Бабарикін Ігор Владиславович

Домінський Валентин Олексійович

номер залікової книжки: 9311

номер у списку: 9

Київ 2021

Зміст

[Мета: 3](#_Toc73289391)

[Завдання: 3](#_Toc73289392)

[Хід роботи: 4](#_Toc73289393)

[Початок роботи: 4](#_Toc73289394)

[Тестування: 7](#_Toc73289395)

[FileWorker 7](#_Toc73289396)

[PasswordHasher - AddCredentials 11](#_Toc73289397)

[PasswordHasher - UpdateCredentials 14](#_Toc73289398)

[Результати тестування 17](#_Toc73289399)

[Сирцеві коди: 18](#_Toc73289400)

[TestDatabaseAndLibrariesInteraction 18](#_Toc73289401)

[Висновки: 28](#_Toc73289402)

[Джерела: 28](#_Toc73289403)

Мета:

Провести інтеграційне тестування модульної системи

Завдання:

|  |  |  |  |
| --- | --- | --- | --- |
| N п/п | Nz mod 6 | Input | Output |
| 1 | 0, 3 | [IIG.PasswordHashingUtils (.dll)](https://drive.google.com/drive/u/1/folders/19DgnFmAXtl_Q1qUgVP4xoucQzqLqDzEY)  [IIG.BinaryFlag (project)](https://github.com/IGORGOD/IIG.BinaryFlag) | File ([.dll](https://drive.google.com/drive/u/1/folders/12mKELhYUBLMVmqkItTxemrLNo0w3cQm8))  Database ([.dll](https://drive.google.com/drive/u/1/folders/1znKlrrJsmYBV2wbaFMBwGcqaZekgb6XG) + [.cs](https://drive.google.com/drive/u/1/folders/1D628B-fdROwQxis7muHQ3fQF64i0YEly) + [.bak](https://drive.google.com/drive/u/1/folders/1wRhiEFUiQSOsZ_v8jdD6_Q-ZlLGFPUWV)) |
| 2 | 1, 4 | [IIG.BinaryFlag (.dll)](https://drive.google.com/drive/u/1/folders/1HLsbKf8k27RNVzDhiJkskLX6ypXO5t2u)  [IIG.PasswordHashingUtils (project)](https://github.com/IGORGOD/IIG.PasswordHashingUtils) | File ([.dll](https://drive.google.com/drive/u/1/folders/12mKELhYUBLMVmqkItTxemrLNo0w3cQm8))  Database ([.dll](https://drive.google.com/drive/u/1/folders/1znKlrrJsmYBV2wbaFMBwGcqaZekgb6XG) + [.cs](https://drive.google.com/drive/u/1/folders/1D628B-fdROwQxis7muHQ3fQF64i0YEly) + [.bak](https://drive.google.com/drive/u/1/folders/1wRhiEFUiQSOsZ_v8jdD6_Q-ZlLGFPUWV)) |
| 3 | 2 | [IIG.FileWorker (.dll)](https://drive.google.com/drive/u/1/folders/12mKELhYUBLMVmqkItTxemrLNo0w3cQm8)  [IIG.BinaryFlag (project)](https://github.com/IGORGOD/IIG.BinaryFlag) | Database ([.dll](https://drive.google.com/drive/u/1/folders/1znKlrrJsmYBV2wbaFMBwGcqaZekgb6XG) + [.cs](https://drive.google.com/drive/u/1/folders/1D628B-fdROwQxis7muHQ3fQF64i0YEly) + [.bak](https://drive.google.com/drive/u/1/folders/1wRhiEFUiQSOsZ_v8jdD6_Q-ZlLGFPUWV)) |
| 4 | 5 | [IIG.FileWorker (.dll)](https://drive.google.com/drive/u/1/folders/12mKELhYUBLMVmqkItTxemrLNo0w3cQm8)  [IIG.PasswordHashingUtils (project)](https://github.com/IGORGOD/IIG.PasswordHashingUtils) | Database ([.dll](https://drive.google.com/drive/u/1/folders/1znKlrrJsmYBV2wbaFMBwGcqaZekgb6XG) + [.cs](https://drive.google.com/drive/u/1/folders/1D628B-fdROwQxis7muHQ3fQF64i0YEly) + [.bak](https://drive.google.com/drive/u/1/folders/1wRhiEFUiQSOsZ_v8jdD6_Q-ZlLGFPUWV)) |

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

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 5 | [IIG.FileWorker (.dll)](https://drive.google.com/drive/u/1/folders/12mKELhYUBLMVmqkItTxemrLNo0w3cQm8)  [IIG.PasswordHashingUtils (project)](https://github.com/IGORGOD/IIG.PasswordHashingUtils) | Database ([.dll](https://drive.google.com/drive/u/1/folders/1znKlrrJsmYBV2wbaFMBwGcqaZekgb6XG) + [.cs](https://drive.google.com/drive/u/1/folders/1D628B-fdROwQxis7muHQ3fQF64i0YEly) + [.bak](https://drive.google.com/drive/u/1/folders/1wRhiEFUiQSOsZ_v8jdD6_Q-ZlLGFPUWV)) |

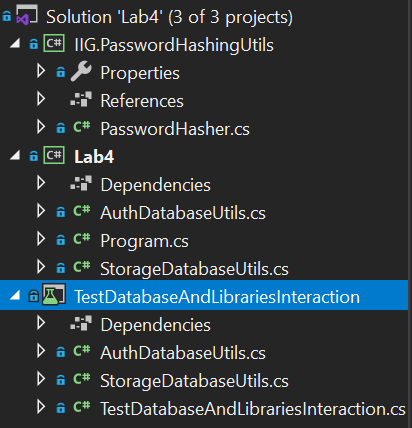
Отже, мій вибір для лабораторної:

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

Хід роботи:

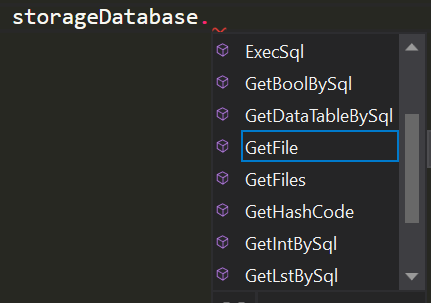
Початок роботи:

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

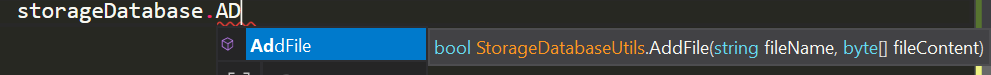


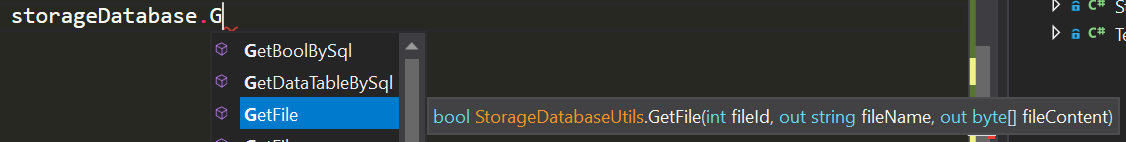
Для початку Нам треба обрати техніку для тестування. Найбільш підходящою, на Мій погляд, є Великий Вибух (“Big Bang” Integration) – всі, або практично всі, розроблені модулі збираються разом у вигляді закінченої системи або її основної частини, а після цього проводиться інтеграційне тестування.

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.

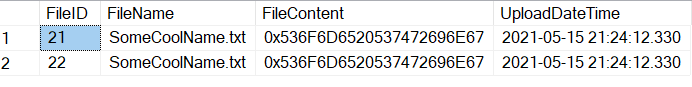


Оскільки Ми тестуємо лише ВЗАЄМОДІЮ, то Мій вибір пав на функції додавання файлів та отримання їх вмісту, тобто AddFile та GetFile



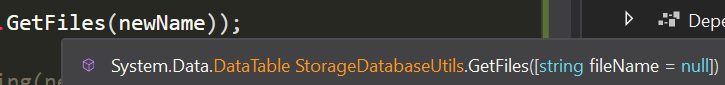


Тут Я почав експериментувати з першим методом, внаслідок чого в БД з’явилося багато файлів кожен з яких мав параметр, який Мною не задавався – FileID. І спочатку Я думав, що це доволі зручно, адже можна діставати вміст файлів за допомогою цього ідентифікатору. При подальших тестах, Я вирішив видалити 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:



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:



**Тестування**:

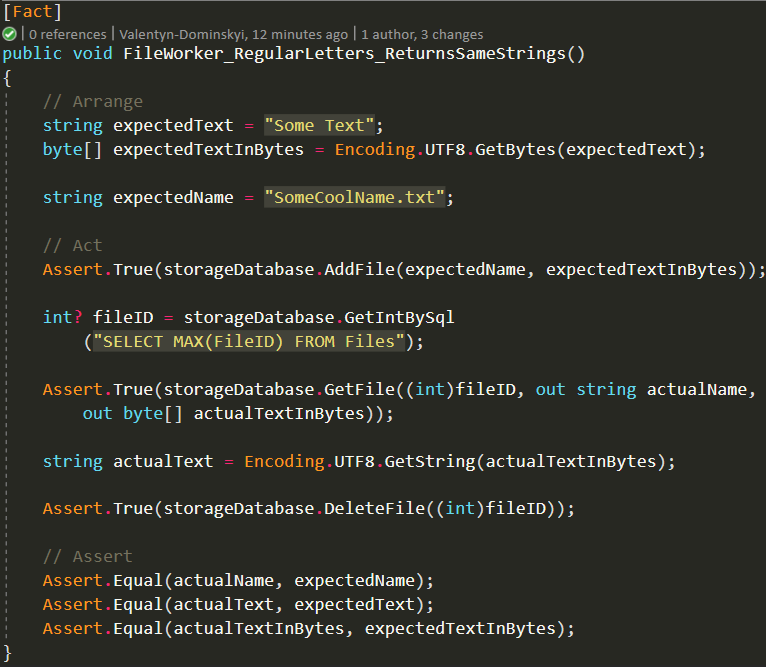
FileWorker

Тести, які стосуються FileWorker мають таку структуру:

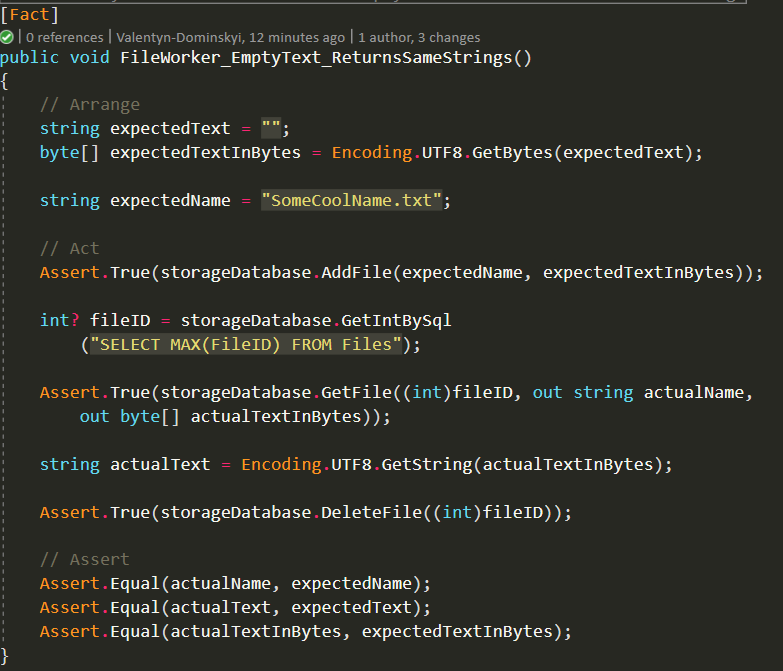
* Задати значення, які будуть використовуватися.
* Додати файл до БД
* Отримати його FileID
* Взяти значення файлу з БД
* Перевірити їх ідентичність

Були перевірені такі кейси:

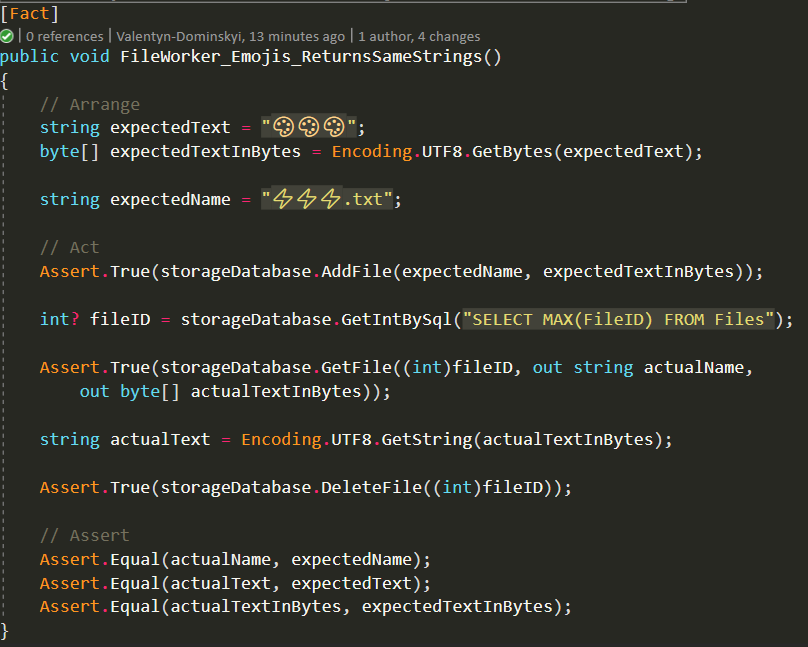
* Звичайні/повсякденні стрічки, які Ми зазвичай і зустрічаємо в якості назви файлу та його вмісту – значення повертаються успішно



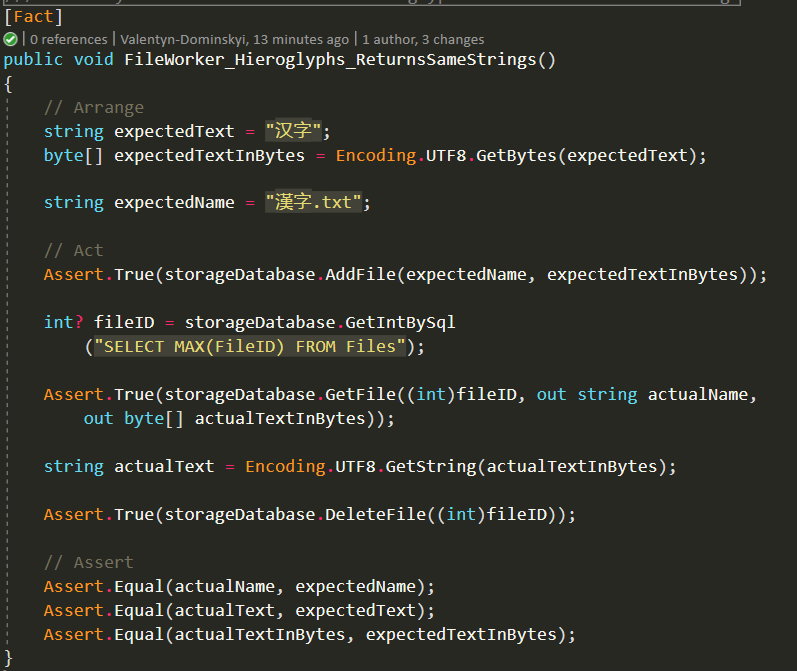
* Порожній вміст файлу - значення повертаються успішно



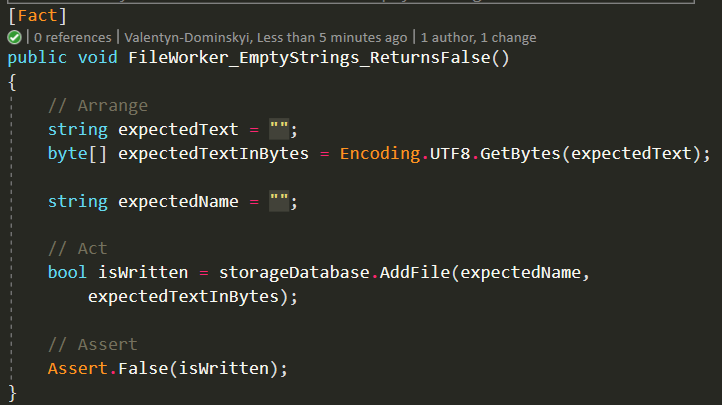
* Емодзі як назва/вміст - значення повертаються успішно



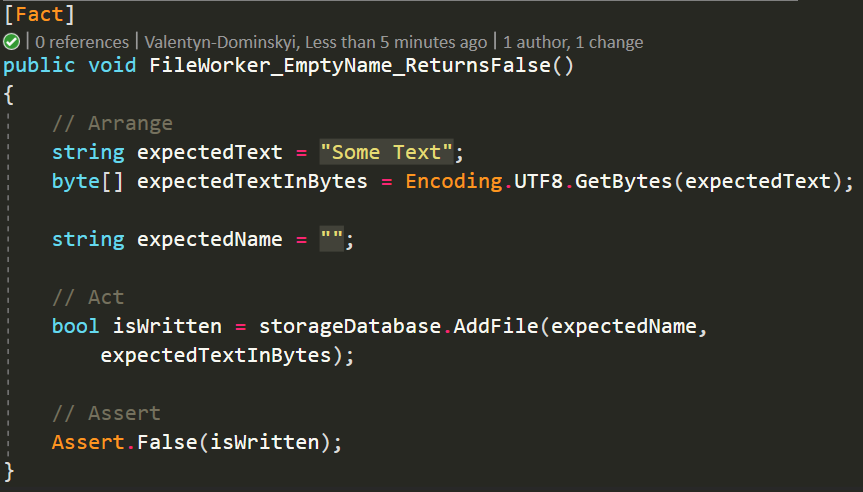
* Ієрогліфи як назва/вміст - значення повертаються успішно



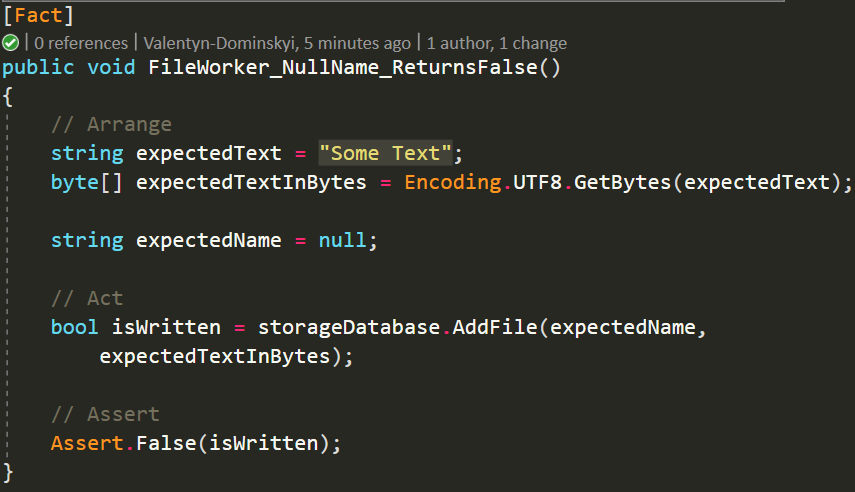
* Усі порожні стрічки - значення не повертаються успішно



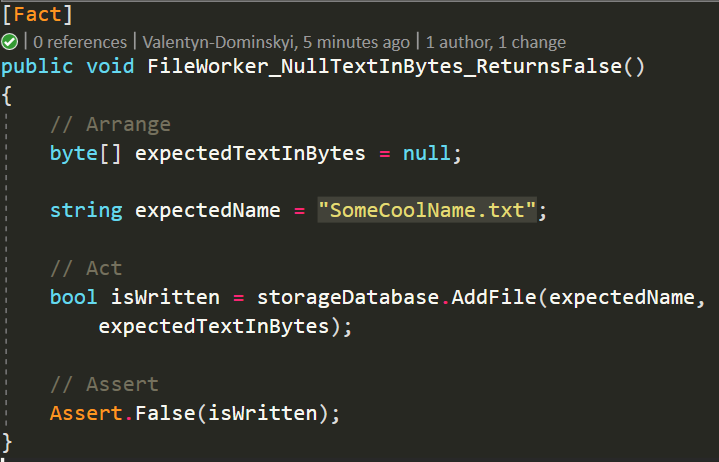
* Порожня назва - значення не повертаються успішно



* Null назва - значення не повертаються успішно



* Null масив байтів - значення не повертаються успішно



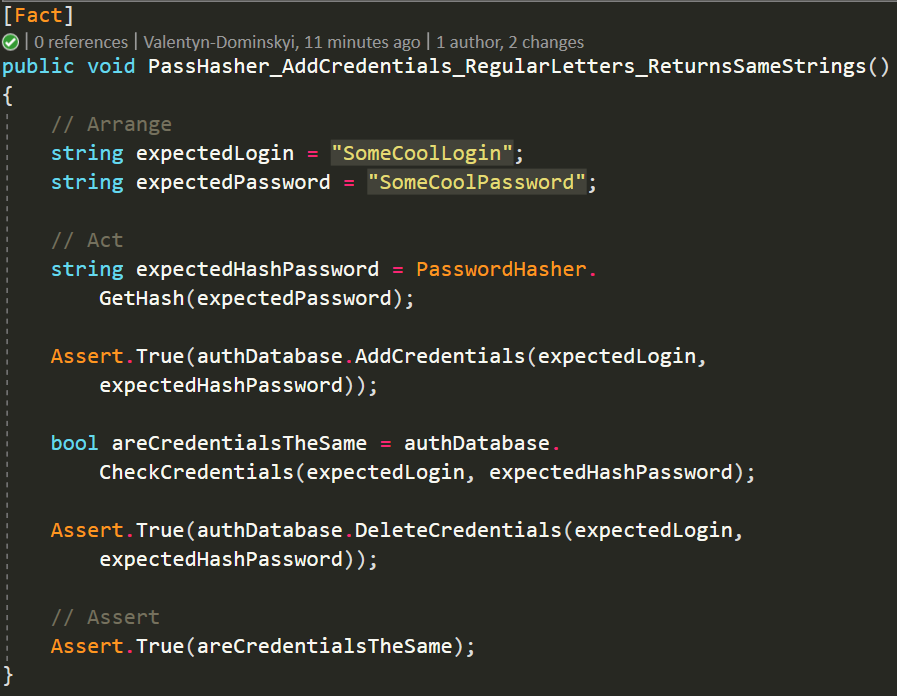
PasswordHasher - AddCredentials

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

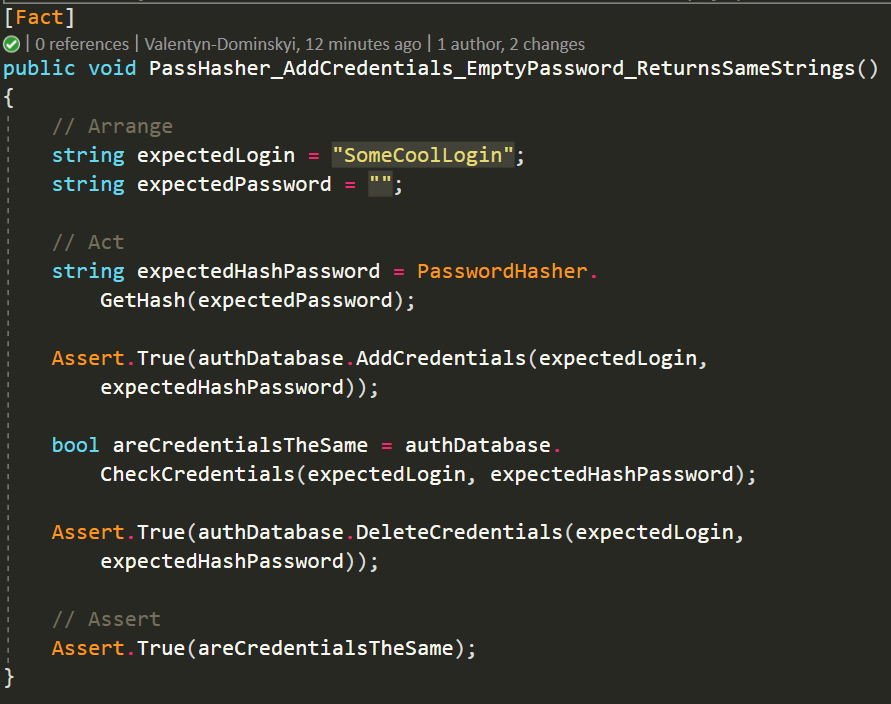
* Задати значення, які будуть використовуватися.
* Отримати хеш
* Додати файл до БД
* Перевірити їх ідентичність за допомогою CheckCredentials

Були перевірені такі кейси:

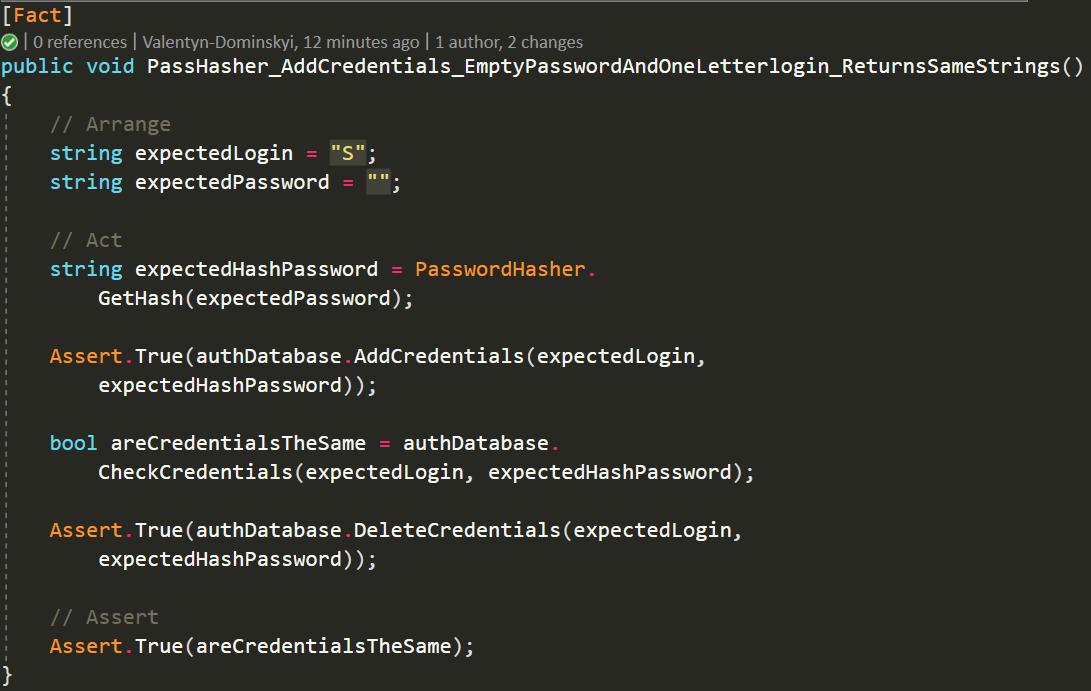
* Звичайні/повсякденні стрічки, які Ми зазвичай і зустрічаємо в якості логіну та паролю– значення збігаються



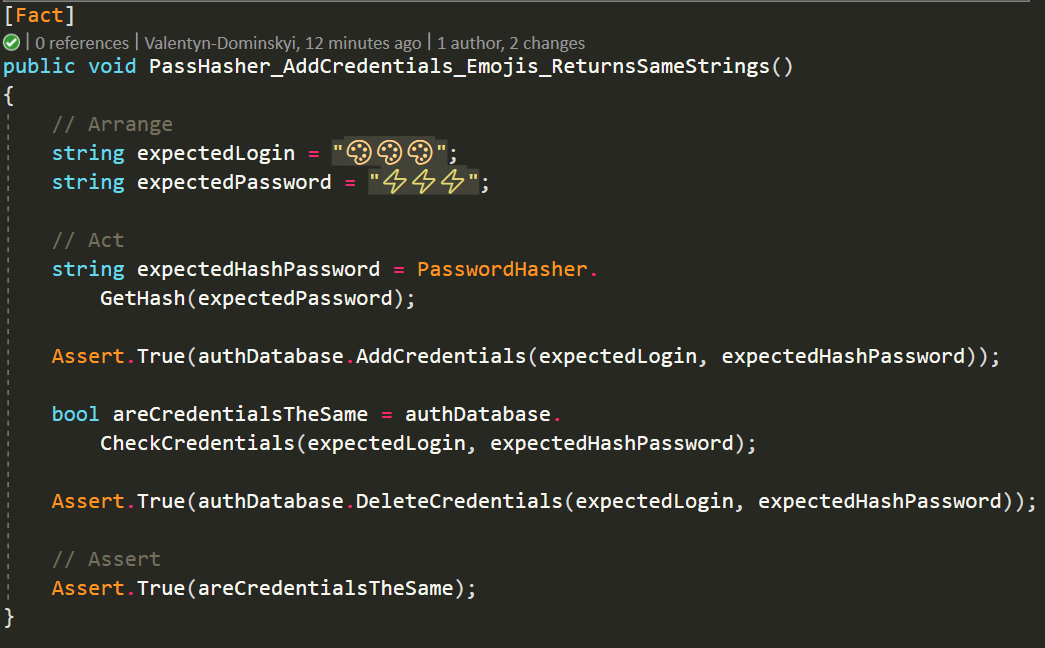
* Порожній пароль - значення збігаються



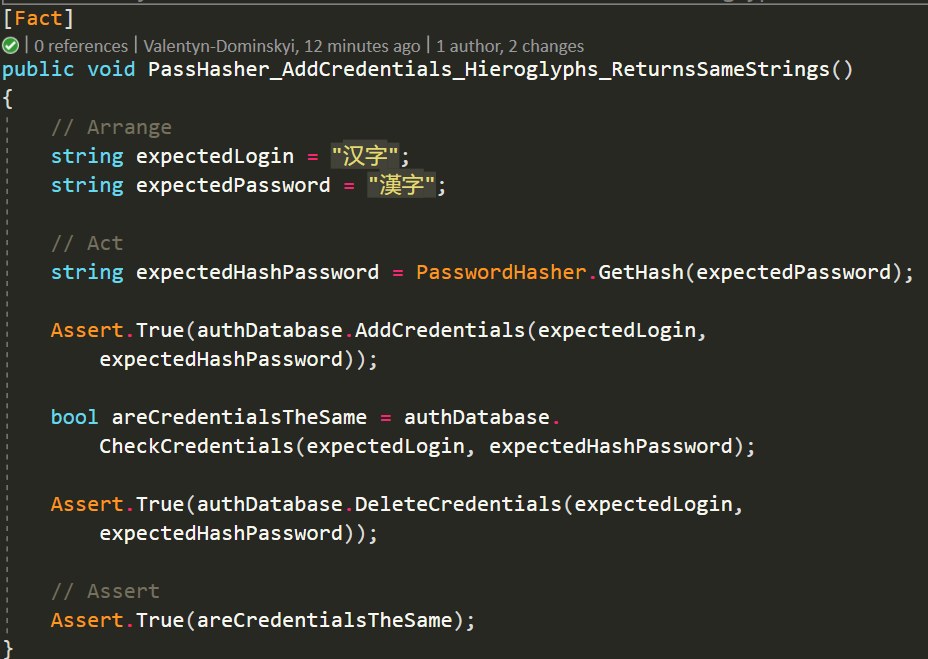
* Порожній пароль та одна літера в логіні - значення збігаються



* Емодзі як логін та пароль - значення збігаються



* Ієрогліфи як логін та пароль значення збігаються



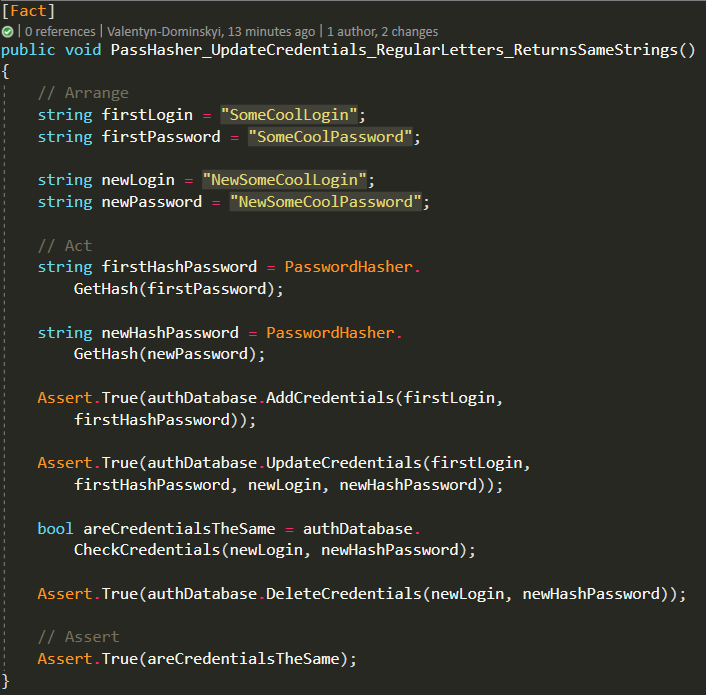
PasswordHasher - UpdateCredentials

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

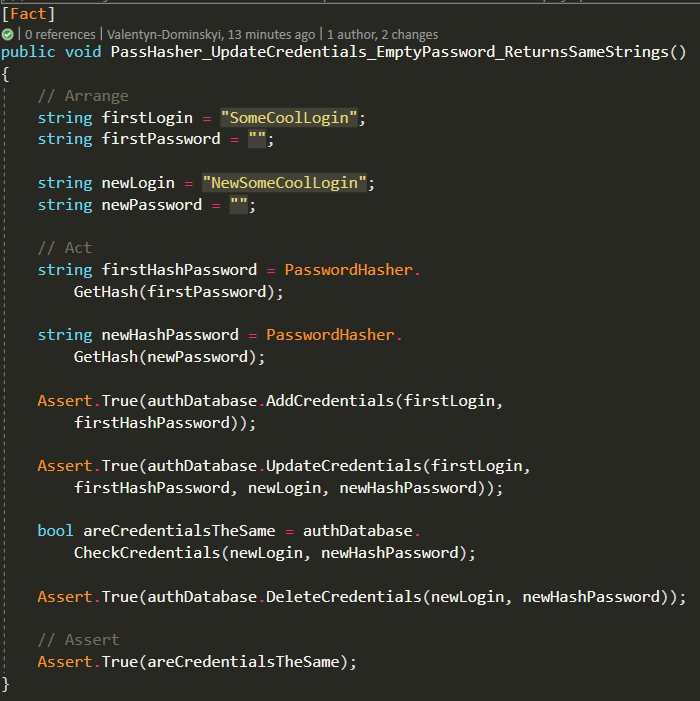
* Задати значення, які будуть використовуватися.
* Отримати хеш
* Додати файл до БД
* Оновити файл у БД
* Перевірити їх ідентичність за допомогою CheckCredentials

Були перевірені такі кейси:

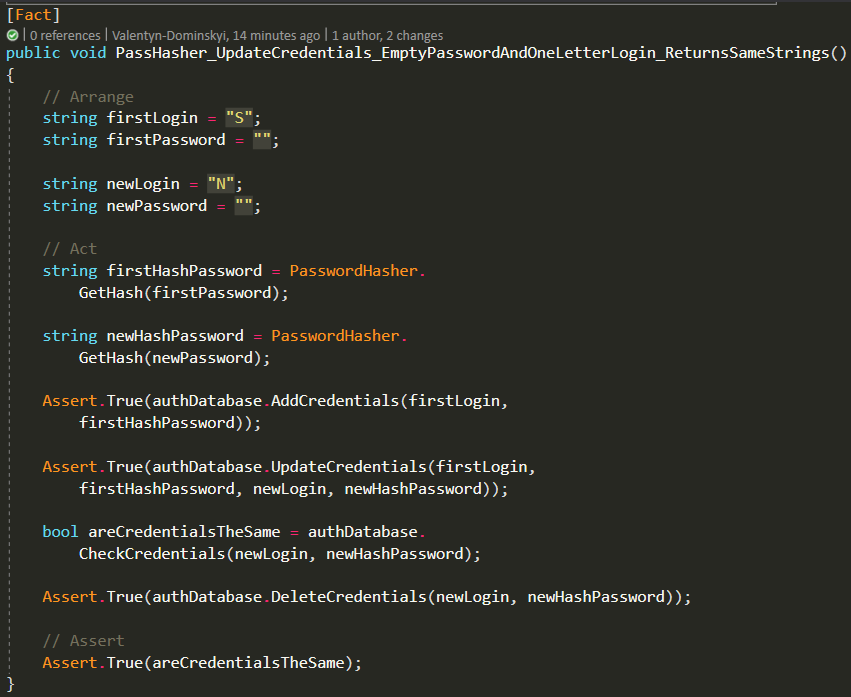
* Звичайні/повсякденні стрічки, які Ми зазвичай і зустрічаємо в якості логіну та паролю– значення збігаються



* Порожній пароль - значення збігаються



* Порожній пароль та одна літера в логіні - значення збігаються



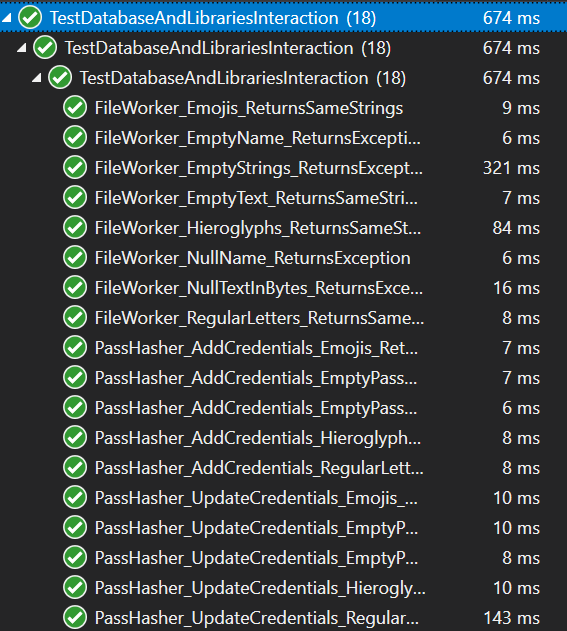
* Емодзі як логін та пароль - значення збігаються

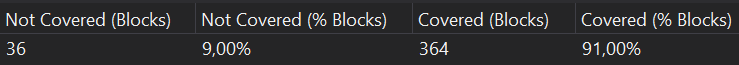


* Ієрогліфи як логін та пароль - значення збігаються



Результати тестування

****

****

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

Assert**.**True**(**storageDatabase**.**AddFile**(**expectedName**,** expectedTextInBytes**));**

int**?** fileID **=** storageDatabase**.**GetIntBySql

**(**"SELECT MAX(FileID) FROM Files"**);**

Assert**.**True**(**storageDatabase**.**GetFile**((**int**)**fileID**,** **out** string actualName**,**

**out** byte**[]** actualTextInBytes**));**

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

Assert**.**True**(**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

Assert**.**True**(**storageDatabase**.**AddFile**(**expectedName**,** expectedTextInBytes**));**

int**?** fileID **=** storageDatabase**.**GetIntBySql

**(**"SELECT MAX(FileID) FROM Files"**);**

Assert**.**True**(**storageDatabase**.**GetFile**((**int**)**fileID**,** **out** string actualName**,**

**out** byte**[]** actualTextInBytes**));**

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

Assert**.**True**(**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

Assert**.**True**(**storageDatabase**.**AddFile**(**expectedName**,** expectedTextInBytes**));**

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

Assert**.**True**(**storageDatabase**.**GetFile**((**int**)**fileID**,** **out** string actualName**,**

**out** byte**[]** actualTextInBytes**));**

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

Assert**.**True**(**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

Assert**.**True**(**storageDatabase**.**AddFile**(**expectedName**,** expectedTextInBytes**));**

int**?** fileID **=** storageDatabase**.**GetIntBySql

**(**"SELECT MAX(FileID) FROM Files"**);**

Assert**.**True**(**storageDatabase**.**GetFile**((**int**)**fileID**,** **out** string actualName**,**

**out** byte**[]** actualTextInBytes**));**

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

Assert**.**True**(**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 false

/// </summary>

**[**Fact**]**

**public** void FileWorker\_EmptyStrings\_ReturnsFalse**()**

**{**

// Arrange

string expectedText **=** ""**;**

byte**[]** expectedTextInBytes **=** Encoding**.**UTF8**.**GetBytes**(**expectedText**);**

string expectedName **=** ""**;**

// Act

bool isWritten **=** storageDatabase**.**AddFile**(**expectedName**,**

expectedTextInBytes**);**

// Assert

Assert**.**False**(**isWritten**);**

**}**

/// <summary>

/// Test FileWorker with empty name

/// Should return false

/// </summary>

**[**Fact**]**

**public** void FileWorker\_EmptyName\_ReturnsFalse**()**

**{**

// Arrange

string expectedText **=** "Some Text"**;**

byte**[]** expectedTextInBytes **=** Encoding**.**UTF8**.**GetBytes**(**expectedText**);**

string expectedName **=** ""**;**

// Act

bool isWritten **=** storageDatabase**.**AddFile**(**expectedName**,**

expectedTextInBytes**);**

// Assert

Assert**.**False**(**isWritten**);**

**}**

/// <summary>

/// Test FileWorker with null name

/// Should return false

/// </summary>

**[**Fact**]**

**public** void FileWorker\_NullName\_ReturnsFalse**()**

**{**

// Arrange

string expectedText **=** "Some Text"**;**

byte**[]** expectedTextInBytes **=** Encoding**.**UTF8**.**GetBytes**(**expectedText**);**

string expectedName **=** **null;**

// Act

bool isWritten **=** storageDatabase**.**AddFile**(**expectedName**,**

expectedTextInBytes**);**

// Assert

Assert**.**False**(**isWritten**);**

**}**

/// <summary>

/// Test FileWorker with null bytes

/// Should return false

/// </summary>

**[**Fact**]**

**public** void FileWorker\_NullTextInBytes\_ReturnsFalse**()**

**{**

// Arrange

byte**[]** expectedTextInBytes **=** **null;**

string expectedName **=** "SomeCoolName.txt"**;**

// Act

bool isWritten **=** storageDatabase**.**AddFile**(**expectedName**,**

expectedTextInBytes**);**

// Assert

Assert**.**False**(**isWritten**);**

**}**

#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**);**

Assert**.**True**(**authDatabase**.**AddCredentials**(**expectedLogin**,**

expectedHashPassword**));**

bool areCredentialsTheSame **=** authDatabase**.**

CheckCredentials**(**expectedLogin**,** expectedHashPassword**);**

Assert**.**True**(**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**);**

Assert**.**True**(**authDatabase**.**AddCredentials**(**expectedLogin**,**

expectedHashPassword**));**

bool areCredentialsTheSame **=** authDatabase**.**

CheckCredentials**(**expectedLogin**,** expectedHashPassword**);**

Assert**.**True**(**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**);**

Assert**.**True**(**authDatabase**.**AddCredentials**(**expectedLogin**,**

expectedHashPassword**));**

bool areCredentialsTheSame **=** authDatabase**.**

CheckCredentials**(**expectedLogin**,** expectedHashPassword**);**

Assert**.**True**(**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**);**

Assert**.**True**(**authDatabase**.**AddCredentials**(**expectedLogin**,** expectedHashPassword**));**

bool areCredentialsTheSame **=** authDatabase**.**

CheckCredentials**(**expectedLogin**,** expectedHashPassword**);**

Assert**.**True**(**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**);**

Assert**.**True**(**authDatabase**.**AddCredentials**(**expectedLogin**,**

expectedHashPassword**));**

bool areCredentialsTheSame **=** authDatabase**.**

CheckCredentials**(**expectedLogin**,** expectedHashPassword**);**

Assert**.**True**(**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**);**

Assert**.**True**(**authDatabase**.**AddCredentials**(**firstLogin**,**

firstHashPassword**));**

Assert**.**True**(**authDatabase**.**UpdateCredentials**(**firstLogin**,**

firstHashPassword**,** newLogin**,** newHashPassword**));**

bool areCredentialsTheSame **=** authDatabase**.**

CheckCredentials**(**newLogin**,** newHashPassword**);**

Assert**.**True**(**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**);**

Assert**.**True**(**authDatabase**.**AddCredentials**(**firstLogin**,**

firstHashPassword**));**

Assert**.**True**(**authDatabase**.**UpdateCredentials**(**firstLogin**,**

firstHashPassword**,** newLogin**,** newHashPassword**));**

bool areCredentialsTheSame **=** authDatabase**.**

CheckCredentials**(**newLogin**,** newHashPassword**);**

Assert**.**True**(**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**);**

Assert**.**True**(**authDatabase**.**AddCredentials**(**firstLogin**,**

firstHashPassword**));**

Assert**.**True**(**authDatabase**.**UpdateCredentials**(**firstLogin**,**

firstHashPassword**,** newLogin**,** newHashPassword**));**

bool areCredentialsTheSame **=** authDatabase**.**

CheckCredentials**(**newLogin**,** newHashPassword**);**

Assert**.**True**(**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**);**

Assert**.**True**(**authDatabase**.**AddCredentials**(**firstLogin**,**

firstHashPassword**));**

Assert**.**True**(**authDatabase**.**UpdateCredentials**(**firstLogin**,**

firstHashPassword**,** newLogin**,** newHashPassword**));**

bool areCredentialsTheSame **=** authDatabase**.**

CheckCredentials**(**newLogin**,** newHashPassword**);**

Assert**.**True**(**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**);**

Assert**.**True**(**authDatabase**.**AddCredentials**(**firstLogin**,**

firstHashPassword**));**

Assert**.**True**(**authDatabase**.**UpdateCredentials**(**firstLogin**,**

firstHashPassword**,** newLogin**,** newHashPassword**));**

bool areCredentialsTheSame **=** authDatabase**.**

CheckCredentials**(**newLogin**,** newHashPassword**);**

Assert**.**True**(**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>