МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра ІСМ

***КУРСОВА РОБОТА***

З дисципліни “Технології створення програмних продуктів.”

На тему:

“Система фінансового обліку проектів.”

Виконав студент групи КН-37 Шандра О.С.

|  |  |  |
| --- | --- | --- |
| Оцінка | Балів | Дата |
|  |  |  |

Номер залікової книжки:

1508522

Керівник роботи доцент кафедри ІСМ Басюк Т. М.

Львів 2018

Завдання на курсову роботу та календарний план

|  |  |
| --- | --- |
| *Завдання до виконання* | *Дата* |
| Анотація | 12.02.2018 |
| Вступ | 15.02.2018 |
| **Розділ 1. Аналіз літературних джерел та предметної галузі** | 02.03.2018 |
| 1.1 Основні засади дослідження | 12.03.2018 |
| 1.2 Аналіз відомих засобів вирішення проблеми | 13.03.2018 |
| 1.3 Функціональність програмної системи | 18.03.2018 |
| **Розділ 2. Вибір методології проектування та планування робіт** | 20.03.2018 |
| 2.1 Опис системних вимог згідно з методологією RUP | 20.03.2018 |
| 2.2 Деталізація функцій та аналіз категорій користувачів | 22.03.2018 |
| 2.3 Аналіз функціональних точок | 23.03.2018 |
| 2.4 Планування робіт в програмному засобі Gantt Project | 12.04.2014 |
| **Розділ 3. Вибір засобів реалізації та проектування системи** |  |
| 3.1 Вибір архітектурного та програмного рішення | 16.04.2014 |
| 3.2 Реалізація програмного засобу | 18.04.2014 |
| 3.3 Системне тестування програмного засобу | 30.04.2014 |
| 3.4 Оцінка якості створеного програмного забезпечення | 3.05.2014 |
| **Висновок** |  |
| **Список використаних джерел** |  |
| **Додатки** |  |

Завдання прийнято до виконання \_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис студента)

Керівник роботи к.т.н., доц. Басюк Т.М.

Анотація

В цій курсовій роботі було розроблено програму для фінансового обліку ІТ проектів. Для реалізацї було використано мову програмування С# на платформі .Net.

Програма представлена у вигляді зручного для користувача веб інтерфейсу реалізованого засобами веб-програмування а саме мовами програмування JavaScript(та її бібліотекою Angular), HTML, CSS та платформою ASP.NET.

Розроблена програма буде корисною для автоматизації ведення фінансового обліку проектів. Нею можуть користуватися керівники ІТ проектів та бухгалтери компаній що займаються розробкою програмного забезпеченя.

Summary

In this course work, a program for the financial accounting of IT projects was developed. For implementation, the C # programming language was used on the .Net platform.

The program is presented in the form of a user-friendly web interface implemented by means of web programming, in particular JavaScript programming languages ​​(and its Angular library), HTML, CSS and the ASP.NET platform.

The developed program will be useful for automating financial accounting of projects. It can be used by IT project managers and accountants of software development companies.

ЗМІСТ

[РОЗДІЛ 1. АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ ТА ПРЕДМЕТНОЇ ГАЛУЗІ………………………………………………………………………….……](#_Toc483156747) 5

[1.1. Основні засади дослідження](#_Toc483156748) 6

[1.2. Аналіз відомих засобів вирішення проблеми](#_Toc483156749) 7

[1.3. Функціональність програмної системи](#_Toc483156750) 9

[РОЗДІЛ 2. ВИБІР МЕТОДОЛОГІЇ ПРОЕКТУВАННЯ ТА ПЛАНУВАННЯ РОБІТ](#_Toc483156751)

[2.1. Опис системних вимог згідно з методологією Rational Unified Process](#_Toc483156752) 10

[2.2. Деталізація функцій та аналіз категорій користувачів](#_Toc483156753) 12

[2.3. Аналіз функціональних точок](#_Toc483156754) 16

[РОЗДІЛ 3. ВИБІР ЗАСОБІ РЕАЛІЗАЦІЇ ТА ПРОЕКТУВАННЯ СИСТЕМИ](#_Toc483156751)

3.1 Вибір архітектурного та програмного рішення…………………………….21

3.2 Реалізація програмного засобу………………………………………………24

3.3 Системне тестування програмного засобу………………………………….30

3.4 Оцінка якості створеного програмного забезпечення……………………...32

Висновок…………………………………………………………………………….34

Список використаних дерел………………………………………………………..35

Додатки……………………………………………………………………………...36

**ВСТУП**

*Актуальність теми*. Дана тема є особливо актуальною в нас час коли є стрімкий розвиток індустрії інформаційних технологій і приділяється велика увага вирішенню питань з ведення фінансогового обліку проектів у галузі.

*Мета і задачі дослідження*. Метою дослідження стало розроблення системи фінансогового обліку проектів що допоможе розвязати та автоматизувати ряд задач галузі. Для цього слід зробити наступні кроки:

* Проаналізувати стан ринку та наявні схожі системи фінансового обліку проетів.
* Спроектувати свою систему яка викрнуватиме завдання які ще не були реалізовані в інших системах.
* Обрати засоби реалізації системи.
* Реалізувати ряд остновних функцій системи.
* Протестувати систему.
* Зробити висновки та проаналізувати якість створеного продукту.

*Об’єкт дослідження.* Процес фінансового обліку проектів.

*Предмет дослідження.* Ведення фінансового обліку проектів*.*

*Практичне значення одержаних результатів.* Програма буде надавати можливість ведення фінансового обліку проектів що ведуться в інших системащ контролю проектів і на основі даних з них будуватиме статистику, оцінюватиме прибутковість проетку чи компанії, створюватиме рахунки які можна буде відправляти клієнтам.

# **РОЗДІЛ 1. АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ ТА ПРЕДМЕТНОЇ ГАЛУЗІ**

## **Основні засади дослідження**

Проек – це вид діяльності що має на меті створення деякого продукту чи послуги протягом визначеного терміну та зе деяких фінансових обмежень. Будь який проект має обмежений набір людських, матеріальних та фінансових ресурсів і управляння проетами означає процес керівництва усіма цими ресурсами. Проектний аналіз і керування фінанвовими ресурсами проекту стало відіграє велику роль у формуванні успішного з усіх точок зору проекту.

Сукупний фінансовий стан всіх проектів деякої ІТ компанії означає її рентабельність на ринку, і ведення фінансового обліку кожного проекту компанії є невід’ємною складовою ведення даного бізнесу. Прибутковість проектів визначається виручкою після відшкодування всіх витрат на виробничу діяльність (людський ресурс).

Надалі ми розглядатимемо лише ІТ компані що займаються аутсорсингом, оскільки такий вид надання послуг в ІТ сфері є найбільш поширеним в Україні. Це зобумовлено на сам перед високою зарплатнею людей в даній сфері в країнах з добре розвинутою економікою і хорошим професійним рівнем наших працівників з відносно невеликою заробітню плату.

Проблема фінансового обліку проектів є особливо актуальною в наш час і автоматизація даного виду діяльності значною мірою покращить ефективність ведення бізнесу і дасть чітку картину рентабельності проекту і компанії загалом. Методи фінансового обліку проектів є добре розвиненими в наш час проте багато інформаційних рішень даної галузі є недостатньо комплексними і потребують подальшого розвитку і вдосконалення.

Найбільш відомою проблемами галузі є гнучкість такої системи та здатність адаптуватися під різні ІТ проекти. Як відомо, для керування проектами використовуються різні системи керування проектами на кшталт Jira, Team Foundation server та інші. Нездатність системи фінансового обліку проектів використовувати інформацю з даних систем контролю проектів призводить до надлишковості та дублювання інформації в різних системах. Як приклад, для різних цілей учасники проекту починають використовувати різні системи (для ведення інформації про завдання – деяку систему контролю версій, а для звітування відпрацьованого часу іншу систему вже для фінансового обліку проекту).

Є безліч рішень даної проблеми зі своїми перевагами і недоліками. Одні передбачають використання плагінів до різних систем контролю версій, інші передбачають створення ІТ компаніями своїх систем для контролю проектів з можливістю фінансового їх обліку. Проблема першого є відсутність зв’язку з іншими системами фінансового обліку проетів до інших систем керування проектами окрім однієї, тісно зав’язеної системи. Проблема другого рішення є тісна прив’язка до однієї системи контролю проектів, складність і недоцільність реалізації такої комплексної програми за наявності ряду доступних аналогів(систем контролю ІТ проектів).

**1.2. Аналіз відомих засобів вирішення проблеми**

Інує ряд програм які виконують схожі функції до нашої систмеи фінансового обліку проектів зі своїми сильними і слабкими сторонами:

* Hubstaf

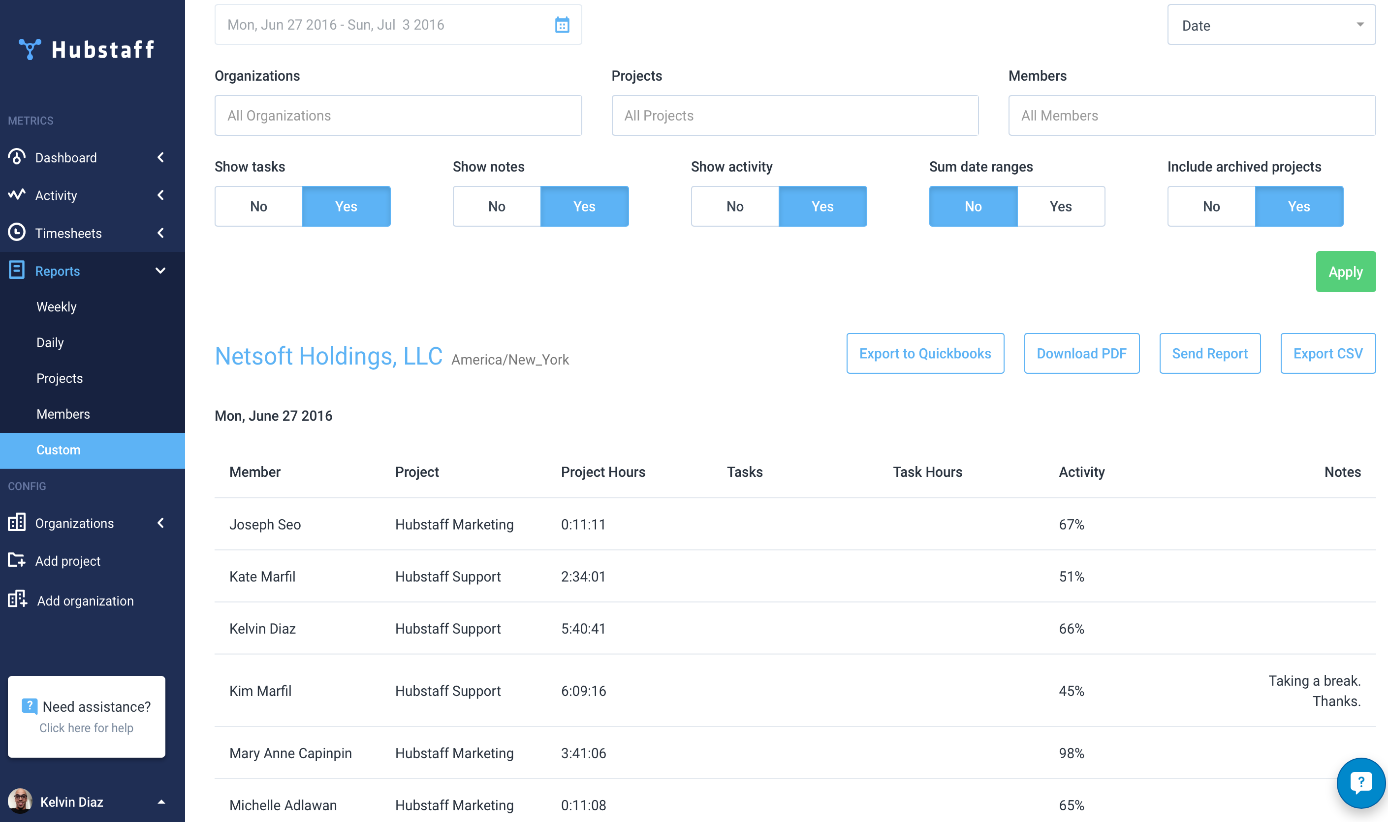
Часовий трекер Hubstaff (рис. 1) працює як програмне забезпечення на вашому компютері або як мобільний додаток, що дозволяє легко відстежувати час. Після відстеження часу ви можете зарахувати рахунки клієнтів, платитню співробітникам, переглядати звіти та багато іншого.

Недоліками даної системи є відстутність підтримки відмої системи кнтролю проетків TFS.

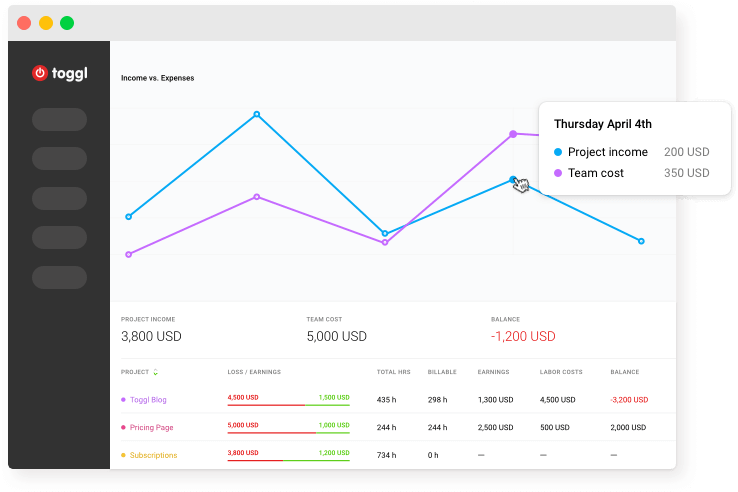
* Toggl

Toggl (рис. 2) - це програма відстеження часу, яка пропонує своїм веб-сайтам онлайн-послуги з відстеження та звітування за часом, а також мобільні та настільні додатки. Toggl відстежує час на основі завдань та проектів, або через інтерактивний таймер завдань або через ручне введення.

Дана система підтримує фінансову звітність проте взагалі не мітить інтеграції з найвідомішими системами управління проектами.



*Рис. 1. Система hubstaff.*



*Рис. 2. Система toggl.*

**1.3. Функціональність програмної системи**

*Таблиця 1*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Функція** | **Статус** | **Пріоритет** | **Трудоємність** | **Ризик** | **Стабільність** | **Цільова версія** |
| Витяг інформації про працівників з СКП | Включена | Критична | Високий рівень | Високий | Низька | 1.0 |
| Редагування інформації про працівників | Включена | Корисна | Високий рівень | Середній | Середня | 1.0 |
| Витяг інформації про проекти з СКП | Включена | Критична | Високий рівень | Високий | Низька | 1.0 |
| Витяг інформації про відпрацьовані години з СКП | Включена | Критична | Середній рівень | Середній | Середня | 1.0 |
| Обрахунок доходу проекту | Пропонована | Корисна | Високий рівень | Середній | Висока | 1.0 |
| Виставлення рахунків клієнтам | Пропонована | Важлива | Низький рівень | Низький | Висока | 1.1 |
| Обрахунок прибутку проекту | Пропонована | Важлива | Середній рівень | Низький | Висока | 1.1 |
| Визначення цінності працівника | Пропонована | Корисна | Середній рівень | Низький | Висока | 1.1 |
| Відправлення рахунків поштою | Пропонована | Корисна | Низький рівень | Низький | Висока | 1.2 |
| Логування як бухгалтер | Пропонована | Корисна | Середній рівень | Високий | Середня | 1.2 |
| Логування як керівник | Пропонована | Важлива | Середній рівень | Низький | Висока | 1.1 |
| Обрахунок прибутку компанії | Пропонована | Корисна | Низький рівень | Низький | Висока | 1.1 |
| Обрахунок доходу компанії | Пропонована | Корисна | Низький рівень | Низький | Висока | 1.1 |

**РОЗДІЛ 2. ВИБІР МЕТОДОЛОГІЇ ПРОЕКТУВАННЯ ТА ПЛАНУВАННЯ РОБІТ**

**2.1 Опис системних вимог згідно з методологією Rational Unified Process**

Rational Unified Process (RUP) являє собою продукт, спочатку розроблений Rational Software, яка була придбана компанією IBM в лютому 2003 року. Продукт містить у собі базу знань з гіперпосиланнями, та прикладами артефактів і докладні описи для різних видів діяльності. RUP входить в продукт IBM Rational Method Composer (RMC), який дозволяє налаштування процесу.

RUP заснований на наборі будівельних блоків, чи містить елементи, що описують те, що повинно бути зробленим, необхідні навички, та покрокове пояснення того, як досягаються конкретні цілі розробки. Основними будівельними блоками, чи елементами вмісту, є наступні:

* Ролі (хто). Роль визначає набір навичок, компетенції та відповідальності.
* Робочі продукти (що). Робочий продукт являє собою щось отримане з завдання, в тому числі всі документи і моделі, випущені під час роботи впродовж процесу.
* Завдання (як). Завдання описує одиницю роботи, яка доручена ролі, яка забезпечує значущий результат.

Використаємо RUP для даної системи.

1. **Зацікавлені особи прецеденту та їх вимоги:**

* Бухгалтер: вміння введення фінансового обліку.
* Учасник проекту: ведення звітності своєї роботи.

1. **Користувачі ПС, тобто основні актори цього прецеденту:**

Користувачами системи є працівники деякої компанії які мають різний доступ до різних частин системи:

* Бухгалтер, який оперує інформацією з усіх частин програми.
* Учасник проекту, який має доступ лише до системи звітування роботи.

1. **Передумови прецеденту (preconditions):**

* Бухгалтер повинен вміти оперувати даними у системі фінансового обліку проектів на основі даних системи контролю проектів .
* Учасник проекту повинен мати навички користування системою контролю проектів.

1. **Основний успішний сценарій:**
2. Учасниками проекту створюється деякий проект у системиі контролю проектів.
3. Бухглтер синхронізує дані з системою фінансового управління проектами.
4. Відбувається регулярна звітність своєї роботи учасниками проекту у цій системі.
5. На основі звітності бухгалтер виконує деякі операції у системі фінансового обліку проектів: виписування рахунків, ведення звітності і тд.
6. **Розширення основного сценарію або альтернативні потоки:**
7. За необхідності бухглатер чи керівник проекту може редагувати дані про проект чи відзвітовані години у системі фінансогового контролю проектів, що не являється остновним сценарієм і може призвести до неочікуваних результатів:

* *Система фінансового обліку повинна зафіксувати несинхронізованість даних і попередити про це користувача.*

1. Може бути вибрата така система контролю проектів чи версія підтримуваної системи що не сумісна з системою фінансового обліку проектів:

* *СФОП повинна показати конкретну помилку.*

1. **Пост-умови (postconditions):**

* *Дані звітності збережені в СКП.*
* *Дані успішно синхронізуються з СФОП.*

1. **Спеціальні СВ:**

* *Необхідно забезпечити максимальну надійність обробки всіх транзакцій.*
* *Зрозумілий та доступний інтерфейс для користувача інтерфейс.*

1. **Список необхідних технологій та додаткових пристроїв:**

* *СФОП має бути розроблена як WEB система.*
* *Сервер СФОП має бути розміщений у тій же локальній мережі компанії що і всі СКП.*

**2.2 Деталізація функцій та аналіз категорій користувачів**

СФОП передбачає зв'язок з необмеженою кількістю підтримуваних СКП.

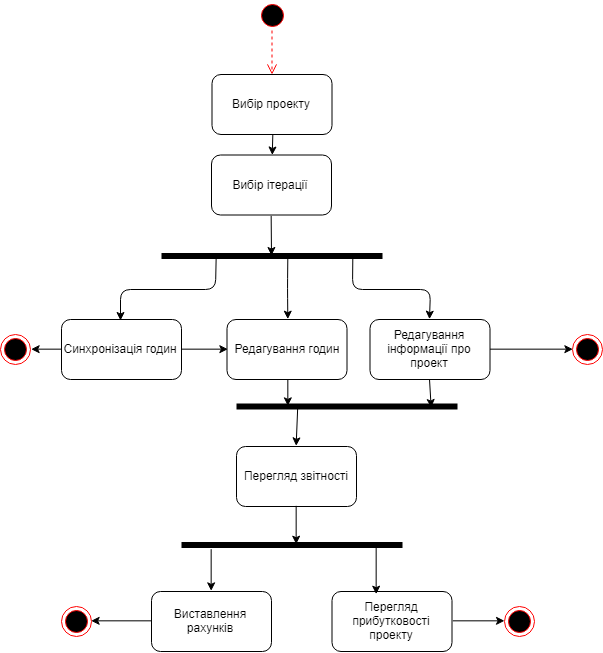
**Діаграма варіантів використання** (use case diagram) дає найбільш загальне уявлення щодо функціонального призначення системи, завдяки чому легко зрозуміти як будуть взаємодіяти користувачі з системою.



*Рис.3 Діаграма варіантів використання для проектованої системи*

Як ми бачимо, бухгалтер та керівник мають доступ до більшості функцій системи, а учасники проекту мають змоглу лише звітувати виконану роботу та вести інформацію про проект.

**Діаграми діяльності** (activity) є засобом опису поведінки у формі графа діяльності. Дана діаграма є компромісним способом ведення розробки – по суті, це проектування зверху-вниз в термінах і позначеннях UML. Таким чином, ми зможемо показати послідовність дій користувачів у нашій системі.

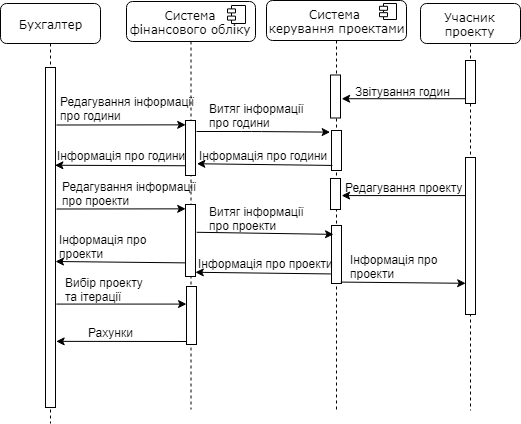
**

*Рис. 4 Діаграма діяльності системи*

Таким чином, ми бачимо можливі варіанти розвитку подій при роботі з системою.

**Діаграма послідовності** (sequence diagram). Реалізація окремого варіанта використання вимагає участі й взаємодії певних екземплярів акторів і класів. Найбільш відповідний інструмент для опису такої взаємодії – це діаграма послідовності.

Така діаграма показує послідовність виконання дій та запитів, а також дає інформацію про тривалість часу взаємодії компонентів системи.

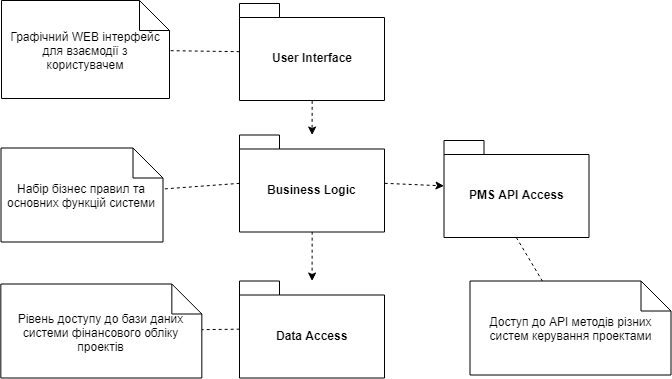


*Рис. 5 Діаграма послідовності системи.*

**Діаграма пакетів** (package diagram). В процесі об‘єктно-орієнтованої розробки ПЗ є потреба розподілити програмне рішення на компоненти. Це потрібно також для розподілення роботи між членами команди розробників. Засобом моделювання в UML, який дає змогу об’єднати окремі компоненти в групи є поняття пакету (package).

Така діаграма дасть нам уявлення про основні компоненти системи, їх взаємодію між собою, а також покаже їхню залежність один від одного.

З цієї діаграми(рис. 6) можна бачити, що система містить 3 основні пакети, які пов’язані між собою. Це дозволяє розбити створення програмного продукту на три окремі підзадачі, досягнення яких буде легшим і допоможе впорядкувати процес розробки програмного продукту та зробить його організованішим.



*Рис. 6. Діаграма пакетів системи.*

**2.3 Аналіз функціональних точок**

Проведемо аналіз функціональних точок для таких функцій:

* Витяг інформації про відпрацьовані години.
* Обрахунок прибутку проекту.

**Витяг інформації про відпрацьовані години –** дана функція містить 2 зовнішні входи: вибору проекту та вибору ітерації; 2 виходи: після виконання функції користувач може або поредагувати години або зберегти поточні години; 1 зовнішній запит: запит до системи контролю проектів в якій місятіться зазвітовані користувачами голини; 1 внутрішній та один завнішній логічні файли (файли баз даних СФОП та СКП).

*Таблиця 2*

**Значення параметрів функції “Витяг інформації про відпрацьовані години”**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Параметр | Легко | | Середньо | | Важко | |
| Кількість | Коефіцієнт | Кількість | Коефіцієнт | Кількість | Коефіцієнт |
| Зовнішні входи | 0 | 3 | 2 | 4 | 0 | 6 |
| Зовнішні виходи | 1 | 4 | 1 | 5 | 0 | 7 |
| Зовнішні запити | 0 | 3 | 0 | 4 | 1 | 6 |
| Внутрішні логічні файли | 1 | 7 | 0 | 10 | 0 | 15 |
| Зовнішні логічні файли | 0 | 5 | 1 | 7 | 0 | 10 |

**Обрахунок прибутку проекту –** дана функція містить 2 зовнішні входи: вибору проекту та вибору часу; 1 внутрішній логічний файл (баз даних СФОП).

*Таблиця 3*

**Значення параметрів функції “Обрахунок прибутку проекту”**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Параметр | Легко | | Середньо | | Важко | |
| Кількість | Коефіцієнт | Кількість | Коефіцієнт | Кількість | Коефіцієнт |
| Зовнішні входи | 1 | 3 | 1 | 4 | 0 | 6 |
| Зовнішні виходи | 0 | 4 | 0 | 5 | 0 | 7 |
| Зовнішні запити | 0 | 3 | 0 | 4 | 0 | 6 |
| Внутрішні логічні файли | 1 | 7 | 0 | 10 | 0 | 15 |
| Зовнішні логічні файли | 0 | 5 | 0 | 7 | 0 | 10 |

Таким чином розмір цих функцій буде наступним:

AF(1)=2\*4+1\*4+1\*5+1\*6+1\*7+1\*7=37

AF(2)=1\*3+1\*4+1\*7=14

Дані числа є попередньою оцінкою та потребує корекції, що здійснюється шляхом присвоєння ваги (від 0 до 5) кожній характеристиці проекту. Крім функціональних вимог на продукт накладаються загальносистемні вимоги, що обмежують розробників у виборі рішення і збільшують складність розробки. Для обліку цієї складності застосовується фактор вирівнювання (VAF). Значення фактора VAF залежить від 14 параметрів, які визначають системні характеристики продукту:

1. Обмін даними (0 – продукт являє собою автономний додаток; 5 – продукт обмінюється даними по більш, ніж одному телекомунікаційному протоколу).
2. Розподілена обробка даних (0 – продукт не використовує розподілену обробку; 5 – розподілена обробка даних виконується декількома компонентами системи).
3. Продуктивність (0 – користувацькі вимоги по продуктивності не встановлені; 5 – час відгуку є обмеженим.
4. Обмеження по апаратних ресурсах (0 – немає обмежень; 5 – продукт повинен функціонувати на певному процесорі і не може бути розподілений).
5. Транзакційне навантаження (0 – транзакцій не багато, без піків; 5 – число транзакцій велике і нерівномірне, потрібні спеціальні рішення та інструменти).
6. Інтенсивність взаємодії з користувачем (0 – всі транзакції обробляються в пакетному режимі; 5 – понад 30% транзакцій – інтерактивні).
7. Ергономіка (ефективність роботи кінцевих користувачів) (0 – немає спеціальних вимог; 5 – жорсткі вимоги щодо ефективності).
8. Інтенсивність зміни даних (ILF) користувачами (0 – не вимагаються; 5 – зміни інтенсивні, жорсткі вимоги по відновленню).
9. Складність обробки (0 – обробка мінімальна; 5 – вимоги безпеки, логічна і математична складність, багатопоточність).
10. Повторне використання (0 – не вимагається; 5 – продукт розробляється як стандартний багаторазовий компонент).
11. Зручність інсталяції (0 – немає вимог; 5 – установка і оновлення ПЗ проводиться автоматично).
12. Зручність адміністрування (0 – не вимагається; 5 – система автоматично самовідновлюється).
13. Чи є потреба в множинній кількості інсталяцій в різних умовах (0 – продукт має тільки 1 інсталяцію на єдиному процесорі; 5 – система є розподіленою і припускає установку на різне апаратне та програмне забезпечення).
14. Гнучкість (0 – не вимагається; 5 – гнучка система запитів і побудова довільних звітів, модель даних змінюється користувачем в інтерактивному режимі).

*Таблиця 5*

**Оцінка параметрів функції “Витяг інформації про відпрацьовані години”**

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | Значення | Характеристика | Значення |
| 1 | 5 | 8 | 2 |
| 2 | 1 | 9 | 2 |
| 3 | 0 | 10 | 3 |
| 4 | 0 | 11 | 5 |
| 5 | 1 | 12 | 0 |
| 6 | 1 | 13 | 0 |
| 7 | 1 | 14 | 3 |

*Таблиця 6*

**Оцінка параметрів функції “ Обрахунок прибутку проекту ”**

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | Значення | Характеристика | Значення |
| 1 | 1 | 8 | 1 |
| 2 | 0 | 9 | 1 |
| 3 | 3 | 10 | 0 |
| 4 | 0 | 11 | 0 |
| 5 | 0 | 12 | 0 |
| 6 | 0 | 13 | 0 |
| 7 | 1 | 14 | 0 |

Тепер проведемо уточнення функціонального розміру за формулою:

(v)AF = AF \* (0,65 + 0,01 \* TDI), де TDI – сумарний ефект характеристик.

Отримаємо:

(v)AF1=37\*(0,65+0,01\*24)=32,93

(v)AF2=14\*(0,65+0,01\*7)=10,08

Отже, таким чином було отримано уточнений функціональний розмір функцій проектованої системи, ці значення можуть бути перетворені в одиниці виміру обсягу ПЗ (кількість рядків коду, SLOC), або на їх основі може бути зроблена оцінка продуктивності по виконанню кількості FP в день (Performance factor), а на його підставі можна отримати оцінку трудомісткості проекту.

**2.4 Планування робіт в програмному засобі Gantt Project**

Початок роботи із системою полягає у створенні нового проекту та заповненні відповідних пунктів для розробки програмного забезпечення (кількість днів, тощо). Вибрати особливі налаштування днів (не ті які за замовчуванням) та додати святкові (рис.7).

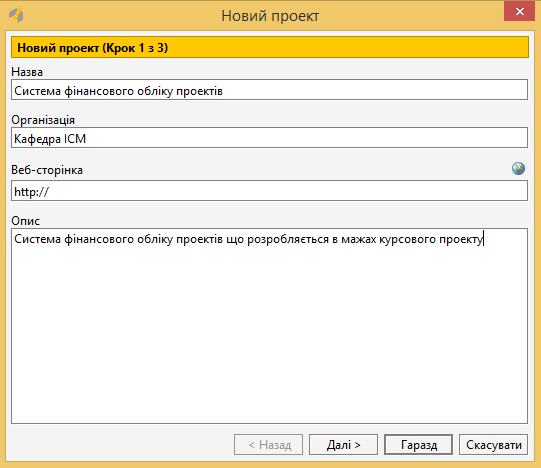
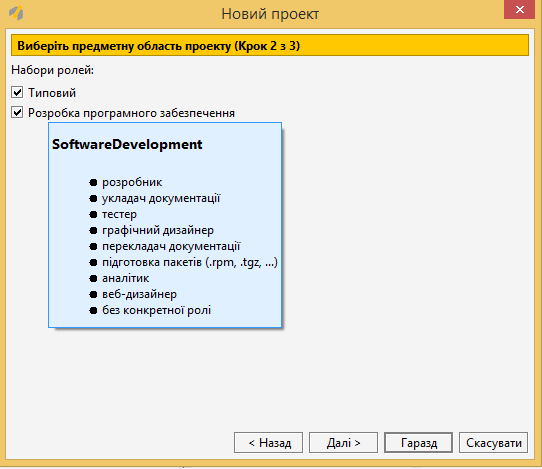
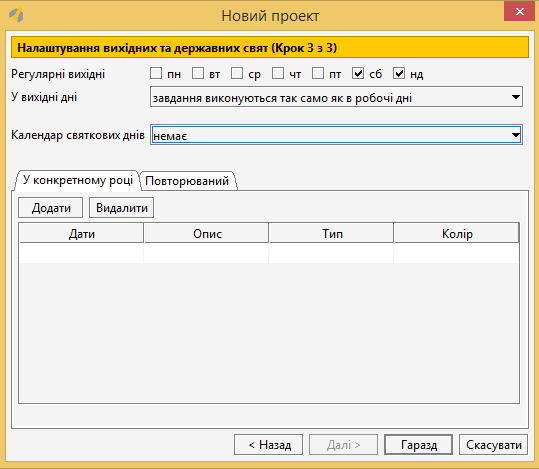
  

Рис. 7. *Заповнення даних про проект СФОП*

Наступним етапом є створення списку учасників проекту, шляхом виконання команди: Людина/Нова людина та вказанням її характеристик та ролі в проекті (Рис.8).

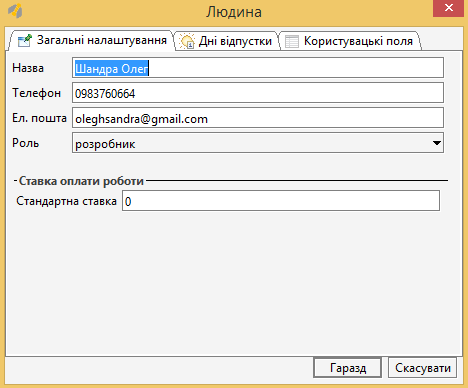


Рис. 8. *Заповнення даних нового учасника проекту*

У відповідності до основних етапів розробки здійснюємо вказання назви задачі та терміни її виконання на вкладці – Діаграма Ганта, налаштовуємо параметри кожної задачі (рис.9-12).

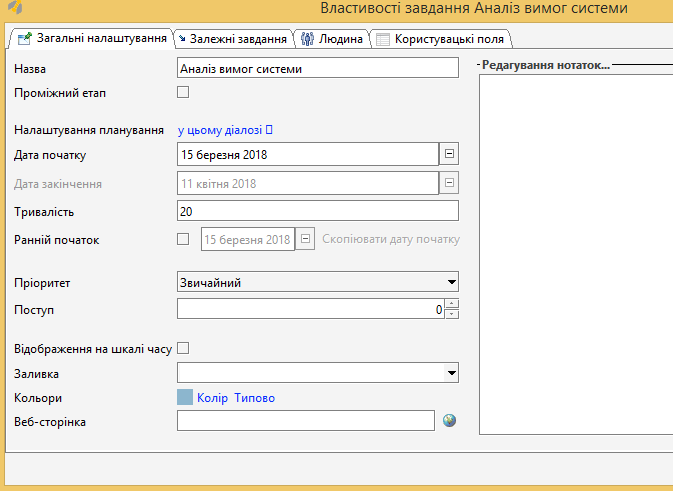


Рис. 9. *Властивості задачі Аналіз вимог до системи*

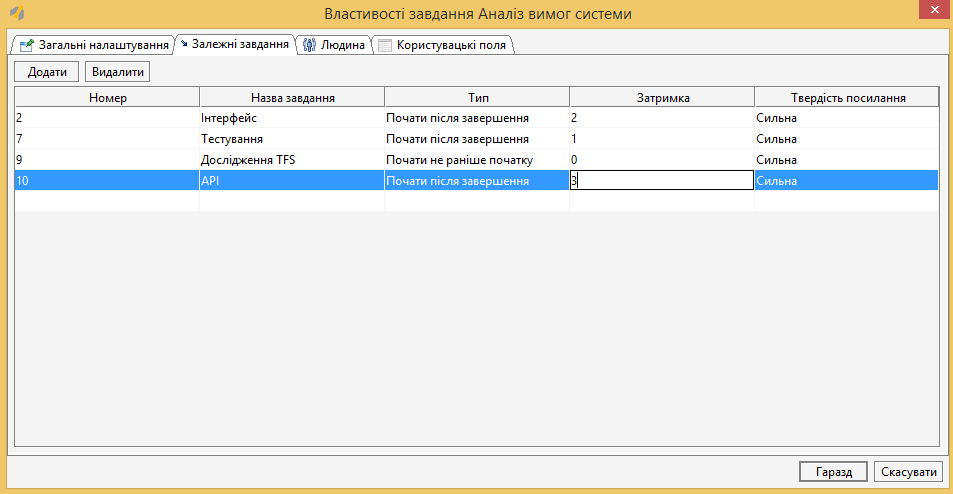


Рис.10. *Встановлення залежних задач*

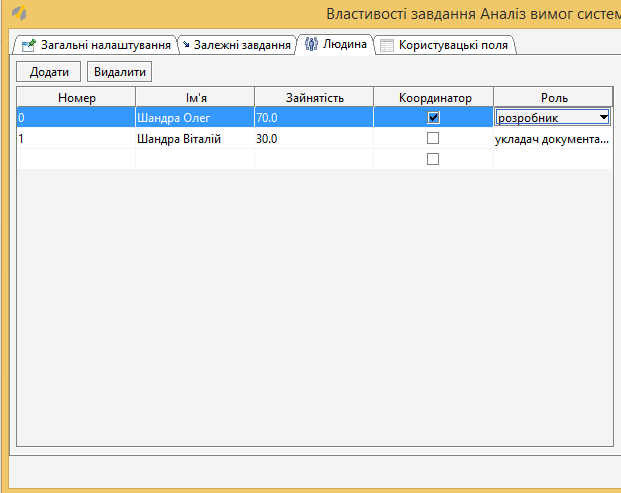


Рис. 11. *Прив’язка людей до задачі*

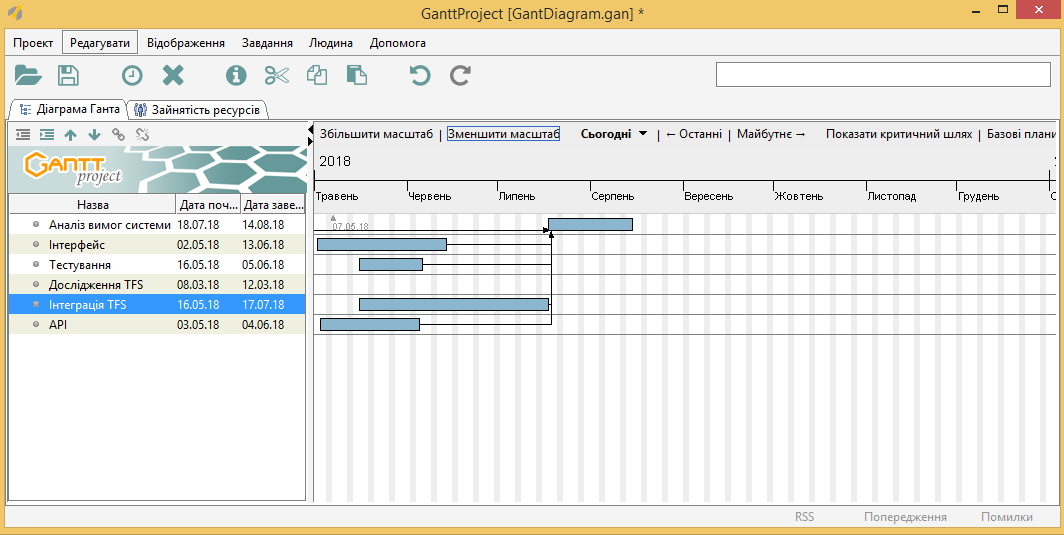


Рис. 12. *Перегляд залежних задач*

Прикладна система крім зазначеної інформації може вивести дані стосовно завантаженості виконавців, що надає засоби для «оптимального керування» (рис.13).

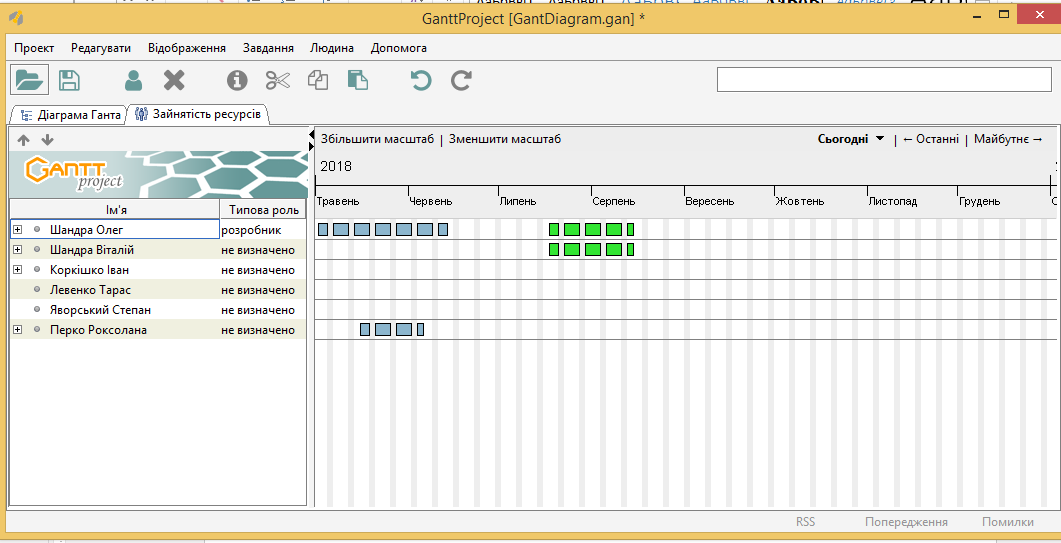


Рис. 13. *Перевірка балансу завантаження виконавців*

**РОЗДІЛ 3**

**ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ ТА ПРОЕКТУВАННЯ СИСТЕМИ**

***3.1. Вибір архітектурного та програмного рішення***

Щоб розробити cистему фінансового управління проектами було вибрано платформу .NET, оскільки вона має ряд готових наборів бібліотек класів для розробки потрібного нам інтернет ресурсу.

Ще однією перевагою .NET Framework є те, що вона надає можливість розробляти графічний інтерфейс користувача за допомогою технологій Windows Forms, WPF, Web Forms, ASP.NET MVC та ін. Для розробки інтерфейсу обрана технологія ASP.NET MVC разом із веб-мовами програмування HTML, CSS, JavaScript оскільки розроблена система повинна бути представлена у вигляді інтернет ресурсу.

Для полекшення створення графічного інтерфейсу користувача було використано веб бібліотеку Bootstrap.

Для запуску програми потрібен ПК з наступними мінімальними вимогами:

* Встановлена ОС WIndows 7, 8, 10, Server 2008, Server 2012.
* Частота процесора: 1 GHz.
* RAM: 512 мб.
* Дисковий простір (для ПЗ): 14 мб.
* Дисковий простір (для встановлення .Net Framework 4.5): 850 GB (32-bit ОС), 2 GB (64-bit ОС).

В якості інтегрованого середовища розробки обрана Visual Studio 2017. Вона надає можливість створювати проекти на основі технології .NET Framework та забезпечує користувача безліччю засобів для відлагодженя програмного засобу.

Основні бібліотеки .NET Framework, які використовуються для створення системи аналізу контенту:

* System.Net – простір імен забезпечує простий інтерфейс програмуванння для різних мережевих протоколів.
* System.Net.MVC - містить класи та інтерфейси, що підтримують платформу ASP.NET MVC для створення веб-додатків. У цей простір імен входять класи, що представляють контролери, фабрики контролерів, моделей, і багато іншого.
* System.Data.Entity – містить класи для доступу до основних компонентів технології Entity Framework.
  1. ***Реалізація програмного засобу.***

Для реалізації програмного засобу було створено такі основні сторінки:

* 1. Форма входу в систему.
  2. Проекти.
  3. Погодинні ставки працівників.
  4. Створення рахунку.
  5. Виставлені рахунки з можливістю їх генерування.

Початковою сторінкою яку бучить користувач є сторінка входу в систему (рис. 14).

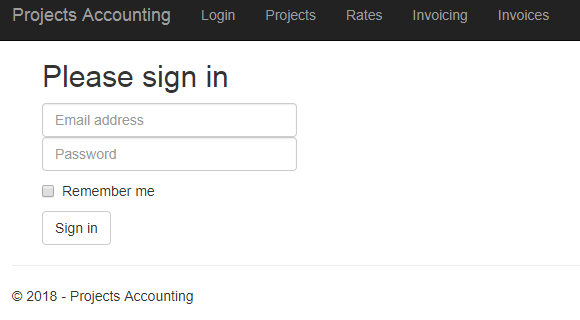


Рис. 14. *Форма входу в систему.*

Після логування користувач може синхронізувати інформацію про проекти з деякої підтримуваної системи керування проектами (рис. 15) із системою фінансового контролю проектів (рис. 16). Для цього йому потрібно натиснути відповідну кнопку, і у СФКП з’являться нові проекти. Далі користувачу необхідно заповнити інформацію про клієнтів, яка необхідна потім, для створення рахунку для деякого проекту.

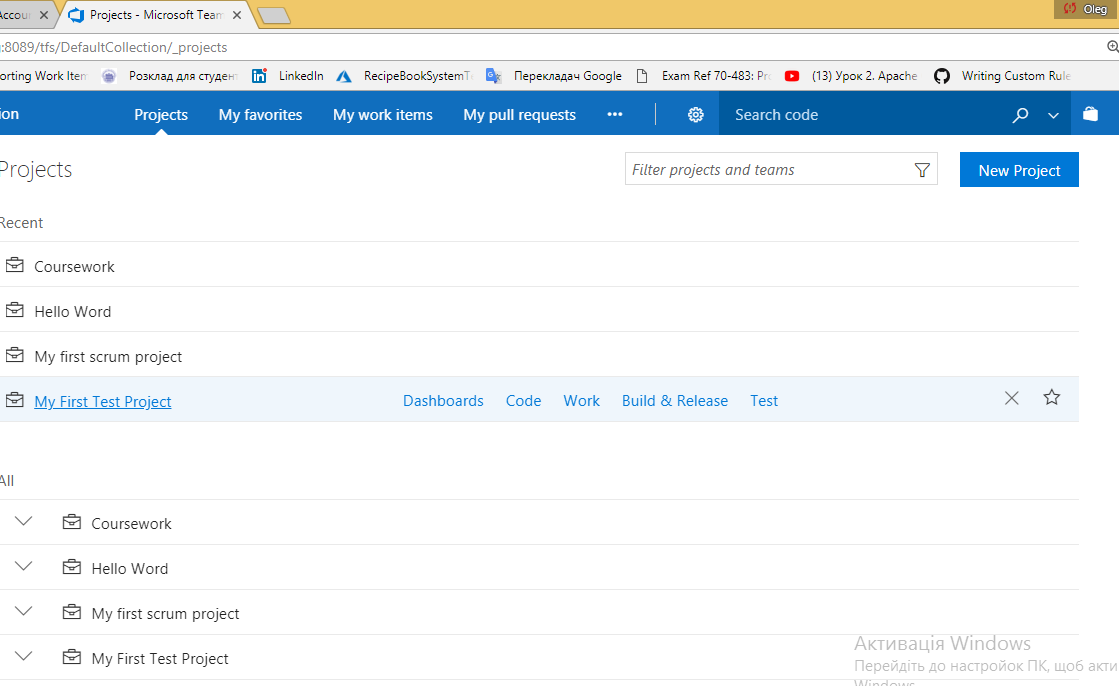


Рис. 15. *Проекти в tfs.*

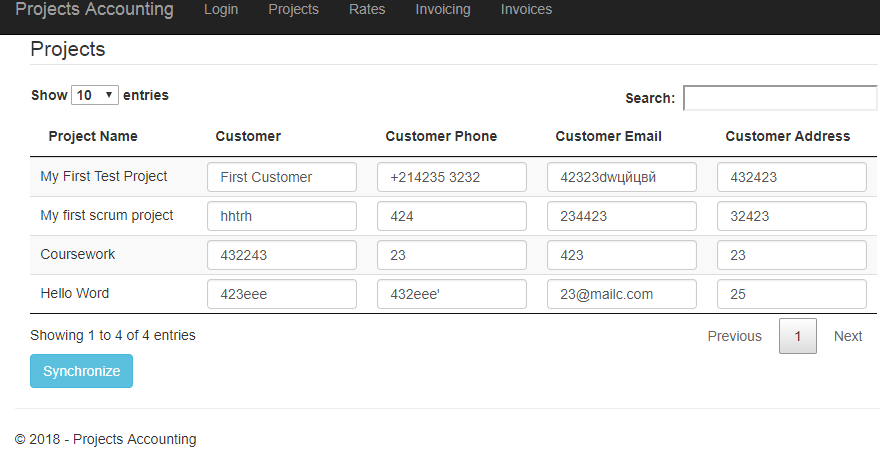


Рис. 16. Сторінка з п*роектами*

Далі користувач має змогу сихронізувати інформацію про працівників і редагувати інформацію про їхні внутрішні та зовнішні погодинні ставки (рис. 17). Під внутрішньою ставкою тут мається на увазі вартість працівника – обсяг гршей які йому виплачуються. Зовнішні ставки – обсяг грошей які платить клієнт за годину роботи працівника. На даній сторінці користувач має змогу редагувати інформацію про всіх працівників на всіх доступних проектах.

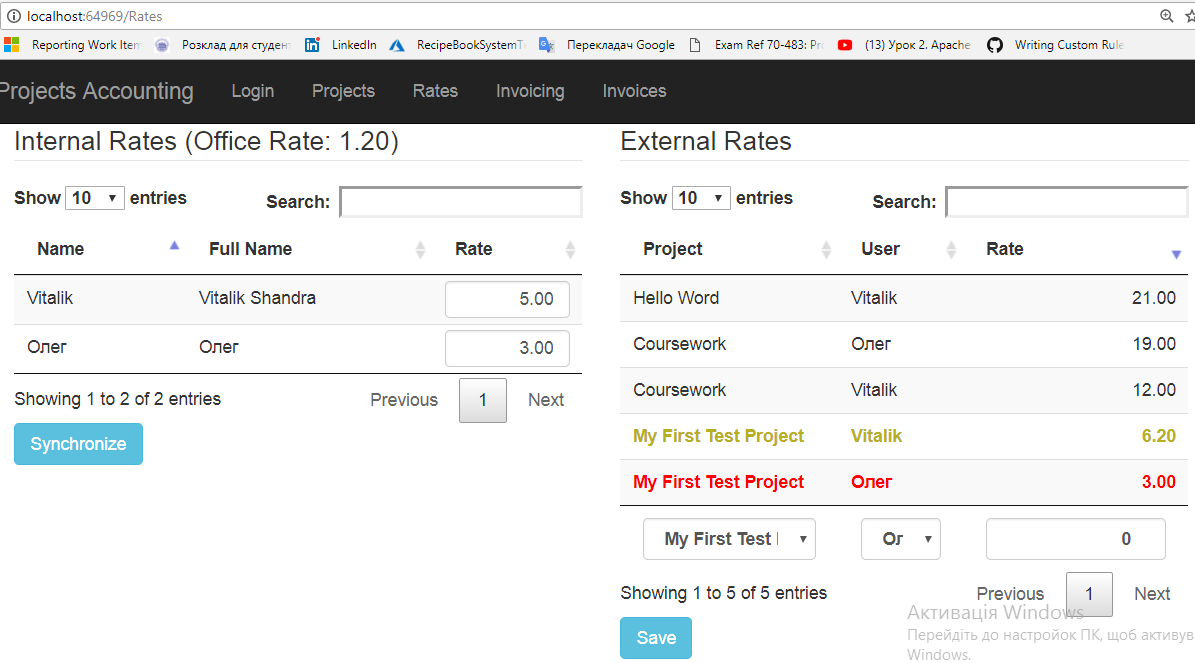


Рис. 17. *Сторінка працівників та їх погодинних ставок*

Наступник кроком є створання рахунку для клієнта (рис. 19). На цій сторінці коистува має заповнити інформацію про дату виставлення рахунку, написати відповідний коментар, вибрати проект та ітерацію і зберегти рахунок-фектуру. Після вибору всіх опцій корисувач бачить спосок усіх завдень що будуть включені в рахунок. Під списком завдань розуміється список інформації про назви завдань, працівників що їх роблять та кількість зазвітованих годин. Вся ця інформація береться із деякої системи контрою проектів (рис 18).

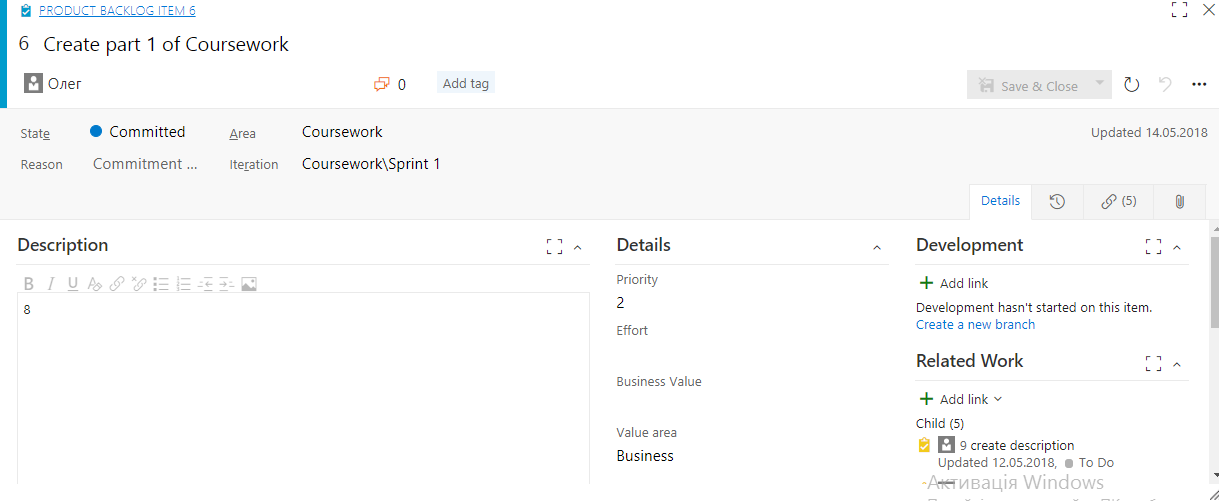


Рис. 18. *Завдання із tfs яке буде включено в рахунок.*

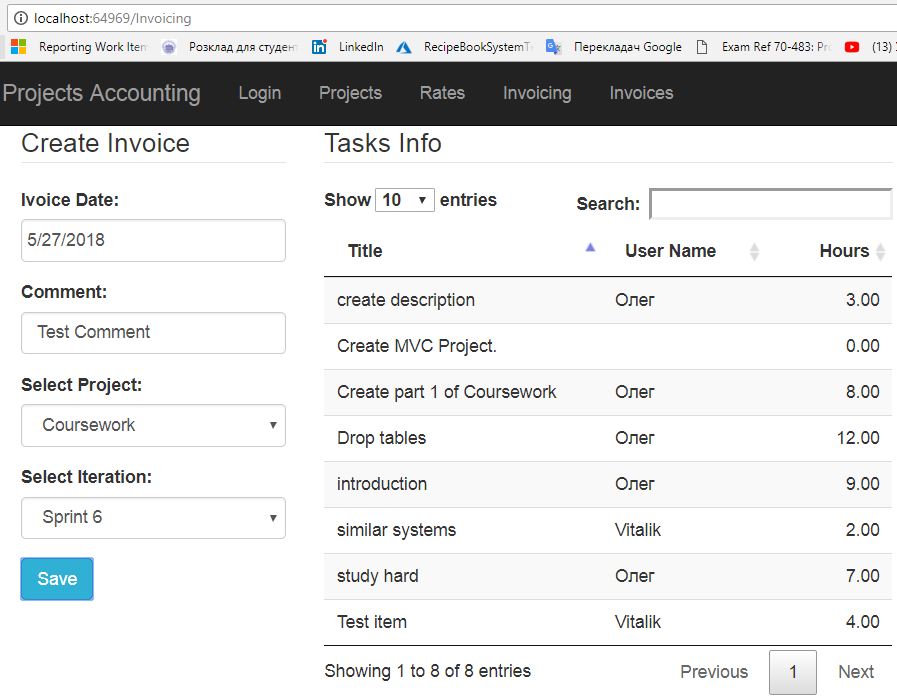


Рис. 19. *Сторінка створення рахунку-фактури.*

Останньою сторінкою доступною для корисувача є сторінка із споском виставлених рахунків-фактур та можливісю їх генерування (рис. 20). На даній сторінці користувач бачить споисок рахункці що були виставлені раніше, а саме номери рахунків, назви проектів, дати, коментарі, назви клієнтів та суму (не враховуючи податкову ставку).

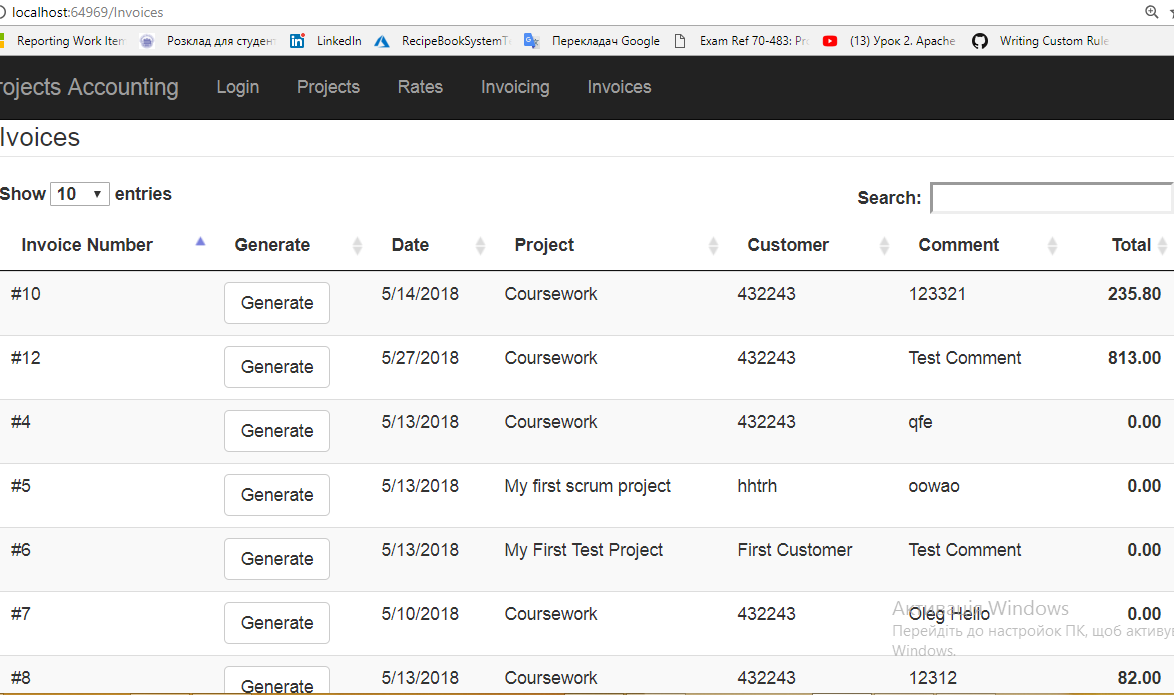


Рис. 20. *Сторінка з виставленими рахунквми*

Після генерування рахунку користувач бачить готовий виставлений рахунок який можна надруккувати (рис. 21). На даному рахунку показано всю необхідну інформацію про компанію що витавляє рахунок (інформація задана в базі даних, відповідної сторінки немає), інформацію про клієнта, дату, номер та коментарі рахунку а також інформацію про працівників – їхні савки, кількість відзвітованих годин та суму.

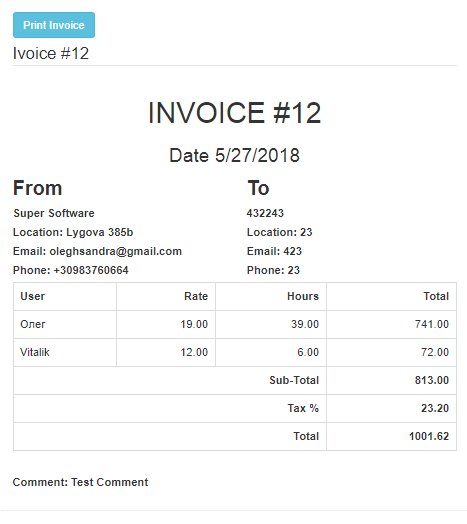


Рис. 21. *Виставлений рахунок.*

Згідно з патерном MVC, кожна вище згадана сторінка має відповідні розмітки сторінок (рис. 22), C# моделі доступу до бізнес логіки та бази даних і контроллери що поєднують ці два рівні.

Для прикладу, клас InvoicingController містить такі методи:

* Index(): збирає всю почтакову інформацію для ініціалізації сторінки з процесом створення рахунку та повартає відповідну сторінку.
* *LoadIterations(int projectId):* повертає список ітерацій для вибраного проекту.
* *LoadTasks(string iterationId):* повертає список завдень для вибраної ітерації деякого проекту.
* *SaveInvoice(InvoiceModel invoiceModel):* зберігає сформований рахунок-фактуру.

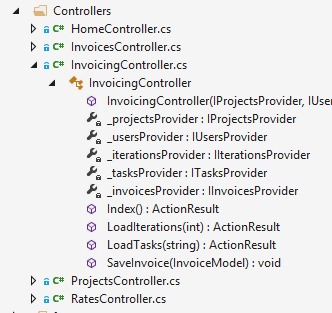
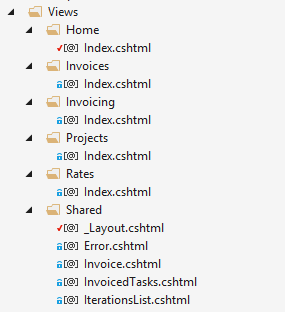
 

Рис. 22. Структура контроллерів та сторінок MVC проекту*.*

Крім MVC проекту у VS рішенні нашої програми містяться і інші бібліотеки класів (рис. 3).

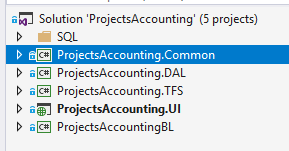


Рис. 23. Структура всього рішення*.*

Бібліотека класів ProjectsAccounting.Common містить перелік спільних для усіх проектів моделей.

Папка SQL мість набір скриптів що розгортають базу даних системи фінансового обліку проектів.

Бібліотека класів ProjectsAccounting.DAL містить класи доступу до бази даних, а саме набір репозиторіїв згідно з патерном Unit of Work.

Для прикладу, клас *ProjectRepository* містить такі методи:

* *Get(int projectId):* повертає модель проекту.
* *GetAll():* повертає список усіх проектів.
* *Insert(ProjectModel model):* зберігає проект в базі даних.
* *UpdateCustomerInfo(ProjectModel model):* зберігає нову інформацію про клієнта для деякого проекту.

Бібліотека класів ProjectsAccounting.TFS містить класи доступу до даних з Team Foundation Server.

Бібліотека класів ProjectsAccounting.BL містить класи з логікою роботи всієї системи. Вона поєднує три вищезгаданих бібліотеки. Даний логічний рівень має набір класів що дають нам повний доступ до основних операцій над абстрактими обєктами нашої системи.

Для прикладу, клас *UsersProvider* містить такі методи:

* *GetAll():* повертає список усіх користувачів наявних в системі фінансвого контрою проектів.
* *ChangeInternalRate(int userId, double internalRate):* даний метод змінює внутрішню погодинну ставку деякого працівника компанії.
* *Synhronize():* даний метод синхронізує наявних корисувачів із усіх підтримуваних систем керування проектів із систмемою фіннасового контролю, тобто відбувається створення користувачів в СФКП яких ще там не було, але вони вже є в деяких підтримуваних системах контролю проектів.

**3.3. Системне тестування програмного засобу**

Вершини керуючого графа (рис. 24):

* + 1. Вхід в систему;
    2. Перегляд інформації про проекти;
    3. Перегляд та редагування інформації про клієнтів;
    4. Перегляд та редагування інформації про працівників на проектах;
    5. Створення рахунку;
    6. Перегляд рахунків;
    7. Генерація рахунку;
    8. Вихід з системи.

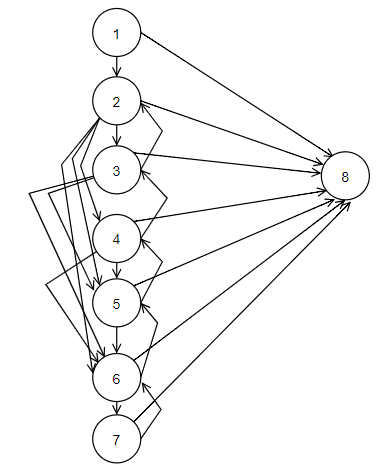


Рис. 24. Керуючий граф*.*

Як видно на графі, більшість частин програми є незалежними від інших, і користувач має доступ до них з будь якої іншої частини. Звісно, все починається з входу систему, і закінчуєтсья тим, що користувач виходить із системи.

Щоб найкраще протестувати систему фінансового управління проектами на ранньому етапі, нам слід обрати статичний підхід її тестування, тобто обрати такий шлях в керуючому графі, який за мінімальну кількість ітерацій покриє всі функціональні чатсини системи (рис. 25).

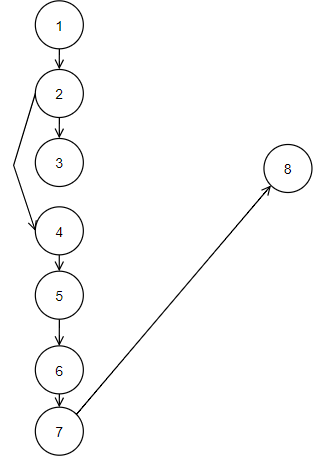


Рис. 25. Шлях що включає всі вершини*.*

Також потрібно проетестувати систему беручи до уваги інші шляхи у керуючому графі (рис. 26).

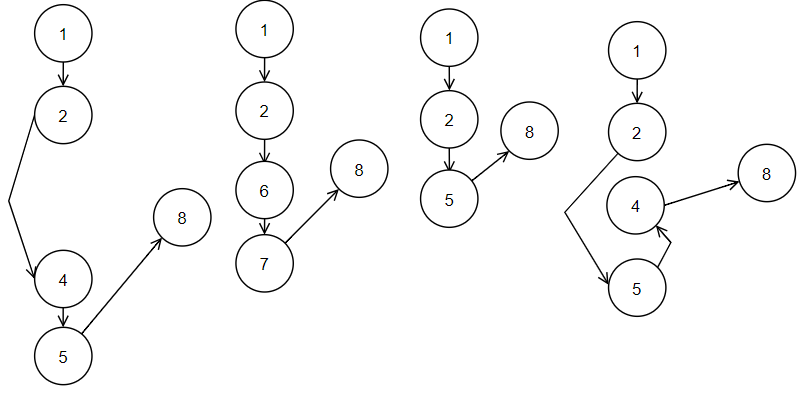


Рис. 25. Інші варіанти тестування системи*.*

**3.4. Оцінка якості створеного програмного забезпечення**

*Функціональна придатність* – сисстема фінансового обліку проектів реалізує всі потрібні функції що були описані як такі що треба реалізувати в першій версії програми, тому програму можна назвати функціонально придатною.

*Оцінка коректності програмного засобу* – наш програмний продукт при деяких вхідних даних видає набір очікуваних вихідних резлутатів, тому можна сказати що вона її робота є коректною. Також це було доведено виконавши тестування програмного продукуту.

*Оцінка здатності до взаємодії –* програма має зручний графіний інтерфейс, тому користувачі мають змогу інтуєтивно зрозуміло робити потрібні їм діїі у нашій системі. Всі частини системи фіннансвого обліку проектів коректно взаємодіють (від бази даних до графічного інтерфейсу користувача). Також наша система успішно взаємодіє з іншоими системами – системами контролю проектів, такими як TFS.

*Оцінка надійності* – програма є достатньо надійною, проте є аспекти які можуть порушити її надійність, такі як зміна специфікації в роботі підтримуваних систем керування проектами (зміна їхньої функціональності, API та інше).

*Супровід* – для створення програми було використано систему контролю версій GIT що дає змогу переключатись між різними версіями програми та описано весь необхідний обсях ресурсів які необхідно для її розгортання.

*Оцінка мобільності* – мобільність системи фінансового обліку проектів обмежена тим, що програма прицює лише на ОС Windows. Програма коректно працює на даній ОС та не вимагає особливих зусиль для її встановлення.

Отже, реалізувавши, ротестувавши та давши оцінку створеному програмному продукту, слід зазначити що програма повністю відповідає вимогам, проста у використанні, містить набір усіх потрібних функціїй для її коректного функціонування та коректно взаємодіє з іншими системами.

**ВИСНОВКИ**

Метою виконання курсової роботи було проектування та програмна реалізація системи фінансовго обліку проектів. Зважаючи на сучасний розвиток індустрії інформаційних технологій, розробка такої системи є дуже актуальною.

Аналізуючи літературні джерела були визначені основні вимоги та функції, які ставилися перед розроблюваною системою. Всі функції успішно реалізовані і протестовані в завершеній програмі.

Для тестування програми був проведений аналіз взаємодії системи з системою керування проектів tfs. Дані що були занесені в tfs (інформацій про проекти, користувачів, завдання, ітерації, відзвітовані години) успішно синхронізувались з даними системи фінансвого контролю проектів для ведення відповідної звітності та свторення рахунків.

Для програми був розроблений зручний графічний інтерфейс за допомогою технології ASP.NET MVC та веб мов програмування HTM, CSS та JavaScript. Вони забезпечують зручну взаємодію користувача з системою.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Бойчик І.М. Економіка підприємства: підручник. / І.М.Бойчик. – К.: Кондор -Видавництво, 2016. – 378 с.
2. [Крачтен Ф.](http://padabum.com/search.php?author=%D0%9A%D1%80%D0%B0%D1%87%D1%82%D0%B5%D0%BD%20%D0%A4.) Введение в Rational Unified Process; пер с англ.: Издательский дом Вильямс, 2002 р; 240 ст.
3. Троелсен Э. Язык программирования С# 2005 и платформа .net 2.0, 3-е издание/ Эндрю Троелсен. - Москва, Санкт-Петербург, Киев: ООО "И.Д.Вильямс", 2007. - 1168 с.
4. Шилдт Г. C# 4.0. Полное руководство / Герберт Шилдт. – М.: Вильямс, 2011. – 1056 с
5. Модели жизненного цикла программного обеспечения. [Електронний ресурс]. Режим доступу: http://swebok.sorlik.ru/software\_lifecycle\_models.html.
6. Проектный треугольник. [Електронний ресурс]. Режим доступу: http://office.microsoft.com/ru-ru/project-help/HA010351692.aspx.
7. Терехов А.М. Вступ до технологій програмування. [Електронний ресурс]. Режим доступу: http://www.intuit.ru/department/se/introprogteach/.

***Додаток A.***

CREATE DATABASE ProjectsAccounting

GO

USE ProjectsAccounting

GO

CREATE TABLE Users (

UserId int identity(1, 1) primary key,

UserName varchar(255),

FullName varchar(255),

PMCID varchar(255),

InternalRate float

);

CREATE TABLE Projects (

ProjectId int identity(1, 1) primary key,

ProjectName varchar(255),

CustomerName varchar(255),

CustomerAddress varchar(255),

CustomerEmail varchar(255),

CustomerPhone varchar(255),

PMCID varchar(255)

);

CREATE TABLE ProjectRates (

ProjectRateId int identity(1, 1) primary key,

ProjectId int,

UserId int,

ExternalRate float,

FOREIGN KEY (UserId) REFERENCES Users(UserId),

FOREIGN KEY (ProjectId) REFERENCES Projects(ProjectId)

);

CREATE TABLE CompanyInfo (

CompanyId int identity(1, 1) primary key,

CompanyName varchar(255),

MainAccpuntantName varchar(255),

OwnerName varchar(255),

LocationAddress varchar(255),

Fax varchar(255),

Phone varchar(255),

TaxRate float,

OfficeRate float

);

CREATE TABLE Invoices (

InvoiceId int identity(1, 1) primary key,

ProjectId int,

InvoiceDate DateTime,

Notes varchar(255),

CompanyName varchar(255),

MainAccpuntantName varchar(255),

OwnerName varchar(255),

LocationAddress varchar(255),

Fax varchar(255),

Phone varchar(255),

TaxRate float,

OfficeRate float,

CustomerName varchar(255),

CustomerAddress varchar(255),

CustomerEmail varchar(255),

CustomerPhone varchar(255),

FOREIGN KEY (ProjectId) REFERENCES Projects(ProjectId)

);

CREATE TABLE InvoicedTasks (

InvoicedTaskId int identity(1, 1) primary key,

InvoiceId int,

UserId int,

UserExternalRate float,

UserInternalRate float,

ReportedHours float,

TaskName varchar(255)

FOREIGN KEY (InvoiceId) REFERENCES Invoices(InvoiceId)

);

INSERT INTO [dbo].[CompanyInfo]

([CompanyName]

,[MainAccpuntantName]

,[OwnerName]

,[LocationAddress]

,[Fax]

,[Phone]

,[TaxRate]

,[OfficeRate])

VALUES

('Super Software'

,'Shandra Oleg'

,'Shandra Oleg'

,'Lygova 385b'

,'oleghsandra@gmail.com'

,'+30983760664'

,23.2

,1.2)

GO

***Додаток B.***

namespace ProjectsAccounting.Common.Models

{

public partial class CompanyInfoModel

{

public int CompanyId { get; set; }

public string CompanyName { get; set; }

public string MainAccpuntantName { get; set; }

public string OwnerName { get; set; }

public string LocationAddress { get; set; }

public string Fax { get; set; }

public string Phone { get; set; }

public double TaxRate { get; set; }

public double OfficeRate { get; set; }

}

}

namespace ProjectsAccounting.Common.Models

{

public class InvoicedTaskModel

{

public int InvoicedTaskId { get; set; }

public int InvoiceId { get; set; }

public int UserId { get; set; }

public double UserExternalRate { get; set; }

public double UserInternalRate { get; set; }

public double ReportedHours { get; set; }

public string TaskName { get; set; }

public virtual InvoiceModel Invoice { get; set; }

public virtual UserModel User { get; set; }

}

}

using System;

using System.Collections.Generic;

namespace ProjectsAccounting.Common.Models

{

public partial class InvoiceModel

{

public int InvoiceId { get; set; }

public int ProjectId { get; set; }

public int IterationId { get; set; }

public DateTime InvoiceDate { get; set; }

public string Notes { get; set; }

public string CompanyName { get; set; }

public string MainAccpuntantName { get; set; }

public string OwnerName { get; set; }

public string LocationAddress { get; set; }

public string Fax { get; set; }

public string Phone { get; set; }

public double TaxRate { get; set; }

public double OfficeRate { get; set; }

public string CustomerName { get; set; }

public string CustomerAddress { get; set; }

public string CustomerEmail { get; set; }

public string CustomerPhone { get; set; }

public List<InvoicedTaskModel> InvoicedTasks { get; set; }

public ProjectModel Project { get; set; }

}

}

namespace ProjectsAccounting.Common.Models

{

public class IterationModel

{

public int IterationId { get; set; }

public string Name { get; set; }

public string PMCID { get; set; }

public int ProjectId { get; set; }

public ProjectModel Project { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ProjectsAccounting.Common.Models

{

public class ProjectRateModel

{

public int ProjectRateId { get; set; }

public int ProjectId { get; set; }

public int UserId { get; set; }

public double ExternalRate { get; set; }

public ProjectModel Project { get; set; }

public UserModel User { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ProjectsAccounting.Common.Models

{

public class ProjectModel

{

public int ProjectId { get; set; }

public string ProjectName { get; set; }

public string CustomerName { get; set; }

public string CustomerAddress { get; set; }

public string CustomerEmail { get; set; }

public string CustomerPhone { get; set; }

public string PMCID { get; set; }

public virtual List<ProjectRateModel> ProjectRates { get; set; }

}

}

namespace ProjectsAccounting.Common.Models

{

public class TaskModel

{

public int TaskId { get; set; }

public string Name { get; set; }

public string PMCID { get; set; }

public string AssignedToUserPMCId { get; set; }

public string Description { get; set; }

public double HoursReported { get; set; }

public UserModel User { get; set; }

}

}

using System.Collections.Generic;

namespace ProjectsAccounting.Common.Models

{

public class UserModel

{

public int UserId { get; set; }

public string UserName { get; set; }

public string FullName { get; set; }

public string PMCID { get; set; }

public double InternalRate { get; set; }

public virtual List<ProjectModel> Projects { get; set; }

public virtual List<ProjectRateModel> ProjectsRates { get; set; }

}

}

using System;

using System.Data.Entity;

namespace ProjectsAccounting.DAL.Repositories

{

public class RepositoryBase<T> : IDisposable where T : DbContext, new()

{

public RepositoryBase()

{

this.Context = new T();

}

public T Context { get; private set; }

public void Save()

{

Context.SaveChanges();

}

private bool disposed = false;

/// <summary>

/// Override dispose method

/// </summary>

protected virtual void Dispose(bool disposing)

{

if (!this.disposed)

{

if (disposing)

{

Context.Dispose();

}

}

this.disposed = true;

}

public void Dispose()

{

Dispose(true);

GC.SuppressFinalize(this);

}

}

}

using ProjectsAccounting.Common.Models;

using ProjectsAccounting.DAL.DB;

using ProjectsAccounting.DAL.Mappers;

using System.Linq;

namespace ProjectsAccounting.DAL.Repositories

{

public class CompanyInfoRepository : RepositoryBase<ProjectsAccountingEntities>, ICompanyInfoRepository

{

/// <summary>

/// Get all company info

/// </summary>

public CompanyInfoModel GetCompanyInfo()

{

return this.Context.CompanyInfo.ToList()

.Select(i => CompanyInfoMapper.ToCompanyInfoModel(i)).FirstOrDefault();

}

}

}

using ProjectsAccounting.Common.Models;

using ProjectsAccounting.DAL.DB;

using ProjectsAccounting.DAL.Mappers;

using System.Collections.Generic;

using System.Linq;

namespace ProjectsAccounting.DAL.Repositories

{

public class InvoicedTasksRepository : RepositoryBase<ProjectsAccountingEntities>, IInvoicedTasksRepository

{

/// <summary>

/// Get all invoiced tasks

/// </summary>

/// <param name="invoiceId"></param>

/// <returns></returns>

public List<InvoicedTaskModel> GetAll(int invoiceId)

{

return this.Context.InvoicedTasks.Where(i => i.InvoiceId == invoiceId).ToList()

.Select(i => InvoicedTaskMapper.ToInvoicedTaskModel(i)).ToList();

}

/// <summary>

/// Insert invoiced task

/// </summary>

public void Insert(InvoicedTaskModel model)

{

var invoiceTask = InvoicedTaskMapper.ToDBInvoicedTask(model);

this.Context.InvoicedTasks.Add(invoiceTask);

this.Save();

}

/// <summary>

/// Insert invoiced tasks range

/// </summary>

public void InsertRange(List<InvoicedTaskModel> models)

{

var invoiceTasks = models.Select(t => InvoicedTaskMapper.ToDBInvoicedTask(t));

this.Context.InvoicedTasks.AddRange(invoiceTasks);

this.Save();

}

}

}

using ProjectsAccounting.Common.Models;

using ProjectsAccounting.DAL.DB;

using ProjectsAccounting.DAL.Mappers;

using System.Collections.Generic;

using System.Linq;

namespace ProjectsAccounting.DAL.Repositories

{

public class InvoicesRepository : RepositoryBase<ProjectsAccountingEntities>, IInvoicesRepository

{

public List<InvoiceModel> GetAll()

{

return this.Context.Invoices.OrderByDescending(i => i.InvoiceDate)

.Select(i => new { invoice = i, project = i.Projects, tasks = i.InvoicedTasks }).ToList()

.Select(i => InvoiceMapper.ToInvoiceModel(i.invoice, i.project, i.tasks)).ToList();

}

public InvoiceModel Get(int invoiceId)

{

var invoice = this.Context.Invoices.FirstOrDefault(i => i.InvoiceId == invoiceId);

var result = InvoiceMapper.ToInvoiceModel(invoice, invoice.Projects, invoice.InvoicedTasks);

return result;

}

public void Insert(InvoiceModel model)

{

var invoice = InvoiceMapper.ToDBInvoice(model);

this.Context.Invoices.Add(invoice);

this.Save();

model.InvoiceId = invoice.InvoiceId;

}

}

}

using ProjectsAccounting.Common.Models;

using ProjectsAccounting.DAL.DB;

using ProjectsAccounting.DAL.Mappers;

using System.Collections.Generic;

using System.Linq;

namespace ProjectsAccounting.DAL.Repositories

{

public class ProjectRateRepository : RepositoryBase<ProjectsAccountingEntities>, IProjectRateRepository

{

public List<ProjectRateModel> GetAll()

{

return this.Context.ProjectRates.Select(r => new { rate = r, project = r.Projects, user = r.Users }).ToList()

.Select(p => new ProjectRateModel()

{

ProjectId = p.rate.ProjectId ?? 0,

UserId = p.rate.UserId ?? 0,

ExternalRate = p.rate.ExternalRate ?? 0,

Project = ProjectMapper.ToProjectModel(p.project),

User = UserMapper.ToUserModel(p.user),

}).ToList();

}

public List<ProjectRateModel> GetForUser(int userId)

{

return this.Context.ProjectRates.Where(r => r.UserId == userId).ToList()

.Select(p => ProjectRateMapper.ToProjectRateModel(p)).ToList();

}

public List<ProjectRateModel> GetForProject(int projectId)

{

return this.Context.ProjectRates.Where(r => r.ProjectId == projectId).ToList()

.Select(p => ProjectRateMapper.ToProjectRateModel(p)).ToList();

}

public ProjectRateModel GetForUserAndProject(int userId, int projectId)

{

return this.Context.ProjectRates.Where(r => r.ProjectId == projectId && r.UserId == userId).ToList()

.Select(p => ProjectRateMapper.ToProjectRateModel(p)).FirstOrDefault();

}

public void Insert(ProjectRateModel model)

{

var rate = ProjectRateMapper.ToDBProjectRate(model);

this.Context.ProjectRates.Add(rate);

this.Save();

}

public void Update(ProjectRateModel model)

{

var rate = this.Context.ProjectRates.FirstOrDefault(r => r.ProjectRateId == model.ProjectRateId);

rate.ExternalRate = model.ExternalRate;

this.Save();

}

}

}

using ProjectsAccounting.Common;

using ProjectsAccounting.Common.Models;

using ProjectsAccounting.DAL.DB;

using ProjectsAccounting.DAL.Mappers;

using System.Collections.Generic;

using System.Linq;

namespace ProjectsAccounting.DAL.Repositories

{

public class ProjectRepository : RepositoryBase<ProjectsAccountingEntities>, IProjectRepository

{

public ProjectModel Get(int projectId)

{

return ProjectMapper.ToProjectModel(this.Context.Projects.FirstOrDefault(p => p.ProjectId == projectId));

}

public List<ProjectModel> GetAll()

{

return this.Context.Projects.ToList()

.Select(p => ProjectMapper.ToProjectModel(p)).ToList();

}

public void Insert(ProjectModel model)

{

var project = ProjectMapper.ToDBProject(model);

this.Context.Projects.Add(project);

this.Save();

}

public void UpdateCustomerInfo(ProjectModel model)

{

var project = this.Context.Projects.FirstOrDefault(p => p.ProjectId == model.ProjectId);

project.CustomerName = model.CustomerName;

project.CustomerAddress = model.CustomerAddress;

project.CustomerEmail = model.CustomerEmail;

project.CustomerPhone = model.CustomerPhone;

this.Save();

}

}

}

using ProjectsAccounting.Common.Models;

using ProjectsAccounting.DAL.DB;

using ProjectsAccounting.DAL.Mappers;

using System.Collections.Generic;

using System.Linq;

namespace ProjectsAccounting.DAL.Repositories

{

public class UsersRepository : RepositoryBase<ProjectsAccountingEntities>, IUsersRepository

{

public List<UserModel> GetAll()

{

return this.Context.Users.ToList()

.Select(p => UserMapper.ToUserModel(p)).ToList();

}

public void Insert(UserModel model)

{

var rate = UserMapper.ToDBUser(model);

this.Context.Users.Add(rate);

this.Save();

}

public void ChangeInternalRate(int userId, double internalRate)

{

var user = this.Context.Users.FirstOrDefault(u => u.UserId == userId);

user.InternalRate = internalRate;

this.Save();

}

}

}

using ProjectsAccounting.TFS.TFSDB;

using System.Collections.Generic;

using System.Linq;

using ProjectsAccounting.Common.Models;

using System;

namespace ProjectsAccounting.TFS.Repositories

{

public class TFSIterationsRepository : RepositoryBase<Tfs\_DefaultCollectionEntities>, ITFSIterationsRepository

{

/// <summary>

/// Get all iterations for project

/// </summary>

public List<IterationModel> GetAll(string projectTFCid)

{

var projectId = Int32.Parse(projectTFCid.Trim());

var result = from i in this.Context.tbl\_Iteration

join p in this.Context.tbl\_Project on i.ProjectUri equals p.ProjectUri

where projectId == p.ProjectId && i.Iteration.Contains("\\")

select new { i };

return result.ToList().Select(i => new IterationModel()

{

PMCID = i.i.IterationId.ToString(),

Name = i.i.Iteration.Split('\\')[1]

}).ToList();

}

}

}

using ProjectsAccounting.TFS.TFSDB;

using System.Collections.Generic;

using System.Linq;

using ProjectsAccounting.Common.Models;

using ProjectsAccounting.DAL.Mappers;

namespace ProjectsAccounting.TFS.Repositories

{

public class TFSProjectsRepository : RepositoryBase<Tfs\_DefaultCollectionEntities>, ITFSProjectsRepository

{

/// <summary>

/// Get all projects

/// </summary>

/// <returns></returns>

public List<ProjectModel> GetAll()

{

return Context.tbl\_Project.ToList().Select(p => ProjectMapper.ToProjectModel(p)).ToList();

}

}

}

using ProjectsAccounting.TFS.TFSDB;

using System.Collections.Generic;

using System.Linq;

using ProjectsAccounting.Common.Models;

using System;

namespace ProjectsAccounting.TFS.Repositories

{

public class TFSTasksRepository : RepositoryBase<Tfs\_DefaultCollectionEntities>, ITFSTasksRepository

{

/// <summary>

/// Get all tasks for iteration

/// </summary>

public List<TaskModel> GetAll(string iterationId)

{

var iterationIdNumber = Convert.ToInt32(iterationId);

var result = from task in this.Context.vw\_denorm\_WorkItemCoreLatest

let reportedField = this.Context.WorkItemLongTexts

.Where(f => f.ID == task.Id && f.FldID == 52)

.OrderByDescending(f => f.AddedDate)

.FirstOrDefault()

let nameField = this.Context.WorkItemLongTexts

.Where(f => f.ID == task.Id && f.FldID == 1)

.OrderByDescending(f => f.AddedDate)

.FirstOrDefault()

where task.IterationId == iterationIdNumber

select new { task, reportedField, nameField };

return result.ToList().Select(i => new TaskModel()

{

PMCID = i.task.Id.ToString(),

AssignedToUserPMCId = i.task.System\_AssignedTo\_TeamFoundationId == null

? "" : i.task.System\_AssignedTo\_TeamFoundationId.ToString(),

Name = i.nameField == null ? "" : i.nameField.Words,

Description = i.reportedField == null ? "" : i.reportedField.Words,

HoursReported = i.reportedField == null ? 0.0 : (double.TryParse(i.reportedField.Words, out double value) ? value : 0)

}).ToList();

}

}

}

using ProjectsAccounting.TFS.TFSDB;

using System.Collections.Generic;

using System.Linq;

using ProjectsAccounting.Common.Models;

namespace ProjectsAccounting.TFS.Repositories

{

public class TFSUsersRepository : RepositoryBase<Tfs\_DefaultCollectionEntities>, ITFSUsersRepository

{

/// <summary>

/// Get all users

/// </summary>

public List<UserModel> GetAll()

{

var result = from grp in this.Context.ADObjects

join om in this.Context.ADObjectMemberships on grp.ObjectSID equals om.ObjectSID

join member in this.Context.ADObjects on om.MemberObjectSID equals member.ObjectSID

where (grp.SamAccountName == "Team Foundation Administrators" && member.ObjectCategory == (byte)2)

select new { member };

return result.ToList().Select(u => new UserModel()

{

PMCID = u.member.TeamFoundationId.ToString(),

FullName = u.member.DisplayName,

UserName = u.member.SamAccountName

}).ToList();

}

}

}

using System.Collections.Generic;

using System.Linq;

using ProjectsAccounting.DAL.Repositories;

using ProjectsAccounting.TFS.Repositories;

using ProjectsAccounting.Common.Models;

using ProjectsAccountingBL.Providers.Abstract;

using System.Transactions;

using System;

namespace ProjectAccountingBL.Providers.Implementation

{

public class InvoicesProvider : IInvoicesProvider

{

public InvoicesProvider(

IInvoicedTasksRepository invoicedTasksRepository,

IInvoicesRepository invoicesRepository,

IUsersRepository usersRepository,

IProjectRateRepository projectRateRepository,

ICompanyInfoRepository companyInfoRepository,

IProjectRepository projectRepository,

ITFSTasksRepository tfsTasksRepository)

{

this.\_invoicedTasksRepository = invoicedTasksRepository;

this.\_invoicesRepository = invoicesRepository;

this.\_usersRepository = usersRepository;

this.\_projectRateRepository = projectRateRepository;

this.\_companyInfoRepository = companyInfoRepository;

this.\_projectRepository = projectRepository;

this.\_tfsTasksRepository = tfsTasksRepository;

}

private IInvoicedTasksRepository \_invoicedTasksRepository { get; set; }

private IInvoicesRepository \_invoicesRepository { get; set; }

private IUsersRepository \_usersRepository { get; set; }

private IProjectRateRepository \_projectRateRepository { get; set; }

private ICompanyInfoRepository \_companyInfoRepository { get; set; }

private IProjectRepository \_projectRepository { get; set; }

private ITFSTasksRepository \_tfsTasksRepository { get; set; }

public InvoiceModel Get(int invoiceId)

{

return this.\_invoicesRepository.Get(invoiceId);

}

/// <summary>

/// Get all invoices

/// </summary>

public List<InvoiceModel> GetAll()

{

return this.\_invoicesRepository.GetAll();

}

/// <summary>

/// Insert invoice with invoiced tasks

/// </summary>

public void SaveInvoice(InvoiceModel invoiceModel)

{

var users = this.\_usersRepository.GetAll();

var project = this.\_projectRepository.Get(invoiceModel.ProjectId);

var rates = this.\_projectRateRepository.GetForProject(invoiceModel.ProjectId);

var tasks = this.\_tfsTasksRepository.GetAll(invoiceModel.IterationId.ToString());

var companyInfo = this.\_companyInfoRepository.GetCompanyInfo();

this.FillCompanyInfo(invoiceModel, companyInfo);

this.FillProjectCustomerInfo(invoiceModel, project);

using (var scope = new TransactionScope(TransactionScopeOption.RequiresNew))

{

this.\_invoicesRepository.Insert(invoiceModel);

this.FillInvoicedTasks(invoiceModel, users, tasks, rates);

this.\_invoicedTasksRepository.InsertRange(invoiceModel.InvoicedTasks);

//The Transaction will be completed

scope.Complete();

}

}

/// <summary>

/// Fill invoiced tasks

/// </summary>

private void FillInvoicedTasks(

InvoiceModel invoiceModel,

List<UserModel> users,

List<TaskModel> tasks,

List<ProjectRateModel> rates)

{

invoiceModel.InvoicedTasks = new List<InvoicedTaskModel>();

foreach (var task in tasks)

{

var internalRate = 0.0;

var externalRate = 0.0;

var user = users.FirstOrDefault(u => u.PMCID == task.AssignedToUserPMCId);

if (user != null)

{

internalRate = Math.Round(user.InternalRate, 2);

var projectUserRate = rates.FirstOrDefault(r => r.UserId == user.UserId);

if (projectUserRate != null)

{

externalRate = Math.Round(projectUserRate.ExternalRate, 2);

}

}

invoiceModel.InvoicedTasks.Add(new InvoicedTaskModel() {

InvoiceId = invoiceModel.InvoiceId,

UserId = user == null ? 0 : user.UserId,

ReportedHours = task.HoursReported,

UserInternalRate = internalRate,

UserExternalRate = externalRate,

TaskName = task.Name

});

}

}

/// <summary>

/// Fill company info

/// </summary>

private void FillCompanyInfo(InvoiceModel invoiceModel, CompanyInfoModel companyInfo)

{

invoiceModel.CompanyName = companyInfo.CompanyName;

invoiceModel.MainAccpuntantName = companyInfo.MainAccpuntantName;

invoiceModel.OwnerName = companyInfo.OwnerName;

invoiceModel.LocationAddress = companyInfo.LocationAddress;

invoiceModel.Fax = companyInfo.Fax;

invoiceModel.Phone = companyInfo.Phone;

invoiceModel.TaxRate = companyInfo.TaxRate;

invoiceModel.OfficeRate = companyInfo.OfficeRate;

}

/// <summary>

/// Fill customer info from project

/// </summary>

/// <param name="invoiceModel"></param>

/// <param name="project"></param>

private void FillProjectCustomerInfo(InvoiceModel invoiceModel, ProjectModel project)

{

invoiceModel.CustomerName = project.CustomerName;

invoiceModel.CustomerAddress = project.CustomerAddress;

invoiceModel.CustomerEmail = project.CustomerEmail;

invoiceModel.CustomerPhone = project.CustomerPhone;

}

}

}

using System.Collections.Generic;

using System.Linq;

using ProjectsAccounting.DAL.Repositories;

using ProjectsAccounting.TFS.Repositories;

using ProjectsAccounting.Common.Models;

using ProjectsAccountingBL.Providers.Abstract;

namespace ProjectAccountingBL.Providers.Implementation

{

public class UsersProvider : IUsersProvider

{

public UsersProvider(IUsersRepository userRepository, ITFSUsersRepository tfsUsersRepository)

{

this.\_userRepository = userRepository;

this.\_tfsUsersRepository = tfsUsersRepository;

}

private IUsersRepository \_userRepository { get; set; }

private ITFSUsersRepository \_tfsUsersRepository { get; set; }

public List<UserModel> GetAll()

{

return this.\_userRepository.GetAll();

}

public void ChangeInternalRate(int userId, double internalRate)

{

this.\_userRepository.ChangeInternalRate(userId, internalRate);

}

/// <summary>

/// Synhronize users from DB wit tfs users

/// </summary>

public void Synhronize()

{

var ownUsers = this.\_userRepository.GetAll();

var tfsUsers = this.\_tfsUsersRepository.GetAll();

foreach (var tfsUser in tfsUsers)

{

if (!ownUsers.Any(p => p.PMCID == tfsUser.PMCID))

{

this.\_userRepository.Insert(tfsUser);

}

}

}

}

}

using ProjectsAccounting.Common.Models;

using ProjectsAccounting.DAL.Repositories;

using ProjectsAccounting.TFS.Repositories;

using ProjectsAccountingBL.Providers.Abstract;

using System.Collections.Generic;

using System.Linq;

namespace ProjectsAccountingBL.Providers.Implementation

{

public class ProjectsProvider : IProjectsProvider

{

public ProjectsProvider(IProjectRepository projectRepository, ITFSProjectsRepository tfsProjectsRepository)

{

this.\_projectRepository = projectRepository;

this.\_tfsProjectsRepository = tfsProjectsRepository;

}

private IProjectRepository \_projectRepository { get; set; }

private ITFSProjectsRepository \_tfsProjectsRepository { get; set; }

public ProjectModel Get(int projectId)

{

return this.\_projectRepository.Get(projectId);

}

public List<ProjectModel> GetAll()

{

return this.\_projectRepository.GetAll();

}

public void UpdateCustomerInfo(ProjectModel model)

{

this.\_projectRepository.UpdateCustomerInfo(model);

}

/// <summary>

/// Synhronize projects from DB wit tfs projects

/// </summary>

public void Synhronize()

{

var ownProjects = this.\_projectRepository.GetAll();

var tfsProjects = this.\_tfsProjectsRepository.GetAll();

foreach (var tfsProject in tfsProjects)

{

if (!ownProjects.Any(p => p.PMCID == tfsProject.PMCID))

{

this.\_projectRepository.Insert(tfsProject);

}

}

}

}

}

using ProjectsAccounting.Common.Models;

using ProjectsAccounting.UI.Models;

using ProjectsAccountingBL.Providers.Abstract;

using System.Linq;

using System.Web.Mvc;

namespace ProjectsAccounting.UI.Controllers

{

public class InvoicingController : Controller

{

public InvoicingController(

IProjectsProvider projectsProvider,

IUsersProvider usersProvider,

IIterationsProvider iterationsProvider,

ITasksProvider tasksProvider,

IInvoicesProvider invoicesProvider)

{

this.\_projectsProvider = projectsProvider;

this.\_usersProvider = usersProvider;

this.\_iterationsProvider = iterationsProvider;

this.\_tasksProvider = tasksProvider;

this.\_invoicesProvider = invoicesProvider;

}

private IProjectsProvider \_projectsProvider { get; set; }

private IUsersProvider \_usersProvider { get; set; }

private IIterationsProvider \_iterationsProvider { get; set; }

private ITasksProvider \_tasksProvider { get; set; }

private IInvoicesProvider \_invoicesProvider { get; set; }

public ActionResult Index()

{

var projects = this.\_projectsProvider.GetAll().Select(u => new SelectListItem { Text = u.ProjectName, Value = u.ProjectId.ToString() }).ToList();

projects.Insert(0, new SelectListItem() { Text = "", Value = "" });

var model = new InvoicingViewModel()

{

ProjectsOptions = projects

};

return View(model);

}

/// <summary>

/// Load all iterations for projects

/// </summary>

[HttpPost]

public ActionResult LoadIterations(int projectId)

{

var project = this.\_projectsProvider.Get(projectId);

var iterations = this.\_iterationsProvider.GetAll(project.PMCID).Select(u => new SelectListItem { Text = u.Name, Value = u.PMCID }).ToList();

iterations.Insert(0, new SelectListItem() { Text = "", Value = "" });

return PartialView("IterationsList", iterations);

}

/// <summary>

/// Load all tasks for iteration

/// </summary>

[HttpPost]

public ActionResult LoadTasks(string iterationId)

{

var tasks = this.\_tasksProvider.GetAll(iterationId);

return PartialView("InvoicedTasks", tasks);

}

/// <summary>

/// Insert invoice

/// </summary>

[HttpPost]

public void SaveInvoice(InvoiceModel invoiceModel)

{

this.\_invoicesProvider.SaveInvoice(invoiceModel);

}

}

}

using ProjectsAccounting.Common.Models;

using ProjectsAccounting.UI.Models;

using ProjectsAccountingBL.Providers.Abstract;

using System.Linq;

using System.Web.Mvc;

namespace ProjectsAccounting.UI.Controllers

{

public class RatesController : Controller

{

public RatesController(

IProjectsProvider projectsProvider,

IProjectRatesProvider projectRatesProvider,

ICompanyInfoProvider companyInfoProvider,

IUsersProvider usersProvider)

{

this.\_projectsProvider = projectsProvider;

this.\_projectRatesProvider = projectRatesProvider;

this.\_companyInfoProvider = companyInfoProvider;

this.\_usersProvider = usersProvider;

}

private IProjectsProvider \_projectsProvider { get; set; }

private IProjectRatesProvider \_projectRatesProvider { get; set; }

private ICompanyInfoProvider \_companyInfoProvider { get; set; }

private IUsersProvider \_usersProvider { get; set; }

public ActionResult Index()

{

var companyInfo = this.\_companyInfoProvider.GetCompanyInfo();

var users = this.\_usersProvider.GetAll();

var projects = this.\_projectsProvider.GetAll();

var pojectRates = this.\_projectRatesProvider.GetAll();

var model = new RatesViewModel()

{

Users = users,

ProjectRates = pojectRates,

CompanyInfo = companyInfo,

UsersOptions = users.Select(u => new SelectListItem { Text = u.UserName, Value = u.UserId.ToString() }).ToList(),

ProjectsOptions = projects.Select(u => new SelectListItem { Text = u.ProjectName, Value = u.ProjectId.ToString() }).ToList()

};

return View(model);

}

/// <summary>

/// Save external user rate

/// </summary>

[HttpPost]

public JsonResult SaveProjectUserRate(ProjectRateModel model)

{

if (model.UserId != 0 || model.ProjectId != 0)

{

var exist = this.\_projectRatesProvider.GetForUserAndProject(model.UserId, model.ProjectId);

if (exist == null)

{

this.\_projectRatesProvider.Insert(model);

}

else

{

model.ProjectRateId = exist.ProjectRateId;

this.\_projectRatesProvider.Update(model);

}

return Json(new { Saved = true });

}

else

{

return Json(new { Saved = false, ErrorMsg = "Select project and user." });

}

}

/// <summary>

/// Save internal user rate

/// </summary>

[HttpPost]

public void SaveInternalRate(int userId, double internalRate)

{

this.\_usersProvider.ChangeInternalRate(userId, internalRate);

}

/// <summary>

/// Synhronize Users

/// </summary>

[HttpPost]

public void SynhronizeUsers()

{

this.\_usersProvider.Synhronize();

}

}

}

@{

ViewBag.Title = "Login Page";

}

<div class="container">

<div class="col-md-3">

<form class="form-signin">

<h2 class="form-signin-heading">Please sign in</h2>

<label for="inputEmail" class="sr-only">Email address</label>

<input type="email" id="inputEmail" class="form-control" placeholder="Email address" required="" autofocus="">

<label for="inputPassword" class="sr-only">Password</label>

<input type="password" id="inputPassword" class="form-control" placeholder="Password" required="">

<div class="checkbox">

<label>

<input type="checkbox" value="remember-me"> Remember me

</label>

</div>

<button class="btn btn-default" type="submit">Sign in</button>

</form>

</div>

</div>

@model List<ProjectsAccounting.Common.Models.InvoiceModel>

@{

ViewBag.Title = "Invoices";

}

@section scripts {

@Scripts.Render("~/bundles/invoices")

}

<div class="container">

<div class="row">

<div class="col-md-12">

<fieldset class="fieldset-details">

<legend>Ivoices</legend>

<div class="table-responsive">

<table class="table table-striped" id="invoicesTable">

<thead>

<tr>

<th>Invoice Number</th>

<th>Generate</th>

<th>Date</th>

<th>Project</th>

<th>Customer</th>

<th>Comment</th>

<th class="text-right">Total</th>

</tr>

</thead>

<tbody>

@if (Model != null && Model.Any())

{

foreach (var invoice in Model)

{

<tr>

<td>#@invoice.InvoiceId</td>

<th>

<button class="btn btn-default generateReportButton"

data-invoice-id="@invoice.InvoiceId"

data-request-url="@Url.Action("LoadInvoice", "Invoices")">Generate</button>

</th>

<td>@invoice.InvoiceDate.ToShortDateString()</td>

<td>@invoice.Project.ProjectName</td>

<td>@invoice.Project.CustomerName</td>

<td>@invoice.Notes</td>

<td class="text-right text-bold">@invoice.InvoicedTasks.Sum(t => t.UserExternalRate \* t.ReportedHours).ToString("0.00")</td>

</tr>

}

}

</tbody>

</table>

</div>

</fieldset>

</div>

</div>

<div class="row">

<div class="col-md-6" id="invoiceContainer">

</div>

</div>

</div>

@model ProjectsAccounting.UI.Models.InvoicingViewModel

@{

ViewBag.Title = "Invoicing";

}

@section scripts {

@Scripts.Render("~/bundles/invoicing")

}

<div class="container">

<form id="invoicingForm">

<div class="row">

<div class="col-md-3">

<fieldset class="fieldset-details">

<legend>Create Invoice</legend>

<div class="form-group">

<label for="InvoiceDate">Ivoice Date:</label>

<input type="text" class="form-control datepicker" id="InvoiceDate" name="InvoiceDate" value="@DateTime.Now.ToShortDateString()">

</div>

<div class="form-group">

<label for="Notes">Comment:</label>

<input type="text" class="form-control" id="Notes" name="Notes">

</div>

<div class="form-group">

<label for="ProjectId">Select Project:</label>

@Html.DropDownList("ProjectId", Model.ProjectsOptions, new { id = "ProjectId", @class = "form-control", data\_request\_url = Url.Action("LoadIterations", "Invoicing") })

</div>

<div class="form-group">

<label for="IterationId">Select Iteration:</label>

<input class="form-control" disabled id="IterationId" />

</div>

</fieldset>

<button class="btn btn-info" id="saveInvoiceBtn" data-request-url="@Url.Action("SaveInvoice", "Invoicing")" disabled>Save</button>

</div>

<div id="invoicesTasksContainer"></div>

</div>

</form>

</div>

@model List<ProjectsAccounting.Common.Models.ProjectModel>

@{

ViewBag.Title = "Projects";

}

@section scripts {

@Scripts.Render("~/bundles/projects")

}

<div class="container">

<div class="row">

<div class="col-md-9">

<fieldset class="fieldset-details">

<legend>Projects</legend>

<div class="table-responsive">

<table class="table table-striped" id="projectsTable">

<thead>

<tr>

<th>Project Name</th>

<th>Customer</th>

<th>Customer Phone</th>

<th>Customer Email</th>

<th>Customer Address</th>

</tr>

</thead>

<tbody>

@if (Model != null && Model.Any())

{

foreach (var project in Model)

{

<tr data-project-id="@project.ProjectId" data-request-url="@Url.Action("UpdateProjectCustomerInfo", "Projects")">

<td>@project.ProjectName</td>

<td class="td-with-input">

<input class="form-control table-input-text customer-info name" value="@project.CustomerName" />

</td>

<td class="td-with-input">

<input class="form-control table-input-text customer-info phone" value="@project.CustomerPhone" />

</td>

<td class="td-with-input">

<input type="email" class="form-control table-input-text customer-info email" value="@project.CustomerEmail" />

</td>

<td class="td-with-input">

<input class="form-control table-input-text customer-info address" value="@project.CustomerAddress" />

</td>

</tr>

}

}

</tbody>

</table>

</div>

<button class="btn btn-info" id="synchronize" data-request-url="@Url.Action("SynhronizeProjects", "Projects")">Synchronize</button>

</fieldset>

</div>

</div>

</div>

@model ProjectsAccounting.UI.Models.RatesViewModel

@{

ViewBag.Title = "Rates";

}

@section scripts {

@Scripts.Render("~/bundles/rates")

}

<div class="container">

<div class="row">

<div class="col-md-6">

<fieldset class="fieldset-details">

<legend>Internal Rates (Office Rate: @Model.CompanyInfo.OfficeRate.ToString("0.00"))</legend>

<div class="table-responsive">

<table class="table table-striped" id="usersTable">

<thead>

<tr>

<th>Name</th>

<th>Full Name</th>

<th>Rate</th>

</tr>

</thead>

<tbody>

@if (Model.Users != null && Model.Users.Any())

{

foreach (var user in Model.Users)

{

<tr>

<td>@user.UserName</td>

<td>@user.FullName</td>

<td class="td-with-input">

<input class="form-control table-input internal-rate text-right" data-user-id="@user.UserId" data-request-url="@Url.Action("SaveInternalRate", "Rates")" value="@user.InternalRate.ToString("0.00")" />

</td>

</tr>

}

}

</tbody>

</table>

</div>

<button class="btn btn-info" id="synchronize" data-request-url="@Url.Action("SynhronizeUsers", "Rates")">Synchronize</button>

</fieldset>

</div>

<div class="col-md-6">

<fieldset class="fieldset-details">

<legend>External Rates</legend>

<div class="table-responsive">

<table class="table table-striped" id="externalRatesTable">

<thead>

<tr>

<th>Project</th>

<th>User</th>

<th>Rate</th>

</tr>

</thead>

<tbody>

@if (Model.ProjectRates != null && Model.ProjectRates.Any())

{

foreach (var projectRate in Model.ProjectRates)

{

var user = Model.Users.FirstOrDefault(u => u.UserId == projectRate.UserId);

var isYellowRow = user != null && ((user.InternalRate + Model.CompanyInfo.OfficeRate) == projectRate.ExternalRate);

var isRedRow = user != null && ((user.InternalRate + Model.CompanyInfo.OfficeRate) > projectRate.ExternalRate);

<tr data-project-id="@projectRate.UserId" class="@(isYellowRow ? "yellowRow" : isRedRow ? "redRow" : "")">

<td>@projectRate.Project.ProjectName</td>

<td>@projectRate.User.UserName</td>

<td class="text-right">@projectRate.ExternalRate.ToString("0.00")</td>

</tr>

}

}

</tbody>

<tfoot>

<tr>

<th>@Html.DropDownList("projectSelector", Model.ProjectsOptions, new { id = "projectSelector", @class = "form-control" })</th>

<th>@Html.DropDownList("userSelector", Model.UsersOptions, new { id = "userSelector", @class = "form-control" })</th>

<th><input type="number" min="0" class="form-control text-right" id="externalRate" value="0" /></th>

</tr>

</tfoot>

</table>

<button class="btn btn-info" id="saveExteralRate" data-request-url="@Url.Action("SaveProjectUserRate", "Rates")">Save</button>

</div>

</fieldset>

</div>

</div>

</div>

var initInvoicePrint = function () {

$("#printInvoive").click(function () {

var invoiceDoc = $("#invoice");

invoiceDoc.print();

});

}

$(document).ready(function () {

$("#invoicesTable").dataTable({});

$(".generateReportButton").click(function () {

var btn = $(this);

var invoiceId = btn.data("invoice-id");

var url = btn.data("request-url");

$("#invoiceContainer").load(url, { invoiceId }, function () {

initInvoicePrint();

});

});

});

$(document).ready(function () {

$("#projectsTable").dataTable({ ordering: false });

$("#synchronize").click(function () {

var url = $(this).data("request-url");

$.post(url, {}, function () {

window.location.reload();

});

});

$(".customer-info").change(function () {

var input = $(this);

var row = input.parents().eq(1);

var url = row.data("request-url");

var param = {

ProjectId: row.data("project-id"),

CustomerName: row.find(".name").val(),

CustomerPhone: row.find(".phone").val(),

CustomerEmail: row.find(".email").val(),

CustomerAddress: row.find(".address").val(),

};

$.post(url, param, () => { });

});

});

var bindIterationChange = function () {

$("#IterationId").change(function () {

var input = $(this);

var saveInvoiceBtn = $("#saveInvoiceBtn");

tasksContainer = $("#invoicesTasksContainer");

tasksContainer.empty();

var iterationId = input.val();

saveInvoiceBtn.prop("disabled", true);

if (iterationId !== "") {

var url = input.data("request-url");

tasksContainer.load(url, { iterationId }, function () {

$("#tasksTable").dataTable({});

saveInvoiceBtn.prop("disabled", false);

});

}

});

};

$(document).ready(function () {

$('.datepicker').datepicker();

$("#ProjectId").change(function () {

var input = $(this);

var iterationsList = $("#IterationId");

$("#invoicesTasksContainer").empty();

$("#saveInvoiceBtn").prop("disabled", true);

iterationsList.empty().prop("disabled", true);

var projectId = input.val();

if (projectId !== "") {

var url = input.data("request-url");

$.post(url, { projectId }, function (iterationsPartial) {

iterationsList.replaceWith(iterationsPartial);

bindIterationChange();

iterationsList.empty().prop("disabled", false);

});

}

});

$("#saveInvoiceBtn").click(function () {

event.preventDefault();

var url = $(this).data("request-url");

var form = $("#invoicingForm");

$.post(url, form.serialize(), function () {

alert("Invoice saved!");

window.location.reload();

});

});

});

$(document).ready(function () {

$("#usersTable").dataTable({});

$("#externalRatesTable").dataTable({});

$("#synchronize").click(function () {

var url = $(this).data("request-url");

$.post(url, {}, function () {

window.location.reload();

});

});

$("#saveExteralRate").click(function () {

var url = $(this).data("request-url");

var param = {

ProjectId: $("#projectSelector").val(),

UserId: $("#userSelector").val(),

ExternalRate: Number($("#externalRate").val())

};

$.post(url, param, function () {

window.location.reload();

});

});

$(".internal-rate").change(function () {

var input = $(this);

var url = input.data("request-url");

var userId = input.data("user-id");

var internalRate = Number(Number(input.val()).toFixed(2));

$.post(url, { userId, internalRate }, function () {

window.location.reload();

});

});

});