МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра «Інформаційних систем та мереж»



**Звіт**

**про проходження практики**

Студента 4 курсу групи КН-47  
напряму підготовки Комп’ютерні науки

Шандра О.С.   
                                  (прізвище та ініціали)   
Керівник практики від університету:

к.т.н. асистент Худий А. М.   
   (посада, вчене звання, науковий ступінь, прізвище та ініціали)

### 

### Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Кількість балів: \_\_\_\_\_\_\_ Оцінка: ECTS \_\_\_\_\_\_\_

### Кількість балів: \_\_\_\_\_\_\_ Оцінка: ECTS \_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (прізвище та ініціали) |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (прізвище та ініціали) |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (прізвище та ініціали) |
|  | | |

Львів - 2018

**ЗМІСТ**

[Вступ……………………………………………………](#_Toc482457643)……………………………3

[1. ХАРАКТЕРИСТИКА ПІДПРИЄМСТВА 4](#_Toc482457644)

[2. ХАРАКТЕРИСТИКА ОФІСУ 4](#_Toc482457645)

[3. ПРАКТИКА НА РОБОЧОМУ МІСЦІ 10](#_Toc482457646)

[4. ВИКОНАННЯ ІНДИВІДУАЛЬНОГО ЗАВДАННЯ 1](#_Toc482457647)4

[ВИСНОВКИ 26](#_Toc482457653)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 2](#_Toc482457654)7

**ВСТУП**

Практика студентів є невід’ємною складовою освітньо-професійної програми підготовки студентів вищих навчальних закладів України. Вона становить важливу та обов’язкову ланку в підготовці висококваліфікованих спеціалістів до майбутньої діяльності за фахом. Спрямована на закріплення теоретичних знань, отриманих студентами за час навчання, та набуття, вдосконалення і розширення практичних навичок і умінь в роботі за обраною спеціальністю.

Виробнича практика була пройдена у підприємстві ФОП Музичука Олександра Анатолійовича.

Метою даної практики є:

* поглиблення та закріплення знань, які одержали студенти під час теоретичної підготовки;
* оволодіння сучасними формами та методами роботи з комплексом задач, розв’язуваних на підприємстві з використанням комп’ютерної техніки та інформаційних технологій;
* набуття необхідних навичок у здійсненні операцій технологічного процесу обробки інформації;
* виховання потреби систематичного оновлення своїх знань та їх творчого застосування у практичній діяльності.

Завданнями практики є вивчення організації і етапів розробки програмного продукту, набуття практичних навичок програмування, самостійного вирішення технічних задач на базі сучасних комп’ютеризованих систем, ознайомлення з сучасними технологічними процесами розробки, впровадження та налагодження програмного продукту, з сучасним апаратним та програмним забезпеченням, з організацією праці і економікою виробництва, набуття умінь організаторської роботи по спеціальності, узагальнення і поглиблення знань з вивчених дисциплін.

1. **ХАРАКТЕРИСТИКА ПІДПРИЄМСТВА**

Підприємство ФОП Музичука Олександра Анатолійовича, Abto Software — аутсорсингова компанія, що з 2007 року створює унікальні програмні рішення для лідерів у галузі Інтернету, електронного урядування, енергетики, медицини, будівництва, безпеки та ін.

Потужна наукова база, глибокі знання з математики, фізики і комп’ютерних наук та значний досвід роботи з великим спектром технологій дозволяє реалізовувати проекти найвищого рівня складності. А власна R&D-лабораторія дозволяє на повну реалізовувати свої креативні ідеї, експериментувати з технологіями і в команді однодумців творити проривні рішення.

**2. ХАРАКТЕРИСТИКА ОФІСУ**

Відповідно до тематичного плану і завдань практики необхідно описати наявні апаратні засобів і системне програмне забезпечення бази практики.

На місці проведення практики було виділено робоче місце та комп’ютер, який має наступні характеристики:

Системний блок – Core i7 2.7GHz/B150M-C/1\*16

Операційна система: Windows 10.

Монітор – LG LS24E65UPLX/CI (2 шт.)

На комп’ютерах встановлено Microsoft Visual Studio 2017, SourceTreeVisual Studio Code. Серед веб-браузерів використовуються Google Chrome, Mozilla Firefox, Microsoft Edge.

**Операційні системи:**

**Windows 10**

Операційна система (ОС) від компанії Microsoft для персональних комп’ютерів, ноутбуків, планшетів, лептопів-трансформерів і смартфонів. Ця операційна система є альтернативою попередниці Windows 8, для користувачів, звиклих до старого інтерфейсу. У компанії цю версію операційної системи називають останньою, позаяк надалі вона надаватиметься за моделлю «програмне забезпечення як послуга».

Microsoft представила попередню версію Windows 10 у Сан-Франциско 30 вересня 2014 року, очікувані нововведення підтвердились.

Реліз Windows 10 відбувся влітку 2015 року, а саме 29 липня в 190 країнах і 111-ма мовами. Windows 10 є останньою версією Windows. Протягом першого року після виходу системи користувачі мають змогу безкоштовно оновитися до Windows 10 на будь-якому пристрої під керуванням офіційних версій Windows 7.

**Офісні пакети:**

**Microsoft Office 2010**

Microsoft Office 2010 (кодове ім’я Office 14) – назва версії пакету Microsoft Office для Microsoft Windows. Microsoft почала роботу над Office 2010 в 2006 році, коли закінчувалася робота над пакетом Office 12 (який вийшов під назвою Microsoft Office 2007).

У пакет вбудована покращена анти піратська технологія Office Genuine Advantage, що дебютувала в квітні 2008 року. Всі продукти лінійки (крім серверних і OEM) вимагають активації. Офісний пакет, створений корпорацією Microsoft для операційних систем Microsoft Windows, Apple Мас OS X і Apple iOS (на iPad). До складу цього пакету входить програмне забезпечення для роботи з різними типами документів: текстами, електронними таблицями, презентаціями, базами даних тощо.

Microsoft Office також є сервером OLE об’єктів і його функції можуть використовуватися іншими застосунками, а також самими застосунками

Microsoft Office. Підтримує скрипти і макроси, написані на VBA.

**Антивірусне програмне забезпечення:**

**NOD32**

ESET NOD32 забезпечує захист від вірусів, а також від інших загроз, включаючи троянські програми, черв’яки, spywarе. В ESET NOD32 використовується патентована технологія ThreatSense, призначена для виявлення нових загроз, які виникають у реальному часі, шляхом аналізу виконуваних програм на наявність шкідливого коду, що дозволяє попереджати дії авторів шкідливих програм. Антивірусний пакет, що випускається словацькою фірмою ESET.

Перша версія була випущена в кінці 1987 року. NOD32 – це комплексне антивірусне розв’язання задачі захисту в реальному часі.

**Веб-браузери:**

**Google Chrome**

Розроблений компанією Google на основі веб-переглядача з відкритим кодом Chromium та іншого відкритого програмного забезпечення. Його програмна архітектура була створена з чистого аркуша (однак використовуючи інше програмне забезпечення з відкритим кодом, у тому числі компоненти WebKit та Mozilla) з метою задоволення поточних потреб користувачів.

**Internet Explorer**

Графічний веб-оглядач (браузер), який розробляє корпорація Microsoft.

Найперша версія побачила світ 16 серпня 1995 року. Вона була заснована на браузері Mosaic, права на який корпорація Microsoft придбала у компанії Spyglass. Починаючи з версії 3.0, Microsoft поширювала Internet Explorer у складі операційної системи Windows, що дозволило йому швидко збільшити ринкову частку і навіть витіснити провідний колись веб-оглядач Netscape Navigator.

Останніми версіями для Mac OS і Unix-подібних систем були Internet Explorer 5.2.3 і Internet Explorer 5.0 SP1 Beta відповідно. Зараз розробку Internet Explorer для цих систем припинено.

**Графічні редактори:**

**Paint**

Простий растровий графічний редактор компанії Microsoft, який входить до усіх операційних систем Windows, починаючи з перших версій.

У Windows 95 була введена нова версія Paint. Той же інтерфейс продовжує використовуватися в наступних версіях Windows.

Недоліки:

* Відсутня функція зміни яскравості(насиченості), контрасту тощо.
* Немає можливості при зміні розміру вказати бажаний розмір у пікселях (функція з’явилась в Windows 7).
* Неможливо зробити шрифт з правого краю, чи по центру.
* Неможливо зрушувати кордони області виділення (при цьому змінюються пропорції виділеного фрагменту).
* Неможливо повертати фігури на певну градусну міру.

**Adobe Photoshop**

Часто цю програму називають просто Photoshop. В даний час Photoshop доступний на платформах OS X, Windows, в мобільних системах iOS, Windows Phone і Android. Також існує версія Photoshop Express для Windows Phone 8 і 8.1. Зараз у СІЛА йде бета-тестування продукту для Chrome OS. Графічний редактор, розроблений і поширюваний фірмою Adobe Systems. Цей продукт є лідером ринку в області комерційних засобів редагування растрових зображень, і найвідомішим продуктом фірми Adobe.

**3. ПРАКТИКА НА РОБОЧОМУ МІСЦІ**

Практика проходила з 20 серпня по 2 вересня 2018 року. Основним завданням на практику було розроблення проекту «Книга рецептів».

На підприємстві проведення практики були створені належні умови для проходження виробничої практики. Я був ознайомлений з технікою безпеки на робочому місці та загальними правилами поведіки в офісі.

Впродовж проходження практики було набуто практичні навчки роботи по еджайлу.

Опис першочергових вимог

Програма призначена для створення книги рецептів:

• Зберігати список всіх продуктів, користувачів та їх особистих страв.

• Реєстрація/вхід в систему.

• Можливість редагувати інформацію про будь-який продукт.

• Користувач зможе виготовити особисті страви з продуктів і видаляти їх.

• Вся інформація повинна бути легкодоступною. Якщо користувачеві потрібно внести зміни, вони повинні мати можливість зробити це через простий у використанні інтерфейс настільного ПК.

Щоб краще зрозуміти вимоги програми, ми можемо показати загальну схему використання та опис вимог (рис. 1, 2). Є один актор який взаємодіє з системою. Це користувач системи, яка може бути кухаркою, домробітницею чи іншою людиною.

C:\Users\Oleg\Documents\UseCaseDiagram_f3kpln.png

Рис. 1. Use-case діаграма.

| **User should be able** | **Description** |
| --- | --- |
| Register | The ability to create a new account(in the entry window) with access to all products and a personal list of dishes that have been prepared from these products. |
| Login | If the user already has an account he should enter his personal login information and access the features of the program - he has access to all products and previously created dishes and can control them. If the user inputs incorrect login information, he will be informed about it and asked to enter the information again. |
| Manage products | User, successfully entered into the system, have access to all products and he is able to: remove products; add new products with their images that are stored in the cloud environment; update existing products; view product list in sorted and filtered form; select the needed products for making dishes. |
| Manage dishes | Opened a list of dishes, the user can remove those dishes that he previously added and search dishes by name. It should also be add that the main ability of this program is to create dishes based on the selected products using the "Create recipe" button in the list of all products page. This operation will add all the selected products in one dish and offer the user to enter the weight of each product in the dish. |

Рис. 2. Детальний опис можливостей системи (англійською).

Функціональні вимоги:

* Щоб користуватися функціоналом програми, користувачу необхідно зайти в особистий кабінет CRM системи або запустити додаток на смартфоні.

Нефункціональні вимоги

* Клієнтські ПК з ОС Windows 7 і вище.
* Мінімальні характеристики: процесор – intel Pentium, вільна пам’ять – 300 mb.
* Мова інтерфейсу – англійська.
* Зовнішні апаратні засоби: монітор, клавіатура, мишка.

**Якість**

Розроблена система має зручний і зрозумілий, інтерфейс що дозволить швидко пристосуватися до неї і комфортно користуватись без спеціальних навичок.

Реалізація

* Згідно з поставленими вимогами до етапу реалізації, було проведено розробку продукту. До основних результатів даного етапу належить:
* **Розширений документ, що описує вимоги.**

На даному етапі розробки реалізовані ті вимоги, які було поставлено.

* **Прототип системи.**

Протестований прототип, який показує зручність користування.

* **Дизайн системи**, найважливіший етап у створенні системи, оскільки саме він допоможе користувачам вирішувати свої проблеми швидко та зручно .
* **Код**, що складається з протестованих модулів.

**5. ТЕХНІЧНА ЧАСТИНА ВИКОНАННЯ ІНДИВІДУАЛЬНОГО ЗАВДАННЯ**

Для розроблення програми було використано мову програмування C# та технологію Windows Forms для розробки десктопних аплікацій під Widows. Сервером бази даних став MS SQL Server.

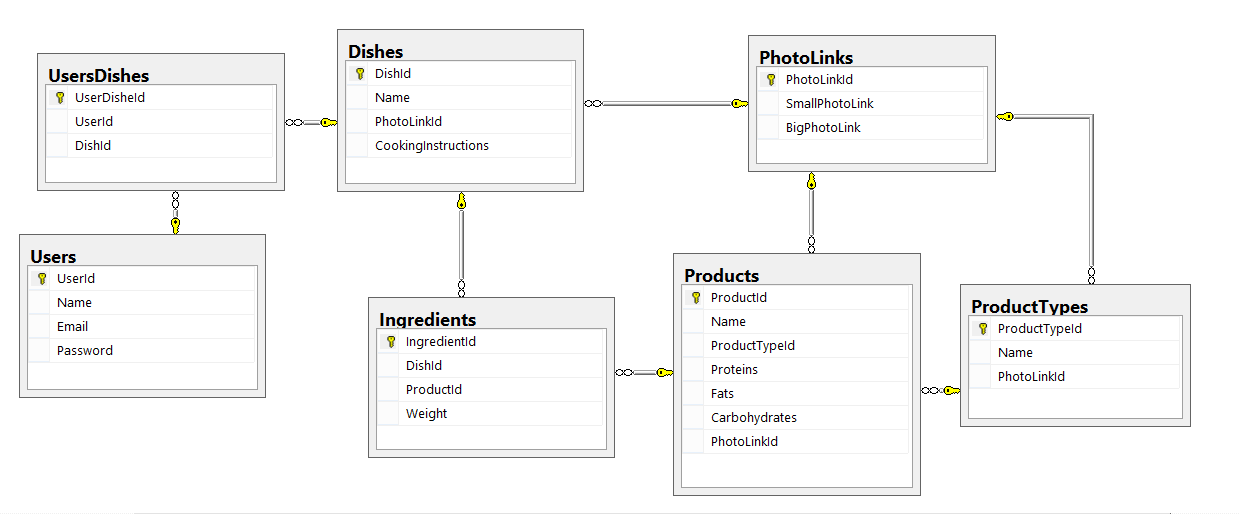


Рис. 3. Схема бази даних програми.

https://camo.githubusercontent.com/e62c475a61dab6271826291721245670d56a3e8e/687474703a2f2f7265732e636c6f7564696e6172792e636f6d2f647562626868786d722f696d6167652f75706c6f61642f76313437363632363134332f4d794469616772616d5f325f61697730656c2e706e67

Рис. 4. Схема класів програми.

У ході створення програми було реалізовано можливість створення нового облікового запису (рис. 5.) з доступом до всіх продуктів(рис. 6.) та особистим списком страв(рис. 7.), які були підготовлені з цих доданих продуктів (рис. 8).

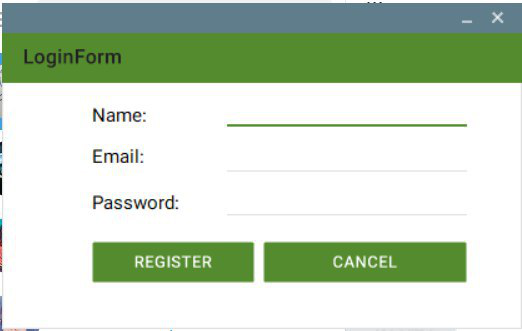


Рис. 5. Вікно реєстрації.

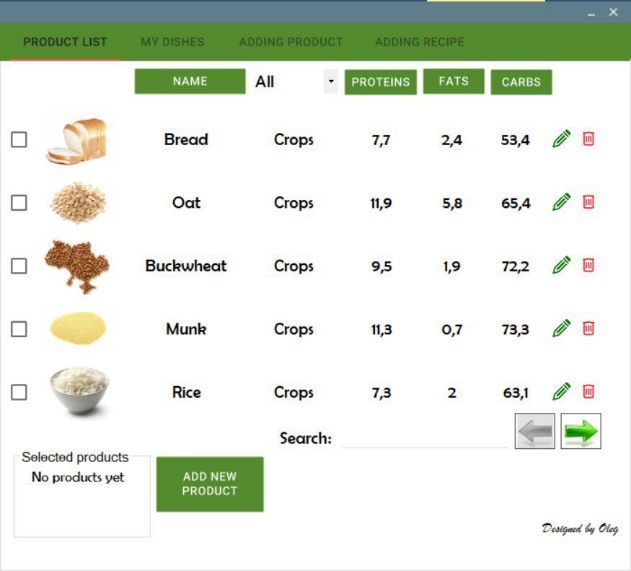


Рис. 6. Список продуктів.

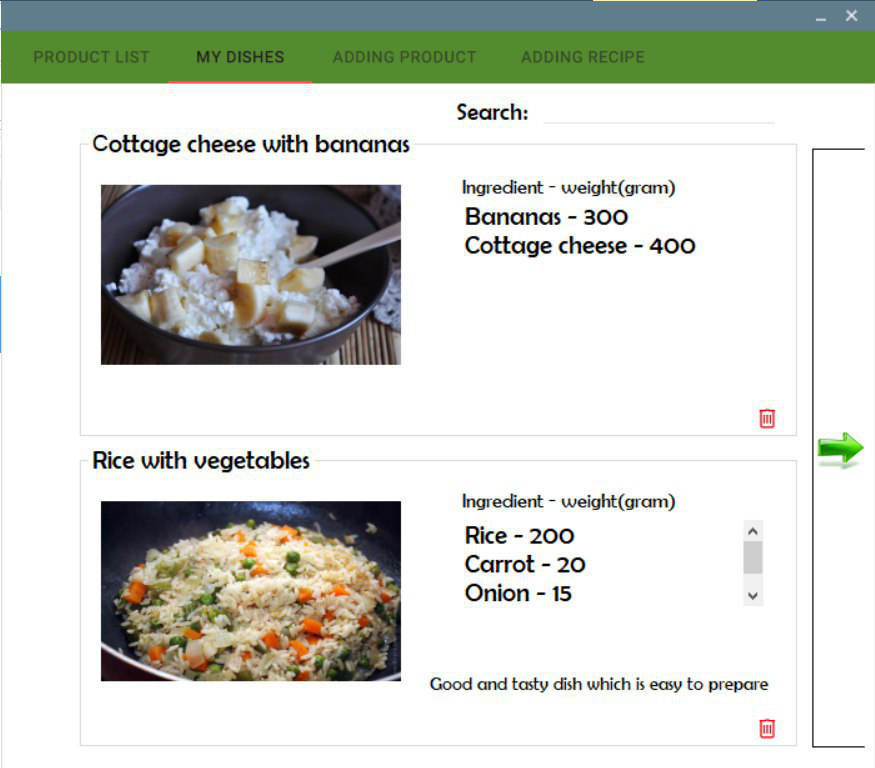


Рис. 7. Список страв.

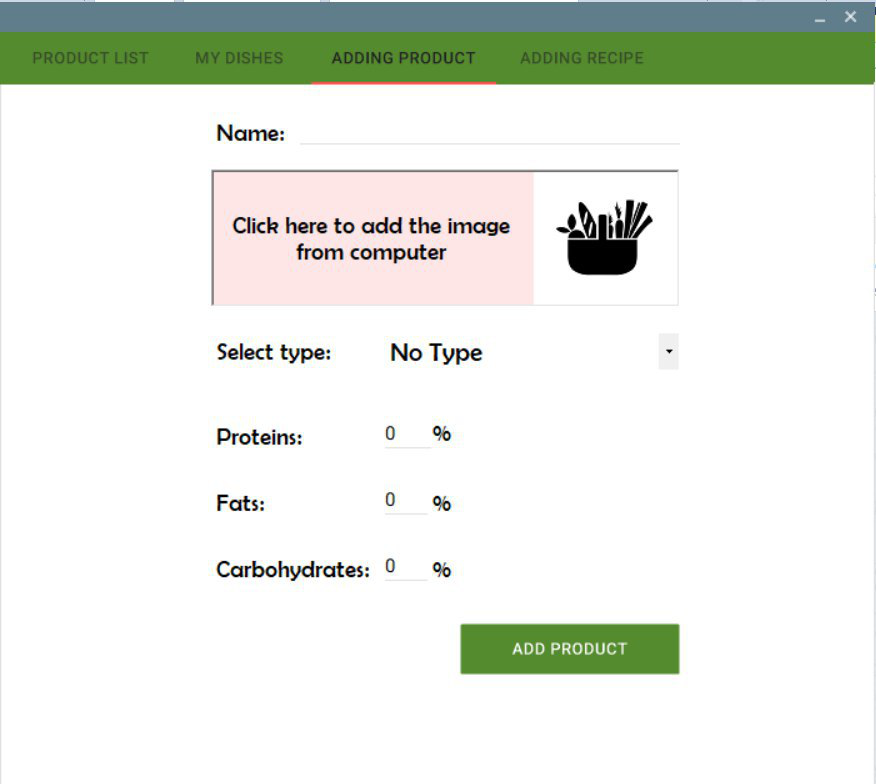


Рис. 8. Додавання продукту.

**ВИСНОВКИ**

Проходження практики надає професійну підготовку майбутнім спеціалістам, вчить самостійності, ініціативності, набуттю організаційних, дослідницьких навичок, які забезпечують повноцінну роботу випускників за фахом.

Протягом практики було здійснено ознайомлення з різними видами прикладного програмного забезпечення, вдосконалено свої навички. Також вивчено правила внутрішнього розпорядку підприємства; організаційну структуру управління підприємством; функції головних спеціалістів одного із відділів підприємства.

Окрім загальних завдань практики було виконано індивідуальне завдання та набуто практичні навички роботи у написанні сайтів.

Під час практики підібрано необхідну інформацію для придбання навичок практичної діяльності за фахом.

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Офіційна документація React JS. [Електронний ресурс]. Режим доступу URL: <https://reactjs.org/> .
2. Документація з використання ASP.NET Web API 2 (C#) [Електронний ресурс]. Режим доступу URL: https://docs.microsoft.com/en-us/aspnet/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api.
3. Офіційна документація Bootstrap. [Електронний ресурс]. Режим доступу URL: https://getbootstrap.com/.
4. Форум програмістів Електронний ресурс]. Режим доступу URL: https://stackoverflow.com/.
5. ГОСТ 34.201-90. Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы. Автоматизированные системы**.** Стадии создания. – М.: Изд-во стандартов, 1991. – 16 с.
6. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення. – К.: Держстандарт України, 1995. – 36 с.

Додаток А

Код серверної частини сайту

--Create new DB

CREATE DATABASE [RecipeBookSystemTestDB];

GO

USE [RecipeBookSystemTestDB];

GO

----------------------------------------------------------

CREATE TABLE Users

(

UserId INT NOT NULL PRIMARY KEY IDENTITY(1, 1),

Name NVARCHAR(100) NOT NULL,

Email NVARCHAR(200) NOT NULL UNIQUE,

[Password] NVARCHAR(200) NOT NULL

)

GO

CREATE TABLE PhotoLinks

(

PhotoLinkId INT NOT NULL PRIMARY KEY IDENTITY(1, 1),

SmallPhotoLink NVARCHAR(MAX) NULL,

BigPhotoLink NVARCHAR(MAX) NULL,

)

GO

CREATE TABLE ProductTypes

(

ProductTypeId INT NOT NULL PRIMARY KEY IDENTITY(1, 1),

Name NVARCHAR(100) NOT NULL UNIQUE,

PhotoLinkId INT NULL FOREIGN KEY REFERENCES PhotoLinks(PhotoLinkId)

)

GO

CREATE TABLE Products

(

ProductId INT NOT NULL PRIMARY KEY IDENTITY(1, 1),

Name NVARCHAR(100) NOT NULL,

ProductTypeId INT NOT NULL FOREIGN KEY REFERENCES ProductTypes(ProductTypeId) ON UPDATE CASCADE,

Proteins REAL NOT NULL CHECK (Proteins >= 0 AND Proteins <= 100) DEFAULT 0,

Fats REAL NOT NULL CHECK (Fats >= 0 AND Fats <= 100) DEFAULT 0,

Carbohydrates REAL NOT NULL CHECK (Carbohydrates >= 0 AND Carbohydrates <= 100) DEFAULT 0,

PhotoLinkId INT NULL FOREIGN KEY REFERENCES PhotoLinks(PhotoLinkId) ON DELETE SET NULL

)

GO

CREATE TABLE Dishes

(

DishId INT NOT NULL PRIMARY KEY IDENTITY(1, 1),

Name NVARCHAR(100) NOT NULL,

PhotoLinkId INT NULL FOREIGN KEY REFERENCES PhotoLinks(PhotoLinkId),

CookingInstructions NVARCHAR(MAX) NULL

)

GO

CREATE TABLE Ingredients

(

IngredientId INT NOT NULL PRIMARY KEY IDENTITY(1, 1),

DishId INT NOT NULL FOREIGN KEY REFERENCES Dishes(DishId) ON DELETE CASCADE,

ProductId INT NULL FOREIGN KEY REFERENCES Products(ProductId) ON DELETE SET NULL,

[Weight] REAL NULL

)

GO

CREATE TABLE UsersDishes

(

UserDisheId INT NOT NULL PRIMARY KEY IDENTITY(1, 1),

UserId INT NOT NULL FOREIGN KEY REFERENCES Users(UserId) ON DELETE CASCADE,

DishId INT NOT NULL FOREIGN KEY REFERENCES Dishes(DishId) ON DELETE CASCADE

)

-- SPROCS

USE [RecipeBookSystemTestDB]

GO

-----------------------------------------------------------------

-- Users Sprocs

-----------------------------------------------------------------

CREATE PROCEDURE spUsers\_Add(

@Name NVARCHAR(100),

@Email NVARCHAR(200),

@Password NVARCHAR(200)

)

AS

SET NOCOUNT ON

INSERT INTO [dbo].[Users]

VALUES(@Name, @Email, @Password);

RETURN SCOPE\_IDENTITY();

GO

-----------------------------------------------------------------

CREATE PROCEDURE spUser\_Get(

@Email NVARCHAR(200),

@Password NVARCHAR(200)

)

AS

SET NOCOUNT ON

SELECT \* FROM Users

WHERE @Email = Users.Email AND @Password = Users.[Password];

GO

-----------------------------------------------------------------

-----------------------------------------------------------------

-- PhotoLinks Sprocs

-----------------------------------------------------------------

CREATE PROCEDURE spPhotoLinks\_Add(

@SmallPhotoLink NVARCHAR(MAX),

@BigPhotoLink NVARCHAR(MAX)

)

AS

SET NOCOUNT ON

IF ((@SmallPhotoLink IS NOT NULL) OR (@BigPhotoLink IS NOT NULL))

BEGIN

INSERT INTO [dbo].[PhotoLinks]

VALUES (@SmallPhotoLink, @BigPhotoLink);

RETURN SCOPE\_IDENTITY();

END

GO

-----------------------------------------------------------------

-- Products Sprocs

----------------------------------------------------------------

CREATE PROCEDURE spProduct\_AddNew(

@Name NVARCHAR(100),

@ProductTypeId INT,

@Proteins REAL,

@Fats REAL,

@Carbohydrates REAL,

@SmallPhotoLink NVARCHAR(MAX) = NULL,

@BigPhotoLink NVARCHAR(MAX) = NULL

)

AS

SET NOCOUNT ON

DECLARE @CurrentProductPhotoLinkId INT = NULL;

IF ((@SmallPhotoLink IS NOT NULL) OR (@BigPhotoLink IS NOT NULL))

BEGIN

EXECUTE @CurrentProductPhotoLinkId = spPhotoLinks\_Add @SmallPhotoLink, @BigPhotoLink;

END

INSERT INTO Products

VALUES (@Name,

@ProductTypeId,

@Proteins,

@Fats,

@Carbohydrates,

@CurrentProductPhotoLinkId)

GO

-----------------------------------------------------------------

CREATE PROCEDURE spProduct\_Delete(

@ProductId INT

)

AS

SET NOCOUNT ON

DELETE FROM Products

WHERE Products.ProductId = @ProductId

GO

-----------------------------------------------------------------

CREATE PROCEDURE sp\_Product\_GetById(

@ProductId INT

)

AS

SELECT ProductId, Name, t.ProductTypeName, Proteins, Fats, Carbohydrates, f.SmallPhotoLink, f.BigPhotoLink

FROM Products

INNER JOIN (SELECT Name AS ProductTypeName, ProductTypeId

FROM ProductTypes) AS t

ON (t.ProductTypeId = Products.ProductTypeId)

LEFT JOIN (SELECT PhotoLinkId, SmallPhotoLink, BigPhotoLink

FROM PhotoLinks) AS f

ON (f.PhotoLinkId = Products.PhotoLinkId)

WHERE ProductId = @ProductId

GO

-----------------------------------------------------------------

CREATE PROCEDURE spProduct\_SearchByName(

@Name NVARCHAR(100)

)

AS

SET NOCOUNT ON

SELECT ProductId, Name, t.ProductTypeName, Proteins, Fats, Carbohydrates, f.SmallPhotoLink, f.BigPhotoLink

FROM Products

INNER JOIN (SELECT Name AS ProductTypeName, ProductTypeId

FROM ProductTypes) AS t

ON (t.ProductTypeId = Products.ProductTypeId)

LEFT JOIN (SELECT PhotoLinkId, SmallPhotoLink, BigPhotoLink

FROM PhotoLinks) AS f

ON (f.PhotoLinkId = Products.PhotoLinkId)

WHERE Name LIKE (@Name + '%')

GO

-----------------------------------------------------------------

CREATE PROCEDURE spProduct\_Get(

@ProductCount INT,

@PageNumber INT,

@SortColumn NVARCHAR(100) = 'ProductId',

@SortOrder NVARCHAR(10) = 'DESC',

@FilterProductTypeId INT = NULL

)

AS

SET NOCOUNT ON

SELECT ProductId, Name, t.ProductTypeId, t.ProductTypeName, Proteins, Fats, Carbohydrates, f.SmallPhotoLink, f.BigPhotoLink

FROM Products

INNER JOIN (SELECT Name AS ProductTypeName, ProductTypeId

FROM ProductTypes) AS t

ON (t.ProductTypeId = Products.ProductTypeId)

LEFT JOIN (SELECT PhotoLinkId, SmallPhotoLink, BigPhotoLink

FROM PhotoLinks) AS f

ON (f.PhotoLinkId = Products.PhotoLinkId)

WHERE (@FilterProductTypeId IS NULL OR @FilterProductTypeId = Products.ProductTypeId)

ORDER BY

CASE WHEN @SortColumn = 'ProductId' and @SortOrder = 'ASC'

THEN ProductId END ASC,

CASE WHEN @SortColumn = 'ProductId' and @SortOrder = 'DESC'

THEN ProductId END DESC,

CASE WHEN @SortColumn = 'Proteins' and @SortOrder = 'ASC'

THEN Proteins END ASC,

CASE WHEN @SortColumn = 'Proteins' and @SortOrder = 'DESC'

THEN Proteins END DESC,

CASE WHEN @SortColumn = 'Fats' and @SortOrder = 'ASC'

THEN Fats END ASC,

CASE WHEN @SortColumn = 'Fats' and @SortOrder = 'DESC'

THEN Fats END DESC,

CASE WHEN @SortColumn = 'Carbohydrates' and @SortOrder = 'ASC'

THEN Carbohydrates END ASC,

CASE WHEN @SortColumn = 'Carbohydrates' and @SortOrder = 'DESC'

THEN Carbohydrates END DESC,

CASE WHEN @SortColumn = 'Name' and @SortOrder = 'ASC'

THEN Name END ASC,

CASE WHEN @SortColumn = 'Name' and @SortOrder = 'DESC'

THEN Name END DESC

OFFSET (@ProductCount \* (@PageNumber - 1)) ROWS -- skip rows

FETCH NEXT (@ProductCount) ROWS ONLY; -- take rows

GO

-----------------------------------------------------------------

CREATE PROCEDURE spProduct\_Update(

@UpdateProductId INT,

@Name NVARCHAR(100),

@ProductTypeId INT,

@Proteins REAL,

@Fats REAL,

@Carbohydrates REAL,

@SmallPhotoLink NVARCHAR(MAX) = NULL,

@BigPhotoLink NVARCHAR(MAX) = NULL

)

AS

SET NOCOUNT ON

DECLARE @CurrentProductPhotoLinkId INT = NULL;

IF ((@SmallPhotoLink IS NOT NULL) OR (@BigPhotoLink IS NOT NULL))

BEGIN

EXECUTE @CurrentProductPhotoLinkId = spPhotoLinks\_Add @SmallPhotoLink, @BigPhotoLink;

END

UPDATE Products

SET Name = @Name,

ProductTypeId = @ProductTypeId,

Proteins = @Proteins,

Fats = @Fats,

Carbohydrates = @Carbohydrates,

PhotoLinkId = @CurrentProductPhotoLinkId

WHERE (Products.ProductId = @UpdateProductId)

go

-----------------------------------------------------------------

-----------------------------------------------------------------

-- ProductType Sprocs

-----------------------------------------------------------------

CREATE PROC spProductType\_Add(

@Name NVARCHAR(100),

@SmallPhotoLink NVARCHAR(MAX) = NULL,

@BigPhotoLink NVARCHAR(MAX) = NULL

)

AS

SET NOCOUNT ON

DECLARE @CurrentProductTypePhotoLinkId INT = NULL;

EXECUTE @CurrentProductTypePhotoLinkId = spPhotoLinks\_Add @SmallPhotoLink, @BigPhotoLink;

INSERT INTO ProductTypes

VALUES(@Name, @CurrentProductTypePhotoLinkId);

GO

-----------------------------------------------------------------

--EXEC spProductType\_GetAll;

CREATE PROC spProductType\_GetAll

AS

SET NOCOUNT ON

SELECT ProductTypeId, Name, PhotoLinks.SmallPhotoLink, PhotoLinks.BigPhotoLink

FROM ProductTypes

LEFT JOIN PhotoLinks

ON (PhotoLinks.PhotoLinkId = ProductTypes.PhotoLinkId)

GO

-----------------------------------------------------------------

-----------------------------------------------------------------

-- Dish Sprocs

-----------------------------------------------------------------

CREATE PROC spDish\_Add(

@Name NVARCHAR(100),

@OwnerId INT,

@CookingInstructions NVARCHAR(MAX) = NULL,

@SmallPhotoLink NVARCHAR(MAX) = NULL,

@BigPhotoLink NVARCHAR(MAX) = NULL,

@DishId INT OUTPUT

)

AS

SET NOCOUNT ON

DECLARE @CurrentDishPhotoLinkId INT = NULL;

IF ((@SmallPhotoLink IS NOT NULL) OR (@BigPhotoLink IS NOT NULL))

BEGIN

EXECUTE @CurrentDishPhotoLinkId = spPhotoLinks\_Add @SmallPhotoLink, @BigPhotoLink;

END

INSERT INTO Dishes

VALUES(@Name, @CurrentDishPhotoLinkId, @CookingInstructions);

DECLARE @CurrentDishId INT = SCOPE\_IDENTITY();

SELECT @DishId = @CurrentDishId;

INSERT INTO UsersDishes

VALUES(@OwnerId, @CurrentDishId);

GO

-----------------------------------------------------------------

CREATE PROC spDish\_Delete(

@DishId INT

)

AS

SET NOCOUNT ON

DELETE FROM Dishes

WHERE Dishes.DishId = @DishId;

GO

-----------------------------------------------------------------

CREATE PROC spDishes\_Get(

@OwnerId INT,

@DishCount INT,

@PageNumber INT,

@Name NVARCHAR(100) = ''

)

AS

SET NOCOUNT ON

SELECT Dishes.DishId, Name, CookingInstructions, PhotoLinks.SmallPhotoLink, PhotoLinks.BigPhotoLink

FROM Dishes

INNER JOIN UsersDishes

ON Dishes.DishId = UsersDishes.DishId

LEFT JOIN PhotoLinks

ON (PhotoLinks.PhotoLinkId = Dishes.PhotoLinkId)

WHERE (UsersDishes.UserId = @OwnerId) AND (Name LIKE (@Name + '%'))

ORDER BY Dishes.DishId DESC

OFFSET (@DishCount \* (@PageNumber - 1)) ROWS -- skip rows

FETCH NEXT (@DishCount) ROWS ONLY; -- take rows

GO

-----------------------------------------------------------------

-----------------------------------------------------------------

-- Ingrediends Sprocs

-----------------------------------------------------------------

CREATE PROC spDishIngrediends\_Get(

@DishId INT

)

AS

SET NOCOUNT ON

DECLARE @AllProductCount INT;

SELECT IngredientId, DishId, Products.ProductId, Name, [Weight] FROM Ingredients

INNER JOIN Products

ON Products.ProductId = Ingredients.ProductId

WHERE DishId = @DishId

GO

-----------------------------------------------------------------

CREATE PROC spIngredient\_Add(

@DishId INT,

@ProductId INT,

@Weight REAL

)

AS

SET NOCOUNT ON

INSERT INTO Ingredients

VALUES (@DishId, @ProductId, @Weight)

GO

using RecipeBookSystem.DAL.Abstraction.Repositories;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RecipeBookSystem.DAL.Abstraction.UnitOfWork

{

public interface IUnitOfWork

{

IGenericRepository<TEntity> GetRepository<TEntity>() where TEntity : class, new();

IDishRepository DishRepository { get; }

IProductRepository ProductRepository { get; }

IIngredientRepository IngredientRepository { get; }

IUserRepository UserRepository { get; }

IProductTypeRepository ProductTypeRepository { get; }

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RecipeBookSystem.DAL.Concrete.SQL

{

public static class DataRecordExtensions

{

/// <summary>

/// Get value of specific column from data record

/// </summary>

/// <typeparam name="T">Type of column</typeparam>

/// <param name="record">Data record</param>

/// <param name="columnName">Column name</param>

/// <returns>

/// Returns value of column.

/// If value is null - returns default value for specific type.

/// </returns>

public static T Get<T>(this IDataRecord record, string columnName)

{

if (HasColumn(record, columnName))

{

var result = record[columnName] is DBNull

? default(T)

: (T)(record[columnName]);

return result;

}

return default(T);

}

/// <summary>

/// The method checks for columns in DB record

/// </summary>

/// <param name="record">Record object to search for value </param>

/// <param name="columnName"></param>

/// <returns>Returns true if column is existed in record, otherwise - false</returns>

private static bool HasColumn(IDataRecord record, string columnName)

{

for (int i = 0; i < record.FieldCount; i++)

{

if (string.Equals(record.GetName(i), columnName, StringComparison.OrdinalIgnoreCase))

{

return true;

}

}

return false;

}

}

}

using RecipeBookSystem.DAL.Abstraction.Repositories;

using RecipeBookSystem.DAL.Concrete.Parsers;

using RecipeBookSystem.DAL.Concrete.SQL;

using RecipeBookSystem.Model.Models;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

namespace RecipeBookSystem.DAL.Concrete.Repositories

{

public class DishRepository : GenericRepository<DishModel>, IDishRepository

{

public DishRepository(string connection)

: base(connection)

{

}

/// <summary>

/// Gets dishes for some user

/// </summary>

/// <param name="userId">Owner id</param>

/// <param name="count">Count of products to retrieve</param>

/// <param name="pageNumber">Page index</param>

/// <param name="name">Name for searching</param>

/// <returns>List of dishes</returns>

public IEnumerable<DishModel> GetUserDishes(int userId, int count, int pageNumber, string name = null)

{

var parameters = new[]

{

new SqlParameter(StoredProcedureParameters.OwnerId, userId),

new SqlParameter(StoredProcedureParameters.DishCount, count),

new SqlParameter(StoredProcedureParameters.PageNumber, pageNumber),

new SqlParameter(StoredProcedureParameters.Name, name)

};

var dishIngredients = base.ExecuteReader(StoredProcedureNames.spGetUserDishes,

DishParser.MakeBuildingResult, parameters);

return dishIngredients;

}

/// <summary>

/// Adds new user model to DB

/// </summary>

/// <param name="dish">Dish model to add</param>

/// <returns>Id of new dish</returns>

public int AddDish(DishModel dish)

{

SqlParameter outputDishIdParam = new SqlParameter(StoredProcedureParameters.DishId, SqlDbType.Int)

{

Direction = ParameterDirection.Output

};

var parameters = new[]

{

new SqlParameter(StoredProcedureParameters.OwnerId, dish.OwnerId),

new SqlParameter(StoredProcedureParameters.Name, dish.Name),

new SqlParameter(StoredProcedureParameters.CookingInstructions, dish.CookingInstructions),

new SqlParameter(StoredProcedureParameters.SmallPhotoLink, dish.SmallPhotoLink),

new SqlParameter(StoredProcedureParameters.BigPhotoLink, dish.BigPhotoLink),

outputDishIdParam

};

base.ExecuteReader(StoredProcedureNames.spAddDish, null, parameters);

return Convert.ToInt32(outputDishIdParam.Value.ToString());

}

/// <summary>

/// Method removes dish

/// </summary>

/// <param name="dishId">Id of user to delete</param>

public void DeleteDish(int dishId)

{

var parameters = new[]

{

new SqlParameter(StoredProcedureParameters.DishId, dishId)

};

base.ExecuteReader(StoredProcedureNames.spDeleteDish, null, parameters);

}

}

}

using RecipeBookSystem.DAL.Concrete.SQL;

using RecipeBookSystem.Model.Models;

using System.Data.SqlClient;

namespace RecipeBookSystem.DAL.Concrete.Parsers

{

public static class DishParser

{

public static DishModel MakeBuildingResult(SqlDataReader reader)

{

var model = new DishModel();

model.Id = reader.Get<int>(ColumnNames.DishId);

model.Name = reader.Get<string>(ColumnNames.Name);

model.CookingInstructions = reader.Get<string>(ColumnNames.CookingInstructions);

model.SmallPhotoLink = reader.Get<string>(ColumnNames.SmallPhotoLink);

model.BigPhotoLink = reader.Get<string>(ColumnNames.BigPhotoLink);

return model;

}

}

}

using System.Collections.Generic;

namespace RecipeBookSystem.Model.Models

{

/// <summary>

/// The class that describes model of the dish

/// </summary>

public class DishModel

{

/// <summary>

/// Identity of the dish

/// </summary>

public int Id { get; set; }

/// <summary>

/// Name of the dish

/// </summary>

public string Name { get; set; }

/// <summary>

/// Identity of user that has the dish

/// </summary>

public int OwnerId { get; set; }

/// <summary>

/// Description of cooking proccess

/// </summary>

public string CookingInstructions { get; set; }

/// <summary>

/// Link on the small photo(icon) in the internet

/// </summary>

public string SmallPhotoLink { get; set; }

/// <summary>

/// Link on the big photo in the internet

/// </summary>

public string BigPhotoLink { get; set; }

/// <summary>

/// List of the included ingredients

/// </summary>

public List<IngredientModel> Ingredients { get; set; }

}

}

using MaterialSkin.Controls;

using RecipeBookSystem.BL.Helpers;

using RecipeBookSystem.BL.ModelProviders;

using RecipeBookSystem.Model.Models;

using RecipeBookSystem.UI.Models;

using RecipeBookSystem.UI.Models.FormModels;

using RecipeBookSystem.UI.Properties;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Configuration;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Windows.Forms;

namespace RecipeBookSystem.UI

{

public partial class MainForm : MaterialFormBaseModel

{

private const int sortButtonCount = 4;

private const int tableColumnCount = 7;

private const int productsPerPageCount = 5;

private const int dishesPerPageCount = 2;

private const int productImageWigth = 80;

private const int productImageHeight = 64;

private const int dishImageWeigth = 240;

private const int dishImageHeight = 144;

private const int nameSortButtonPosition = 0;

private const int proteinsSortButtonPosition = 1;

private const int fatsSortButtonPosition = 2;

private const int carbsSortButtonPosition = 3;

private const int ingredientNameTablePosition = 0;

private const int ingerdientWeightTablePosition = 1;

private readonly UserModel loggedUser;

private readonly ProductProvider productProvider;

private readonly ProductTypeProvider productTypeProvider;

private readonly ImageHelper imageHelper;

private readonly IngredientProvider ingredientProvider;

private readonly DishProvider dishProvider;

private readonly Font commonFont;

private readonly Padding commonTablePadding;

private List<ProductItem> productItems;

private List<ProductModel> selectedProducts;

private List<ProductTypeModel> productTypes;

private Control[] sortButtons;

private ProductsGridOptions productsGridOptions;

private DishModel newDishModel;

private List<IngredientModel> newRecipeIngredients;

private List<DishItem> dishItems;

private DishGridOptions dishGridOptions;

private ProductModel updatingProduct;

private ProductModel newProduct;

private ProductsManipulationPlan productsManipulationPlan;

private Control[] productTableRowLabels;

public MainForm(UserModel loggedUser)

{

InitializeComponent();

this.loggedUser = loggedUser;

productProvider = new ProductProvider();

imageHelper = new ImageHelper(new CloudHelper());

productTypeProvider = new ProductTypeProvider();

ingredientProvider = new IngredientProvider();

dishProvider = new DishProvider();

//Common Font for all Labels in the forms

commonFont = new Font("Berlin Sans FB", 14F);

//Common Table Padding without horisontal ScrollBar

commonTablePadding = new Padding(0, 0, SystemInformation.VerticalScrollBarWidth, 0);

}

private void MainForm\_Load(object sender, EventArgs e)

{

this.productsManipulationPlan = ProductsManipulationPlan.CreatingNewProduct;

//Initialize all pages

InitializeProductList();

InitializeRecipeAdding();

InitializeDishList();

}

#region Product list view logic

private void InitializeProductList()

{

this.newProduct = new ProductModel();

this.newProduct.ProductTypeModel = new ProductTypeModel();

this.updatingProduct = new ProductModel();

this.updatingProduct.ProductTypeModel = new ProductTypeModel();

this.creatingProductPhotoLabel.Image = imageHelper.GetProductDefaultImage();

this.selectedProducts = new List<ProductModel>();

this.productItems = new List<ProductItem>();

this.sortButtons = new Control[sortButtonCount];

productsGridOptions = new ProductsGridOptions()

{

ItemsCount = productsPerPageCount,

PageNumber = 1

};

try

{

this.productTypes = productTypeProvider.GetAllProductTypes().ToList();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message, Resources.ErrorMessage);

}

//Fill filterProductComboBox with all available product type

foreach (var productType in productTypes)

{

filterProductComboBox.Items.Insert(productType.Id, productType.Name);

creatingProductTypeSelector.Items.Insert(productType.Id, productType.Name);

}

this.sortButtons[nameSortButtonPosition] = nameSortButton;

this.sortButtons[proteinsSortButtonPosition] = proteinsSortButton;

this.sortButtons[fatsSortButtonPosition] = fatsSortButton;

this.sortButtons[carbsSortButtonPosition] = carbsSortButton;

var leftSideImage = Resources.LeftSideImage;

var rightSideImag = Resources.RightSideImage;

this.leftSideProductListButton.Image = leftSideImage;

this.rightSideProductListButton.Image = rightSideImag;

this.leftSideProductListButton.Enabled = false;

this.filterProductComboBox.SelectedIndex = 0;

this.creatingProductTypeSelector.SelectedIndex = 0;

this.makeRecipeButton.Hide();

//Filter froduct when value of ComboBox is changed

this.filterProductComboBox.SelectedValueChanged += (sender, e) =>

{

filterProducts();

};

this.backgroundProductImageUploader.WorkerSupportsCancellation = true;

showProducts();

this.backgroundProductImageUploader.RunWorkerAsync();

}

private void clearTable()

{

this.productTableView.Controls.Clear();

this.productItems = new List<ProductItem>();

}

private void changePlan(ProductItem productItem)

{

if (selectedProducts.Count > 1)

{

this.makeRecipeButton.Show();

}

else if (selectedProducts.Count == 1)

{

this.makeRecipeButton.Hide();

}

else

{

//Hide makeRecipeButton when there no eny selected product

this.makeRecipeButton.Hide();

}

}

private void showProducts()

{

this.backgroundProductImageUploader.CancelAsync();

clearTable();

int row = productItems.Count;

List<ProductModel> filteredProducts = new List<ProductModel>();

//In case when there are no

if (string.IsNullOrEmpty(productsGridOptions.searchProductName))

{

try

{

filteredProducts = productProvider.GetProducts(

productsPerPageCount,

productsGridOptions.PageNumber,

productsGridOptions.SortColumnName,

productsGridOptions.SortOrder,

productsGridOptions.FilterProductTypeId).ToList();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, Resources.ErrorMessage);

return;

}

}

else

{

this.productsGridOptions.PageNumber = 1;

this.leftSideProductListButton.Enabled = false;

try

{

filteredProducts = productProvider.SearchProductByName(

productsGridOptions.searchProductName).ToList();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, Resources.ErrorMessage);

return;

}

}

if (filteredProducts.Count == 0)

{

this.noMoreProductMessageLabel.Visible = true;

this.rightSideProductListButton.Enabled = false;

return;

}

this.noMoreProductMessageLabel.Visible = false;

this.rightSideProductListButton.Enabled = true;

foreach (var product in filteredProducts)

{

MaterialCheckBox productCheckBox = new MaterialCheckBox();

productCheckBox.Anchor = AnchorStyles.Bottom & AnchorStyles.None;

this.productTableRowLabels = new Control[tableColumnCount];

foreach (var selectedProduct in selectedProducts)

{

if (selectedProduct.Id == product.Id)

{

productCheckBox.Checked = true;

}

}

this.productTableRowLabels[0] = productCheckBox;

int initialTableDataColIndex = 2;

for (int i = initialTableDataColIndex; i < productTableRowLabels.Length; i++)

{

this.productTableRowLabels[i] = new Label();

this.productTableRowLabels[i].AutoSize = true;

this.productTableRowLabels[i].Anchor = AnchorStyles.None;

this.productTableRowLabels[i].Font = commonFont;

this.productTableView.Controls.Add(productTableRowLabels[i], i, row);

}

var productItem = new ProductItem(product);

productItems.Add(productItem);

//Each time will be selecting an item in front of the Check box

productCheckBox.CheckedChanged += (sender, e) =>

{

var chackBox = (MaterialCheckBox)sender;

if (chackBox.Checked)

{

if (selectedProducts.Count == 0)

{

selectedProductsListBox.Items.RemoveAt(0);

}

this.selectedProductsListBox.Items.Insert(0, productItem.ProductModel.Name);

this.selectedProducts.Add(productItem.ProductModel);

}

else

{

this.selectedProducts.RemoveAll((p) => (p.Id == productItem.ProductModel.Id));

this.selectedProductsListBox.Items.Remove(productItem.ProductModel.Name);

if (selectedProducts.Count == 0)

{

this.selectedProductsListBox.Items.Insert(0, Resources.NoProductsYetMessage);

}

}

changePlan(productItem);

};

this.productTableView.Controls.Add(productCheckBox, 0, row);

this.productTableView.Controls.Add(productTableRowLabels[0], 0, row);

var deleteIconLabel = new Label();

var editIconLabel = new Label();

editIconLabel.Anchor = AnchorStyles.Left;

deleteIconLabel.Anchor = AnchorStyles.Left;

deleteIconLabel.Width = 25;

deleteIconLabel.Image = Resources.DeleteIcon;

editIconLabel.Image = Resources.EditIcon;

deleteIconLabel.Click += (sender, e) =>

{

deleteProduct(productItem.ProductModel);

};

editIconLabel.Click += (sender, e) =>

{

setProductToUpdate(productItem);

};

this.productTableView.Controls.Add(editIconLabel, 7, row);

this.productTableView.Controls.Add(deleteIconLabel, 8, row);

//Configure Label element, so that there could put the picture there

productItem.PictureLabel = new Label();

productItem.PictureLabel.AutoSize = true;

productItem.PictureLabel.Font = new Font("Microsoft Sans Serif", 50F);

productItem.PictureLabel.Text = string.Format("{0,4}", string.Empty);

productItem.PictureLabel.Anchor = AnchorStyles.None;

this.productTableView.Controls.Add(productItem.PictureLabel, 1, row);

//Set the text values of Label content in the correct order

this.productTableRowLabels[2].Text = product.Name;

this.productTableRowLabels[3].Text = product.ProductTypeModel.Name;

this.productTableRowLabels[4].Text = product.Proteins.ToString();

this.productTableRowLabels[5].Text = product.Fats.ToString();

this.productTableRowLabels[6].Text = product.Carbohydrates.ToString();

row++;

this.productTableView.Refresh();

}

}

private void deleteProduct(ProductModel productToDelete)

{

try

{

this.productProvider.DeleteProduct(productToDelete.Id);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, Resources.ErrorMessage);

return;

}

foreach (var selectedProduct in selectedProducts)

{

if (selectedProduct.Id == productToDelete.Id)

{

this.selectedProductsListBox.Items.Remove(productToDelete.Name);

break;

}

}

this.selectedProducts.RemoveAll((p) => (p.Id == productToDelete.Id));

if (selectedProductsListBox.Items.Count == 0)

{

this.selectedProductsListBox.Items.Insert(0, Resources.NoProductsYetMessage);

}

showProducts();

if (!backgroundProductImageUploader.IsBusy)

{

this.backgroundProductImageUploader.RunWorkerAsync();

}

}

private void setProductToUpdate(ProductItem productToUpdate)

{

this.newProduct.SmallPhotoLink = productToUpdate.ProductModel.SmallPhotoLink;

this.updete\_createProductButton.Text = Resources.UpdateText;

this.updatingProduct = productToUpdate.ProductModel;

this.productsManipulationPlan = ProductsManipulationPlan.UpdatingExistingProduct;

//Fill fields whith values of available product

this.creatingProductNameTextField.Text = productToUpdate.ProductModel.Name;

this.creatingProductProteinsTextField.Text = productToUpdate.ProductModel.Proteins.ToString();

this.creatingProductFatsTextField.Text = productToUpdate.ProductModel.Fats.ToString();

this.creatingProductCarbsTextField.Text = productToUpdate.ProductModel.Carbohydrates.ToString();

this.creatingProductPhotoLabel.Image = productToUpdate.PictureLabel.Image;

this.creatingProductTypeSelector.SelectedIndex = productToUpdate.ProductModel.ProductTypeModel.Id;

this.pages.SelectedTab = addingProductPage;

}

private void backgroundProductImageUploader\_DoWork(object sender, DoWorkEventArgs e)

{

//New Thread created to provide releasing the background

//worker for next image loading process.

Thread imageLoader = new Thread(() =>

{

showProductsImages();

});

imageLoader.Start();

}

private void showProductsImages()

{

foreach (var item in productItems)

{

if (item.PictureLabel.Image == null)

{

item.PictureLabel.Image = Resources.Spinner;

}

}

foreach (var item in productItems)

{

item.PictureLabel.Image = imageHelper.GetProductImage(item.ProductModel.SmallPhotoLink);

}

}

private void nameSortButton\_Click(object sender, EventArgs e)

{

sortProducts((Button)sender, Resources.NameText);

}

private void proteinsSortButton\_Click(object sender, EventArgs e)

{

sortProducts((Button)sender, Resources.ProteinsText);

}

private void fatsSortButton\_Click(object sender, EventArgs e)

{

sortProducts((Button)sender, Resources.FatsText);

}

private void carbsSortButton\_Click(object sender, EventArgs e)

{

sortProducts((Button)sender, Resources.CarbohydratesText);

}

private void sortProducts(Button clickedButton, string sortColumn)

{

this.leftSideProductListButton.Enabled = false;

clickedButton.Enabled = false;

string upper = Resources.UpperSymbol;

string lower = Resources.LowerSymbol;

foreach (var button in sortButtons)

{

if (button != clickedButton)

{

button.Text = button.Text.Split(' ')[0];

}

}

clearTable();

this.productsGridOptions.SortColumnName = sortColumn;

this.productsGridOptions.PageNumber = 1;

string[] buttonTextWords = clickedButton.Text.Split(' ');

if (buttonTextWords.Length > 1)

{

if (buttonTextWords[1] == upper)

{

clickedButton.Text = buttonTextWords[0] + " " + lower;

this.productsGridOptions.SortOrder = Resources.ASCOrder;

}

else if (buttonTextWords[1] == lower)

{

clickedButton.Text = buttonTextWords[0] + " " + upper;

this.productsGridOptions.SortOrder = Resources.DESCOrder;

}

}

else

{

clickedButton.Text += " " + lower;

this.productsGridOptions.SortOrder = Resources.ASCOrder;

}

showProducts();

if (!backgroundProductImageUploader.IsBusy)

{

this.backgroundProductImageUploader.RunWorkerAsync();

}

clickedButton.Enabled = true;

}

private void leftSideButton\_Click(object sender, EventArgs e)

{

var clicked = (Button)sender;

clicked.Enabled = false;

this.productsGridOptions.PageNumber--;

showProducts();

if (!backgroundProductImageUploader.IsBusy)

{

this.backgroundProductImageUploader.RunWorkerAsync();

}

clicked.Enabled = (productsGridOptions.PageNumber != 1);

}

private void rightSideButton\_Click(object sender, EventArgs e)

{

var clicked = (Button)sender;

clicked.Enabled = false;

this.leftSideProductListButton.Enabled = true;

this.productsGridOptions.PageNumber++;

showProducts();

if (!backgroundProductImageUploader.IsBusy)

{

this.backgroundProductImageUploader.RunWorkerAsync();

}

}

private void filterProducts()

{

this.leftSideProductListButton.Enabled = false;

var selectedTypeId = filterProductComboBox.SelectedIndex;

this.productsGridOptions.PageNumber = 1;

if (selectedTypeId == 0)

{

this.productsGridOptions.FilterProductTypeId = null;

}

else

{

this.productsGridOptions.FilterProductTypeId = selectedTypeId;

}

showProducts();

if (!backgroundProductImageUploader.IsBusy)

{

this.backgroundProductImageUploader.RunWorkerAsync();

}

}

private void searchProductTextField\_TextChanged(object sender, EventArgs e)

{

clearTable();

string name = this.searchProductTextField.Text;

bool isEmptyName = (name == string.Empty);

foreach (var sortButton in this.sortButtons)

{

sortButton.Visible = isEmptyName;

}

this.filterProductComboBox.Visible = isEmptyName;

IEnumerable<ProductModel> searchedProducts;

try

{

searchedProducts = this.productProvider.SearchProductByName(name);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, Resources.ErrorMessage);

return;

}

this.productsGridOptions.searchProductName = name;

showProducts();

this.pages.SelectedTab = productsPage;

if (!backgroundProductImageUploader.IsBusy)

{

this.backgroundProductImageUploader.RunWorkerAsync();

}

}

private void addNewProductButton\_Click(object sender, EventArgs e)

{

setAddProductPlan();

}

private void noMoreProductMessageLabel\_Click(object sender, EventArgs e)

{

setAddProductPlan();

}

private void setAddProductPlan()

{

clearAddingProductPage();

this.productsManipulationPlan = ProductsManipulationPlan.CreatingNewProduct;

this.pages.SelectedTab = addingProductPage;

this.updete\_createProductButton.Text = Resources.AddProductMessage;

}

#endregion

#region Product update/add view logic

private void clearAddingProductPage()

{

this.creatingProductTypeSelector.SelectedIndex = 0;

this.newProduct.SmallPhotoLink = null;

this.updatingProduct.SmallPhotoLink = null;

this.creatingProductPhotoLabel.Image = imageHelper.GetProductDefaultImage();

this.creatingProductNameTextField.Text = string.Empty;

this.creatingProductFatsTextField.Text = string.Empty;

this.creatingProductCarbsTextField.Text = string.Empty;

this.creatingProductProteinsTextField.Text = string.Empty;

}

private void makeRecipeButton\_Click(object sender, EventArgs e)

{

this.pages.SelectedTab = addingRecipePage;

}

private void addingRecipeMessageLabel\_Click(object sender, EventArgs e)

{

this.pages.SelectedTab = productsPage;

}

private void pages\_Selected(object sender, TabControlEventArgs e)

{

if (pages.SelectedTab == addingRecipePage)

{

this.addingRecipeMessageLabel.Visible = (selectedProducts.Count < 2);

}

}

private void updete\_createProductButton\_Click(object sender, EventArgs e)

{

string newProductName = creatingProductNameTextField.Text;

int newProductTypeId = creatingProductTypeSelector.SelectedIndex;

float newProductProteinsCount;

bool hasProteinsValue = Single.TryParse(

this.creatingProductProteinsTextField.Text,

out newProductProteinsCount);

float newProductFatsCount;

bool hasFatsValue = Single.TryParse(

this.creatingProductFatsTextField.Text,

out newProductFatsCount);

float newProductCarbsCount;

bool hasCarbsValue = Single.TryParse(

this.creatingProductCarbsTextField.Text,

out newProductCarbsCount);

if (newProductName == string.Empty)

{

MessageBox.Show("Do not leave a clear name field!", Resources.WarningMessage);

return;

}

else if (newProductTypeId == 0)

{

MessageBox.Show("Select the ptoduct type!", Resources.WarningMessage);

return;

}

else if (newProductName.Length > 20)

{

MessageBox.Show("Name is too long!", Resources.WarningMessage);

return;

}

else if (!hasProteinsValue)

{

MessageBox.Show("Incorrect input in proteins field", Resources.WarningMessage);

return;

}

else if (!hasFatsValue)

{

MessageBox.Show("Incorrect input in fats field", Resources.WarningMessage);

return;

}

else if (!hasCarbsValue)

{

MessageBox.Show("Incorrect input in carbohydrades field", Resources.WarningMessage);

return;

}

else if ((newProductProteinsCount + newProductCarbsCount + newProductFatsCount) > 100)

{

MessageBox.Show("The sum of all product parameters can not be more than 100!", Resources.WarningMessage);

return;

}

this.newProduct.Name = newProductName;

this.newProduct.Proteins = newProductProteinsCount;

this.newProduct.Fats = newProductFatsCount;

this.newProduct.Carbohydrates = newProductCarbsCount;

this.newProduct.ProductTypeModel.Id = newProductTypeId;

if (productsManipulationPlan == ProductsManipulationPlan.UpdatingExistingProduct)

{

this.newProduct.Id = updatingProduct.Id;

try

{

productProvider.UpdateProduct(

newProduct.Id,

newProduct.Name,

newProduct.ProductTypeModel.Id,

newProduct.Proteins,

newProduct.Fats,

newProduct.Carbohydrates,

newProduct.SmallPhotoLink);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, Resources.ErrorMessage);

return;

}

}

else

{

try

{

this.productProvider.AddProduct(newProduct);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, Resources.ErrorMessage);

return;

}

this.productsGridOptions.FilterProductTypeId = null;

this.productsGridOptions.PageNumber = 1;

}

clearTable();

showProducts();

pages.SelectedTab = productsPage;

if (!backgroundProductImageUploader.IsBusy)

{

this.backgroundProductImageUploader.RunWorkerAsync();

}

clearAddingProductPage();

this.updete\_createProductButton.Text = Resources.AddProductMessage;

}

private void photoAddMessageLabel\_Click(object sender, EventArgs e)

{

this.creatingProductPhotoLabel.Image = Resources.Spinner;

string newImagePath = getNewImagePath();

if (string.IsNullOrEmpty(newImagePath))

{

return;

}

var newImage = new Bitmap(newImagePath);

if (newImage != null)

{

Thread newImageLoader = new Thread(() =>

{

this.newProduct.SmallPhotoLink = loadNewImage(

this.creatingProductPhotoLabel,

newImage,

imageHelper.GetProductDefaultImage(),

productImageWigth,

productImageHeight);

});

newImageLoader.Start();

}

}

private string getNewImagePath()

{

var openFileDialog = new OpenFileDialog();

openFileDialog.Filter = ConfigurationManager.AppSettings["Support\_Image\_Filter"];

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

return openFileDialog.FileName;

}

else

{

return string.Empty;

}

}

private string loadNewImage(

Label element,

Bitmap image,

Bitmap defaultImage,

int weidht,

int heigth)

{

try

{

string newImageLink = imageHelper.UploadImage(image, weidht, heigth);

element.Image = imageHelper.GetProductImage(newImageLink);

return newImageLink;

}

catch (Exception ex)

{

element.Image = defaultImage;

return null;

}

}

#endregion

#region Recipe add view logic

private void InitializeRecipeAdding()

{

ingredientsTableView.AutoScroll = true;

ingredientsTableView.Padding = commonTablePadding;

this.newRecipeImageLabel.Image = imageHelper.GetDishDefaultImage();

this.newDishModel = new DishModel();

this.firstDishDeleteIconLabel.Image = Resources.DeleteIcon;

this.secondDishDeleteIconLabel.Image = Resources.DeleteIcon;

}