НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра автоматизованих систем обробки інформації та управління

(повна назва кафедри)

Напрям підготовки 6.050101 «Комп’ютерні науки»

**РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА**

з дисципліни: «Технологія створення програмних продуктів»

на тему: *Система планування робіт на підприємстві*

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата захисту: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени проекту \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сінюков Д.Г.

(підпис) (прізвище та ініціали)

Перевірила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сперкач М.О.

(підпис) (прізвище та ініціали

Київ – 2015

## Зміст

[1 Загальні положення 4](#_Toc421633370)

[1.1 Опис предметного середовища 4](#_Toc421633371)

[1.2 Опис процесу діяльності 5](#_Toc421633372)

[1.3 Опис функціональної моделі 7](#_Toc421633373)

[2 Рішення з Інформаційного забезпечення 14](#_Toc421633374)

[2.1 Вхідні дані. 14](#_Toc421633375)

[2.2 Вихідні дані. 14](#_Toc421633376)

[2.3 Опис інформаційного забезпечення. 14](#_Toc421633377)

[3 Рішення з програмного забезпечення 18](#_Toc421633378)

[3.1 Опис вибраного програмного забезпечення 18](#_Toc421633379)

[3.2 Архітектура програмного забезпечення 19](#_Toc421633380)

[3.3 Специфікація функцій 23](#_Toc421633381)

[3.4 Вимоги до якості 24](#_Toc421633382)

[4 Керівництво користувача 26](#_Toc421633383)

[4.1 Інструкція користувача 26](#_Toc421633384)

[4.2 Методика випробувань 37](#_Toc421633385)

[Висновок 45](#_Toc421633386)

[Список використаної літератури 46](#_Toc421633387)

[Додаток А 47](#_Toc421633388)

[Додаток Б 48](#_Toc421633389)

## **1 Загальні положення**

### Опис предметного середовища

Застосування, що було створено, призначено для використання працівниками підприємства, яке випускає вироби. У кожного вироба є свій директивний термін (deadline), технологічна картка, де описано, які операції та який час необхідно виконувати, щоб його виготовити. Керуванням цього займається співробітник технологічного відділу. На підприємстві є дільниці, якими керує головний інженер. На кожній дільниці є набір приладів, кожен з яких виконує свій набір операцій.

Тож у системі існує три ролі: співробітник технологічного відділу, головний інженер і гість. Співробітнику технологічного відділу доступні сторінки ведення виробів, їх технологічних карт і операцій. Головний інженер має доступ до інформації, що стосується дільниць, приладів та зв’язку між приладами та операціями. Всі автентифіковані користувачі мають доступ до сторінки аналізу розкладу робіт на підприємстві. Гість може переглядати загальнодоступну інформацію, таку, як на сторінках «Головна», «About» та «Контакти», також йому доступні форми для реєстрації та входу у систему.

Можна виділити наступні цілі створення системи планування виробництва:

* поліпшення роботи з даними, які пов’язані з виробничим процесом;
* оптимізація роботи підприємства за рахунок ефективного використання засобів виробництва.

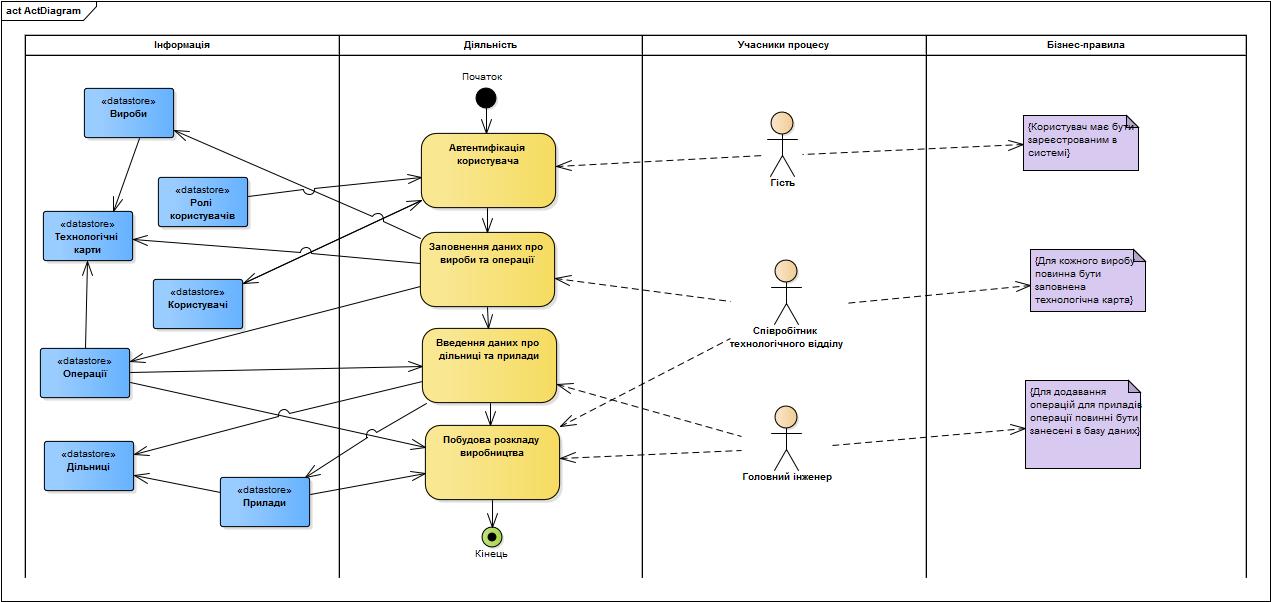
### 1.2 Опис процесу діяльності

Діаграма BPMN [1] описує умові позначення для відображення бізнес-процесів у вигляді діаграм бізнес процесів. Діаграма, що зображена на рисунку 1.1, описує всі бізнес-процеси, які виконуються під час роботи системи.



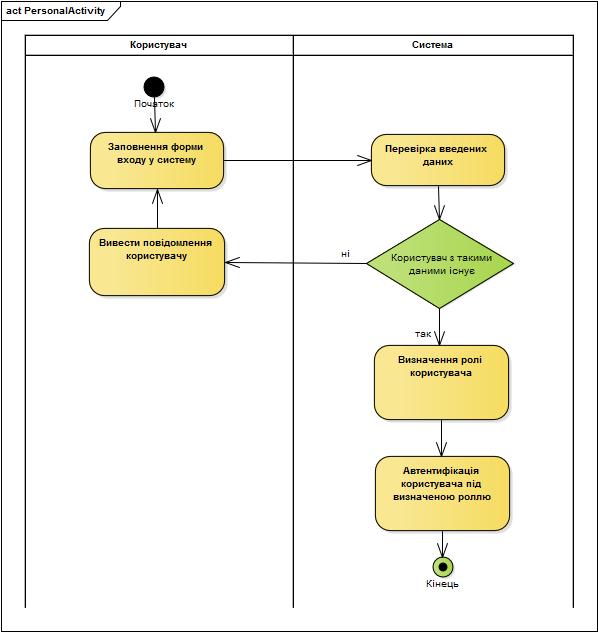
**Рисунок 1.1** – Схема структурного опису бізнес-процесу мовою BPMN

Діаграма опису бізнес-процесу мовою UML [2] представлена на рисунку 1.2. Дана діаграма показує, в яких модулях системи і яким чином відбувається взаємодія акторів з інформацією та які бізнес-правила визначають те, як відбувається дана взаємодія [4].



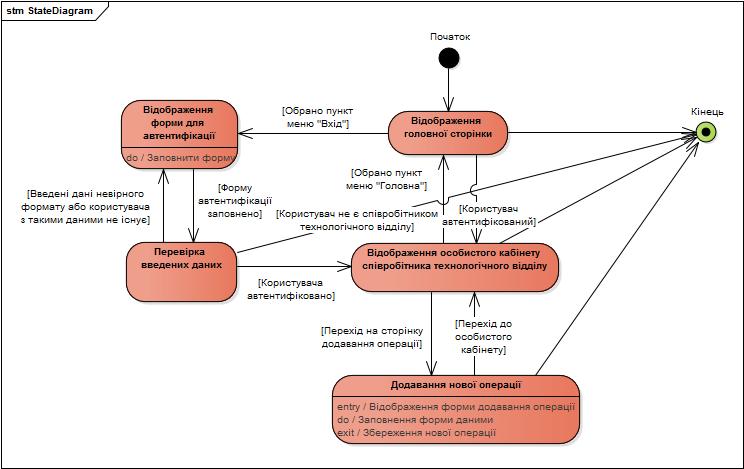
**Рисунок 1.2** – Схема структурна бізнес-процесу мовою UML

Діаграма діяльності представлена на рисунку 1.3. Дана діаграма описує послідовність дій користувача та системи під час проходження процедури автентифікації.



**Рисунок 1.3** – Схема структурна діяльності

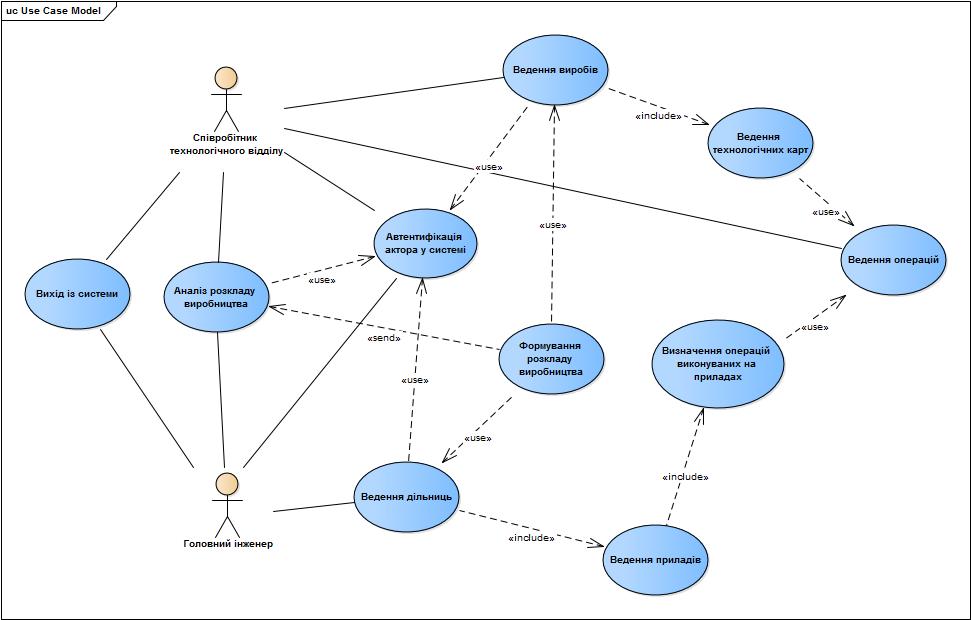
Діаграма станів представлена на рисунку 1.4. Вона показує, яким чином об’єкти переходять із одного стану в інший протягом процесу «Додавання нової операції» співробітником технологічного відділу.



**Рисунок 1.4** – Схема структурна станів

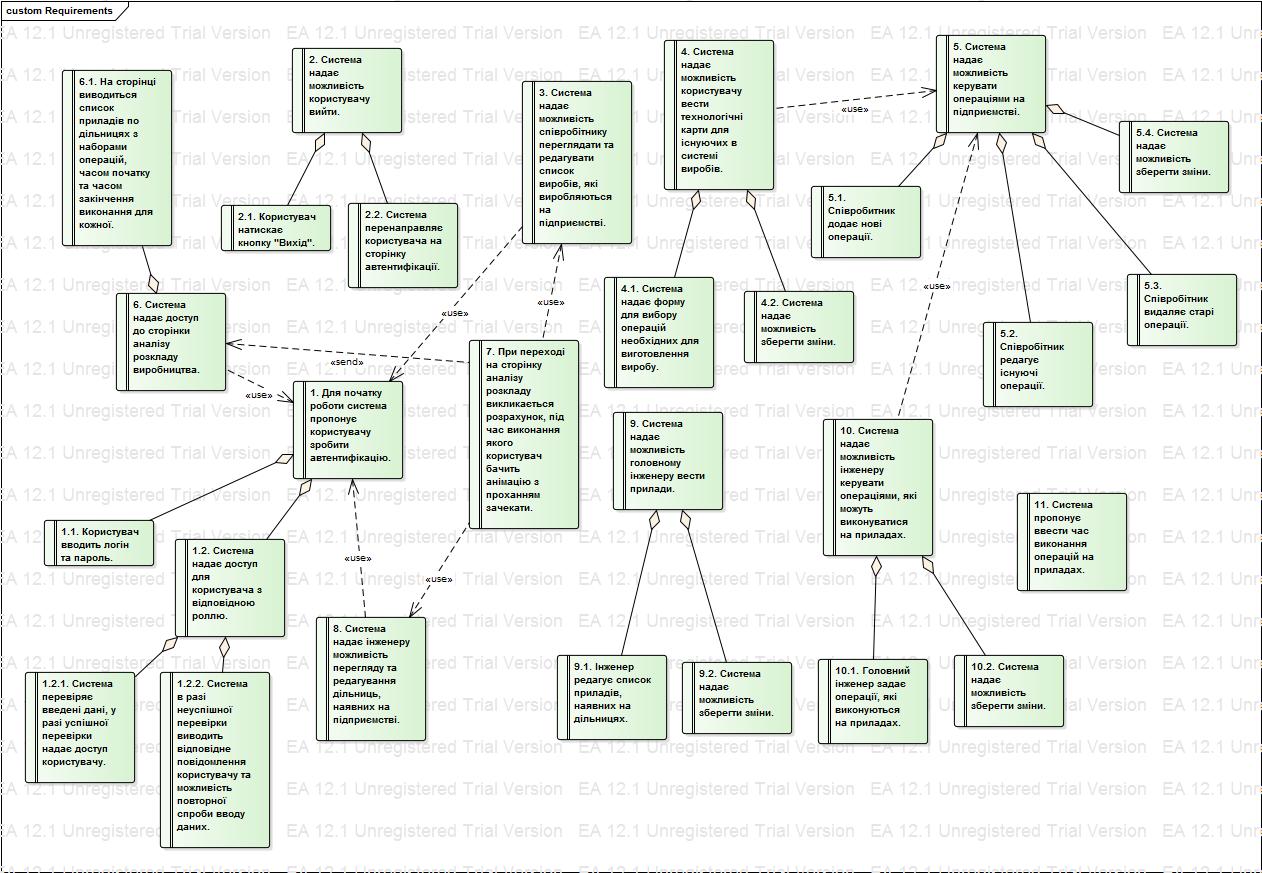
### 1.3 Опис функціональної моделі

Діаграма варіантів використання представлена на рисунку 1.5. Дана діаграма показує, які актори існують в системі, які дії вони виконують та яким чином ці дії пов’язані між собою.



**Рисунок 1.5** – Схема структурна варіантів використання

На рисунку 1.6 представлено модель взаємозв’язку вимог, яка відображає взаємозв’язок всіх представлених вимог до системи.



**Рисунок 1.6** – Схема структурна моделі управління вимогами

Відповідно визначених варіантів використання виявлено функціональні вимоги та встановлена їх пріоритетність, результат для загальної частини наведено у таблиці 2.

Таблиця 1.1 – Виявлені вимоги з варіантів використання

|  |  |  |  |
| --- | --- | --- | --- |
| **Актор** | **Варіант використання** | **Функціональна вимога** | **Пріоритет** |
| Співробітник технологічного відділу, головний інженер | Автентифікація актора у системі | 1. Для початку роботи система пропонує користувачу зробити автентифікацію.  1.1 Користувач вводить логін та пароль.  1.2. Система надає доступ для користувача з відповідною роллю. 1.2.1. Система перевіряє введені дані, у разі успішної перевірки надає доступ користувачу. 1.2.2. Система в разі неуспішної перевірки виводить відповідне повідомлення користувачу та можливість повторної спроби вводу даних. | Високий  Високий  Високий  Високий  Високий |
| Співробітник технологічного відділу, головний інженер | Вихід з системи | 2. Система надає можливість користувачу вийти.  2.1. Користувач натискає кнопку "Вихід".  2.2. Система перенаправляє користувача на сторінку автентифікації. | Високий  Високий  Високий |
| Співробітник технологічного відділу | Ведення виробів | 3. Система надає можливість співробітнику переглядати та редагувати список виробів, які виробляються на підприємстві. | Середній |
| Співробітник технологічного відділу | Ведення технологічних карт | 4. Система надає можливість користувачу вести технологічні карти для існуючих в системі виробів. 4.1. Система надає форму для вибору операцій необхідних для виготовлення виробу. 4.2. Система надає можливість зберегти зміни. | Середній  Середній  Середній |
| Співробітник технологічного відділу | Ведення операцій | 5. Система надає можливість ведення операцій на підприємстві.  5.1. Співробитник додає нові операції.  5.2. Співробітник редагує існуючі операції.  5.3. Співробітник видаляє старі операції.  5.4. Система надає можливість зберегти зміни. | Високий  Високий  Високий  Середній  Високий |
| Співробітник технологічного відділу, головний інженер | Аналіз розкладу виробництва | 6. Система надає доступ до сторінки аналізу розкладу виробництва.  6.1. На сторінці виводиться список приладів по дільницях з наборами операцій, часом початку та часом закінчення виконання для кожної. | Середній  Низький |
| Співробітник технологічного відділу, головний інженер | Формування розкладу виробництва | 7. При переході на сторінку аналізу розкладу викликається розрахунок, під час виконання якого користувач бачить анімацію з проханням зачекати. | Низький |
| Головний інженер | Ведення дільниць | 8. Система надає інженеру можливість перегляду та редагування дільниць, наявних на підприємстві. | Середній |
| Головний інженер | Ведення приладів | 9. Система надає можливість головному інженеру вести прилади.  9.1. Інженер редагує список приладів, наявних на дільницях.  9.2. Система надає можливість зберегти зміни. | Середній  Середній  Середній |
| Головний інженер | Визначення операцій виконуваних на приладах | 10. Система надає можливість інженеру керувати операціями, які можуть виконуватися на приладах.  10.1. Головний інженер задає операції, які виконуються на приладах.  10.2. Система надає можливість зберегти зміни. | Середній  Середній  Середній |
| Головний інженер | Визначення часу виконання операцій на приладах | 11. Система пропонує ввести час виконання операцій на приладах. | Середній |

## **2 Рішення з Інформаційного забезпечення**

### 2.1 Вхідні дані.

Вхідними даними для планування виробництва є:

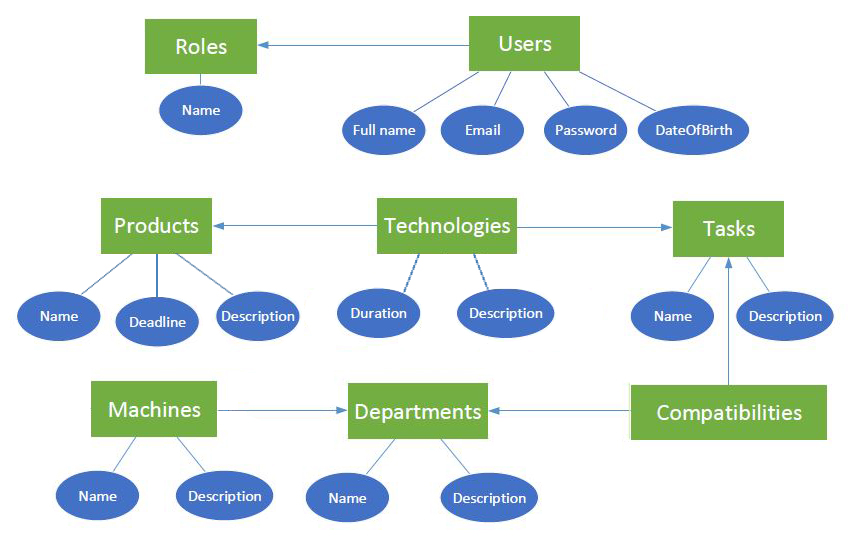
* дані про користувачів: логін, ім’я, електронна адреса, дата народження, роль;
* дані про вироби: ідентифікатор, назва, директивний термін, опис;
* дані про операції: ідентифікатор, назва, опис;
* технологічні карти: виріб, операція, тривалість, опис;
* дані про дільниці: ідентифікатор, назва, опис, сумісні операції;
* дані про прилади: ідентифікатор, назва, дільниця, опис.

### 2.2 Вихідні дані.

У будь-який момент роботи із системою автентифіковані користувачі можуть перейти на сторінку аналізу розкладу виробництва, де користувачу доступна інформація про зайнятість приладів по дільницях у кожен момент часу, часи початку та закінчення робіт.

### 2.3 Опис інформаційного забезпечення.

База даних складається з 8 таблиць: Користувачі (Users), Ролі (Roles), Вироби (Products), Технологічні карти (Technologies), Операції (Tasks), Прилади (Machines), Дільниці (Departments), Сумісності дільниць з операціями (Compatibilities). Зв’язки між таблицями представлені на рисунку 2.1. Детальні описи таблиць бази даних наведені у таблицях 2.1-2.11.



**Рисунок 2.1** – Схема структурна бази даних

Таблиця 2.1 – Опис таблиці Користувачі (Users)

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип | Призначення |
| Id | nvarchar(128) NOT NULL | Ідентифікатор запису |
| Email | nvarchar(256) NOT NULL | Електронна адреса користувача |
| PasswordHash | nvarchar(MAX) NOT NULL | Hash-код пароля |
| UserName | nvarchar(256) NOT NULL | Логін користувача |
| FullName | nvarchar(MAX) NULL | Ім’я користувача |
| BirthDate | Datetime NULL | Дата народження користувача |
| RoleId | nvarchar(128) NULL | Роль користувача |

Таблиця 2.2 – Опис таблиці Ролі (Roles)

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип | Призначення |
| Id | nvarchar(128) NOT NULL | Ідентифікатор запису |
| Name | nvarchar(256) NOT NULL | Назва ролі |

Таблиця 2.3 – Опис таблиці Вироби (Products)

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип | Призначення |
| Id | Int NOT NULL | Ідентифікатор запису |
| Name | nvarchar(50) NOT NULL | Назва виробу |
| Deadline | Datetime NULL | Директивний термін |
| Description | nvarchar(MAX) NOT NULL | Опис виробу |
| C\_Date | Datetime NULL | Дата створення |
| C\_User | nvarchar(128) NULL | Користувач, що створив |

Таблиця 2.4 – Опис таблиці Технологічні карти (Technologies)

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип | Призначення |
| Id | Int NOT NULL | Ідентифікатор запису |
| ProductId | Int NOT NULL | Ідентифікатор виробу |
| TaskId | Int NOT NULL | Ідентифікатор операції |
| Duration | Float NOT NULL | Тривалість в секундах |
| Description | nvarchar(MAX) NOT NULL | Опис |

Таблиця 2.5 – Опис таблиці Операції (Tasks)

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип | Призначення |
| Id | Int NOT NULL | Ідентифікатор запису |
| Name | nvarchar(50) NOT NULL | Назва операції |
| Description | nvarchar(MAX) NOT NULL | Опис операції |
| C\_Date | Datetime NULL | Дата створення |
| C\_User | nvarchar(128) NULL | Користувач, що створив |

Таблиця 2.6 – Опис таблиці Прилади (Machines)

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип | Призначення |
| Id | Int NOT NULL | Ідентифікатор запису |
| Name | nvarchar(50) NOT NULL | Назва приладу |
| DepartmentId | Int NULL | Ідентифікатор дільниці |
| Description | nvarchar(MAX) NOT NULL | Опис приладу |
| C\_Date | Datetime NULL | Дата створення |
| C\_User | nvarchar(128) NULL | Користувач, що створив |

Таблиця 2.7 – Опис таблиці Дільниці (Departments)

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип | Призначення |
| Id | Int NOT NULL | Ідентифікатор запису |
| Name | nvarchar(50) NOT NULL | Назва дільниці |
| Description | nvarchar(MAX) NOT NULL | Опис дільниці |
| C\_Date | Datetime NULL | Дата створення |
| C\_User | nvarchar(128) NULL | Користувач, що створив |

Таблиця 2.8 – Опис таблиці Сумісності дільниць з операціями (Compatibilities)

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип | Призначення |
| Id | Int NOT NULL | Ідентифікатор запису |
| DepartmentId | Int NOT NULL | Ідентифікатор дільниці |
| TaskId | Int NOT NULL | Ідентифікатор операції |

## **3 Рішення з програмного забезпечення**

### 3.1 Опис вибраного програмного забезпечення

Для розробки мобільного застосування було використано платформу Microsoft .NET 4.5.2 та вище [1], середовище Visual Studio [5] версії 2015, мова програмування – C#. Дана система представляє собою веб-застосування на Microsoft ASP.NET Framework. Клієнтська частина застосування зроблена з використанням інструментів HTML, CSS, Bootstrap, Javascript, JQuery. У якості сервера, до якого підключається застосування, обрано IIS Express [6]. База даних створена на СУБД Microsoft SQL Server [7], який поставляється разом із Visual Studio.

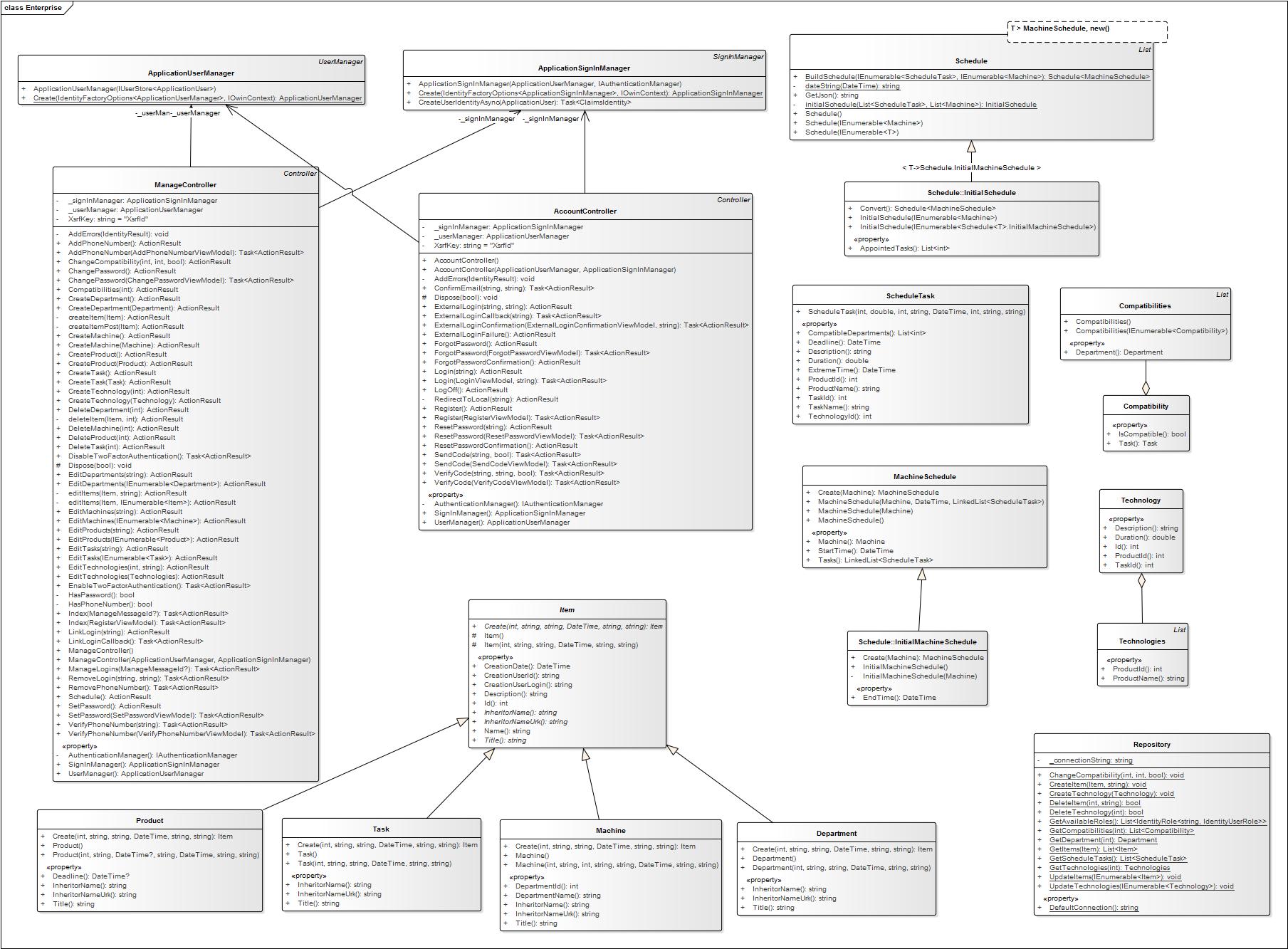
Visual Studio – інтегроване середовище розробки (IDE) для багатьох мов програмування платформи .NET та інших. Дане середовище розробки є офіційним продуктом компанії Microsoft. На сьогоднішній день Visual Studio є найпопулярнішою IDE для розробки застосунків для операційної системи Windows[1].

### 3.2 Архітектура програмного забезпечення

#### 3.2.1 Діаграма класів

Діаграма класів, що зображена на рисунку 3.1, описує головні класи для роботи із системою “Enterprise”, а саме:

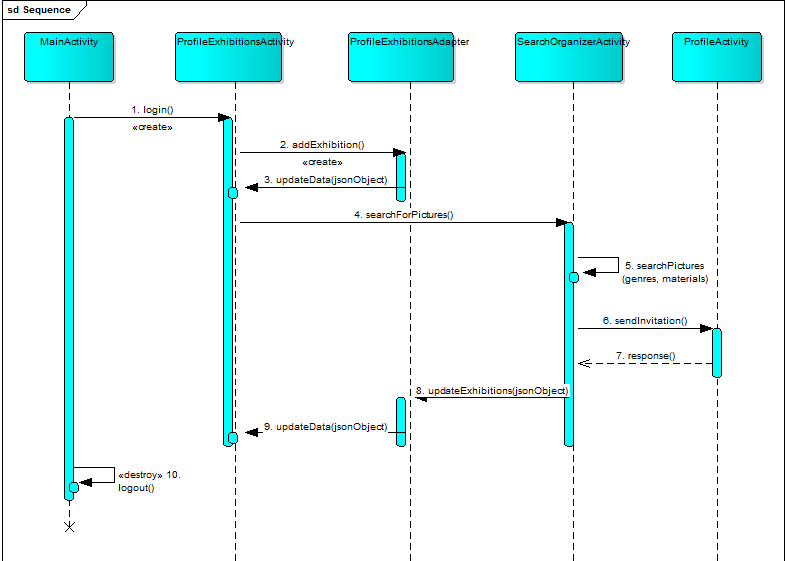
* ApplicationUserManager – сервіс роботи з користувачем;
* ApplicationSignInManager – сервіс автентифікації користувача;
* ManageController – сторінки особистого кабінету користувачів;
* AccountController – сторінки автентифікації;
* Schedule – розклад виробництва;
* InitialSchedule – обслуговуючий клас для побудови розкладу;
* ScheduleTask – клас, що представляє роботу для виконання на приладах;
* MachineSchedule – розклад приладу;
* InitialMachineSchedule – обслуговуючий клас для побудови розкладу одного приладу;
* Repository – сервіс для роботи з базою даних;
* Compatibility – сумісність дільниці з операцією;
* Compatibilities – сумісності дільниць з операціями;
* Technology – технологія;
* Technologies – технологічні карти;
* Item – базовий клас для об’єктів виробництва;
* Product – виріб;
* Task – операція;
* Machine – прилад;
* Department – дільниця.



**Рисунок 3.1** – Схема структурна класів

#### 3.2.2 Діаграма послідовності

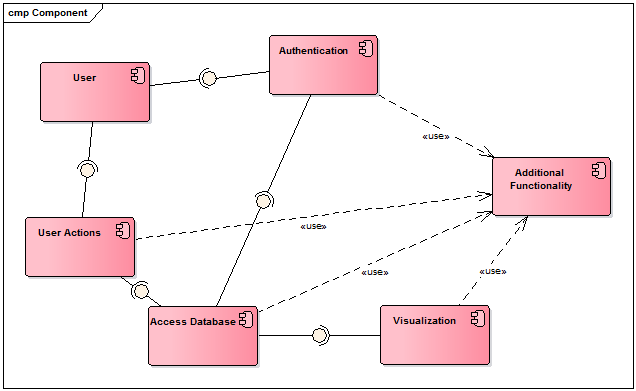
Діаграма послідовностей, що зображена на рисунку 3.2, показує, яким чином відбувається процес додавання нових робіт-учасниць до виставки організатором.



**Рисунок 3.2** – Схема структурна послідовності

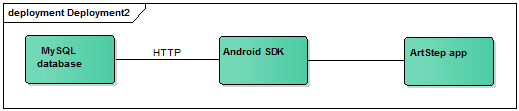
#### 3.2.3 Діаграма компонентів

Діаграма компонентів, що зображена на рисунку 3.3, відображає зв’язки між головними компонентами. В даній системі можна виділити 6 компонент: користувач, аутентифікація, дії користувача, доступ до бази даних, візуалізація та додатковий функціонал.

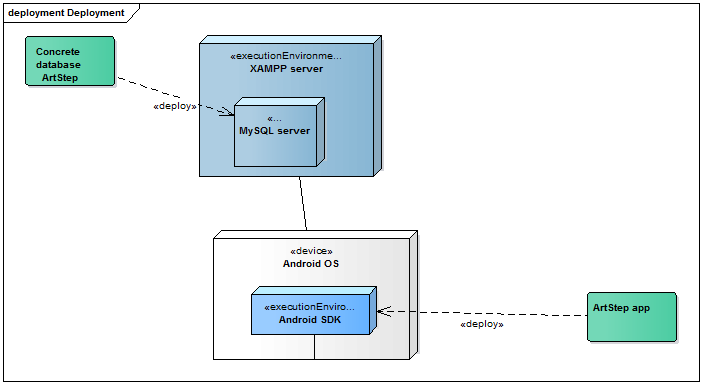


**Рисунок 3.3** – Схема структурна компонентів

На рисунках 3.4 та 3.5 зображено діаграму розгортання за допомогою артефактів та фізичних вузлів відповідно.



**Рисунок 3.4** – Схема структурна розгортання (артефакти)



**Рисунок 3.5** – Схема структурна розгортання (фізичні вузли)

### 3.3 Специфікація функцій

В таблиці 3.1 наведена специфікація функцій до основних класів, а саме: MainActivity, RegistrateActivity, LoginActivity, ProfileAboutUserActivity, ProfileExhibitionsActivity, ProfilePicturesActivity, ReceivedApplicationsActivity, SearchOrganizerActivity, SearchExhibitionsActivity, InviteActivity, ApplyToExhibitionActivity, AddExhibitionActivity, AddPictureActivity.

Таблиця 3.1 – Специфікація функцій

|  |  |  |
| --- | --- | --- |
| Функція | Параметри | Призначення |
| private void getPictures () | - | Зчитування інформації про картини з бази даних |
| private void login() | - | Відправка запиту на вхід до системи |
| private void checkLoggedIn(JSONObject jsonObject) | Дані, отримані з сервера | Перевірка правильності входу |
| private void setUser(JSONObject jsonObject) | Дані про користувача | Запис даних про поточного користувача |
| public boolean validateForm(String login, String password) | Логін та пароль користувача | Перевірка правильності введення даних |
| private void getCountries() | - | Отримання даних про зареєстровані країни |
| private void getCitites(String c) | Назва країни, для якої треба отримати дані про міста | Отримання інформації про міста |
| private void registrate() | - | Відправка запиту про реєстрацію нового користувача |
| private void fillFields(SharedPreferences sh) | Файл із даними, зареєстрований на пристрої | Заповнення полів форми «Профіль» |
| private void getAccepted(int ID) | Номер користувача | Запит на отримання списку прийнятих заявок для користувача |
| public void delete(View v) | Елемент інтерфейсу, який необхідно видалити | Видалення виставки |
| private void getExhibitions(int ID) | Номер користувача | Отримання списку виставок для поточного користувача |
| public void acceptApplication(View v) | Візуальне представлення заявки, яку буде прийнято | Згода на участь у виставці/дозвіл на участь у виставці |
| public void declineApplication(View v) | Візуальне представлення заявки, яку буде прийнято | Відмова від участі у виставці/відмова дозволу на участь у виставці |

Продовження таблиці 3.1

|  |  |  |
| --- | --- | --- |
| public void notShowDeclinedApplication(View v) | Візуальне представлення заявки, яку буде прийнято | Помітка відхиленої заявки як «Переглянутої» |
| public boolean validateFormWithParams(String date, String name, String participants) | Дата проведення виставки, її назва, кількість учасників | Перевірка правильності введення даних |
| public void onActivityResult(int requestCode, int resultCode, Intent data) | Код запиту, код результату, інтент, який показує, яку інформацію отримано | Перевірка вибору картини з галереї пристрою |
| public void addPicture() | - | Завантаження зображення на сервер |
| public void addPictureToDatabase() | - | Додавання інформації про роботу до бази даних |
| public void getPictureDetails(String ID) | Номер роботи | Відправка запиту на отримання детальної інформації про роботу |
| private void addApplication() | - | Відправка запиту на додавання заявки |
| private void invite() | - | Відправка запрошення художнику |
| private void setInfo(JSONObject jsonObject) | Інформація про роботу | Відображення докладної інформації про роботу |
| private Boolean getArtistDetails(String login) | Логін художника | Отримання докладної інформації про художника з бази даних |
| private boolean getExhibitionDetails(String id) | Номер виставки | Отримання докладної інформації про виставку |
| private void apply() | - | Відправка заявки на участь у виставці |
| private boolean checkSelectedCriteria() | - | Перевірка вибраних критеріїв пошуку |
| private void searchPictures() | - | Пошук робіт за вказаними критеріями |
| private void searchArtists() | - | Пошук художників за вказаними критеріями |

### 3.4 Вимоги до якості

#### 3.4.1 Вимоги до надійності

При виникненні збоїв в апаратному забезпеченні, включаючи аварійне вимкнення живлення, програма повинна автоматично відновлювати свою працездатність після усунення збоїв та коректного перезапуску забезпечення (виключаючи ситуації з ушкодженням носіїв інформації з виконуваним програмним кодом).

#### Вимоги до відмовостійкості

Система повинна оброблювати виключні ситуації, пов’язані з некоректно введеними даними. Також система повинна оброблювати ситуації із проблемами з підключенням до бази даних.

## 4 Керівництво користувача

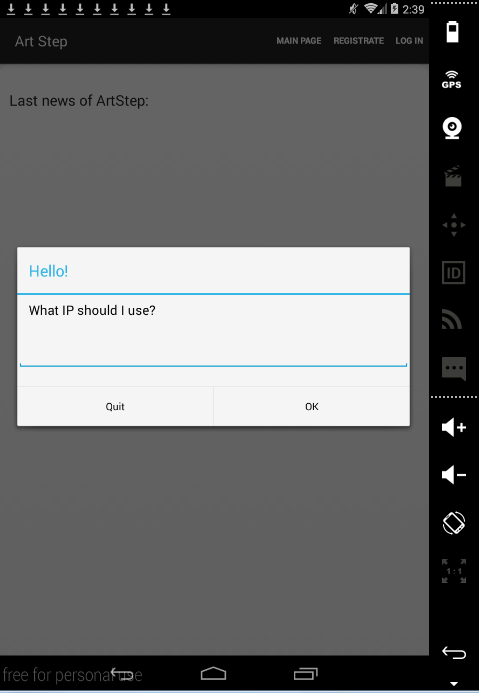
### 4.1 Інструкція користувача

Для початку роботи з системою Вам необхідно:

* скопіювати та розпакувати архів застосунку на вашому комп’ютері;
* налаштувати ваш IIS веб-сервер;
* запустити файл Enterprise.sln у вашій Visual Studio;
* натиснути кнопку «Run», перед цем обравши браузер для запуску.

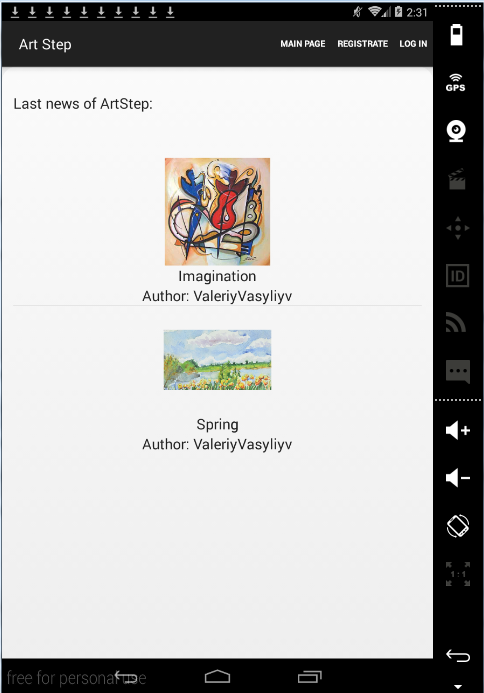
У майбутньому веб-застосування буде розміщено на інтернет-хостингу, після чого його можна буде запустити напряму через браузер, ввівши відповідну адресу, маючи інтернет-підключення.

Після дій вище можна починати роботу з системою. Запустіть додаток на Вашому пристрої. Ви побачите форму, зображену на рисунку 4.1.



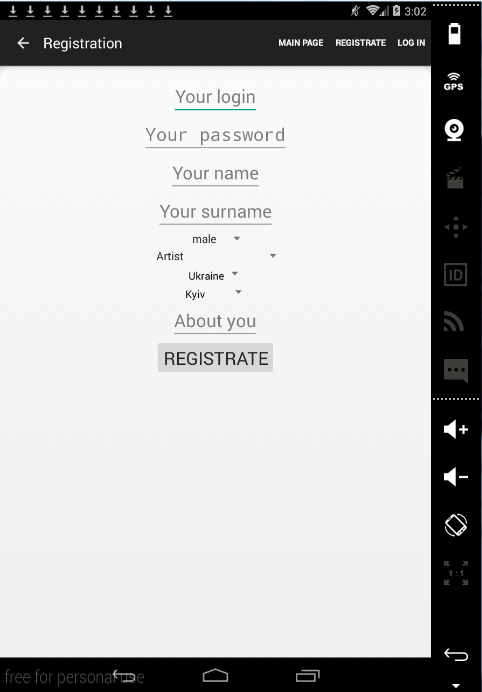
**Рисунок 4.1** – Початкова форма

Для підключення до Вашого сервера введіть його IP-адресу в діалоговому вікні та натисніть ОК. Якщо все вірно, то Ви побачите форму, зображену на рисунку 4.2.



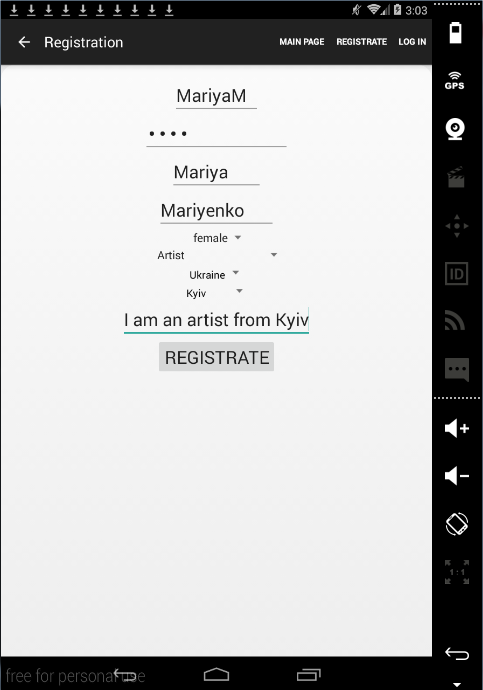
**Рисунок 4.2** – Головна сторінка ArtStep

Розпочнемо із художника. Для початку роботи з системою необхідно зареєструватися. Для цього оберіть пункт меню «Registrate». Відкриється форма, зображена на рисунку 4.3.



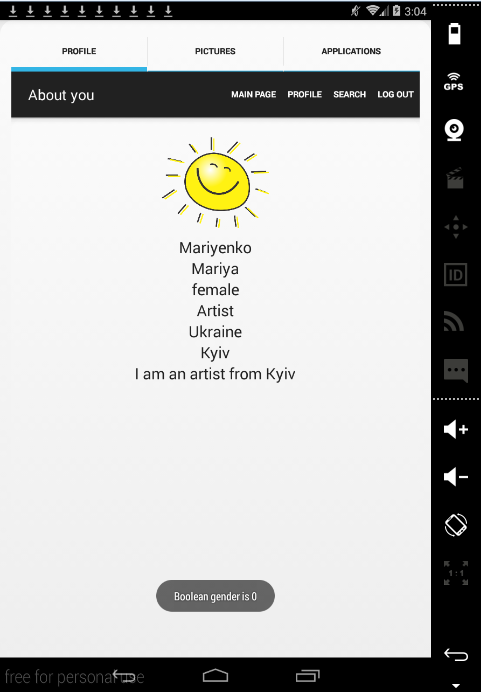
**Рисунок 4.3** – Сторінка реєстрації ArtStep

Заповніть поля форми за прикладом, який показано на рисунку 4.4.



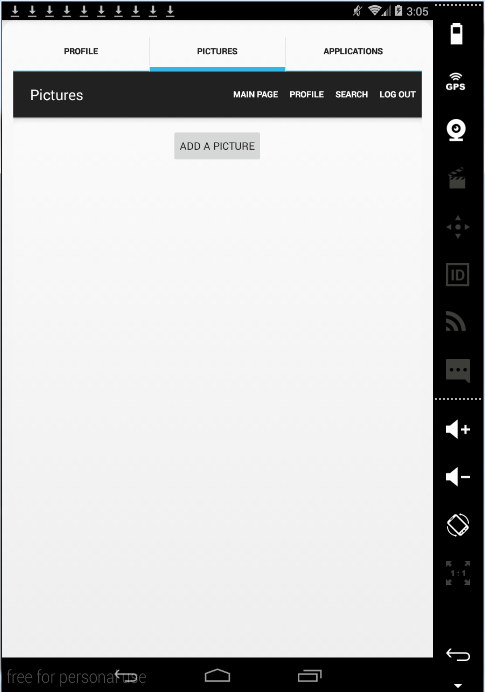
**Рисунок 4.4** – Приклад заповнення форми реєстрації ArtStep

Натисніть кнопку «REGISTRATE». Відкриється форма профілю, яку зображено на рисунку 4.5.



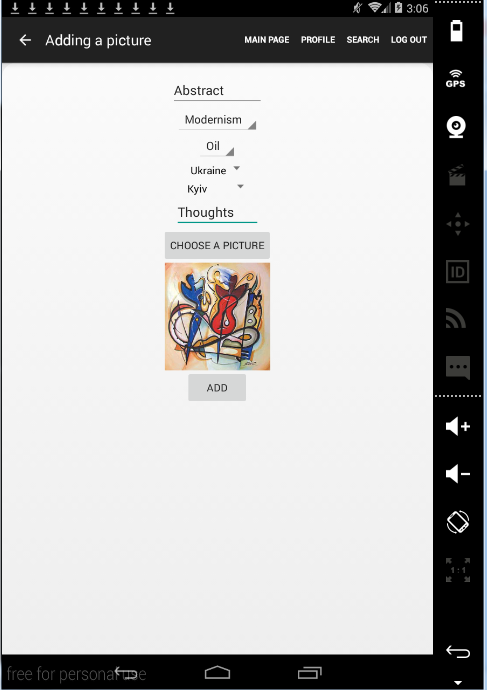
**Рисунок 4.5** – Форма перегляду профілю ArtStep

Натисніть кнопку «PICTURES». Відкриється форма перегляду картин, яку зображено на рисунку 4.6.



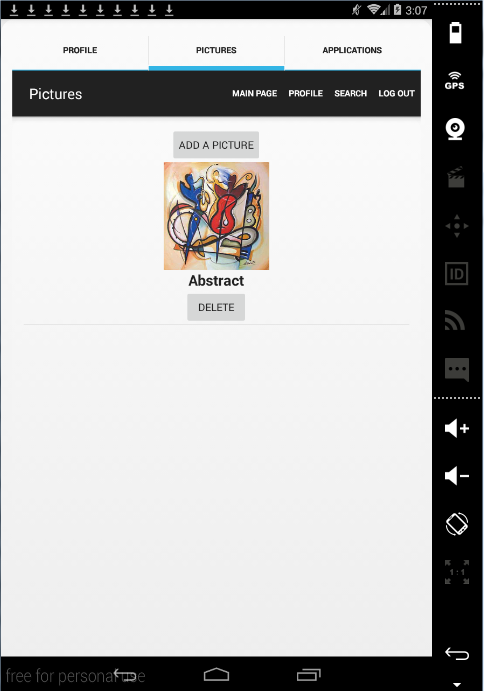
**Рисунок 4.6** – Форма перегляду картин ArtStep

Наразі ця форма пуста. Для того, щоб додати картину, натисніть кнопку «ADD A PICTURE». Відкриється форма додавання картини, яку зображено на рисунку 4.7.



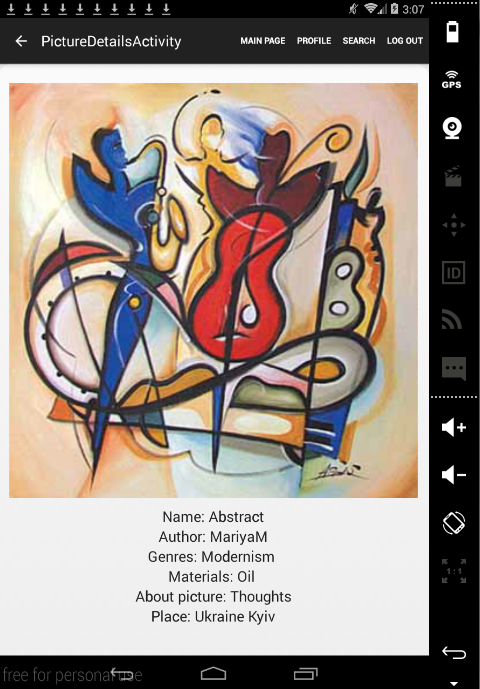
**Рисунок 4.7** – Форма додавання роботи ArtStep

Заповніть поля форми та оберіть фотографію (скан) Вашої картини та. натисніть кнопку «ADD». Відкриється форма профілю. Перейдіть на форму «PICTURES», тепер вона має виглядати так, як показано на рисунку 4.8.



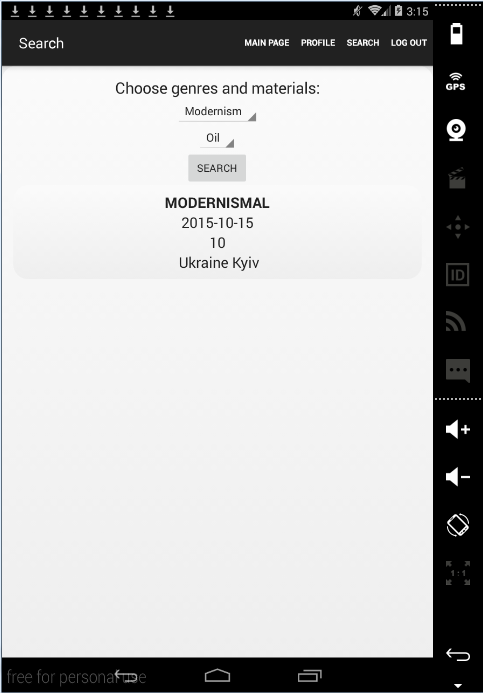
**Рисунок 4.8** – Оновлена форма з картинами ArtStep

Для перегляду докладної інформації про картину натисніть на неї. Відкриється форма перегляду деталей картини, як показано на рисунку 4.9.



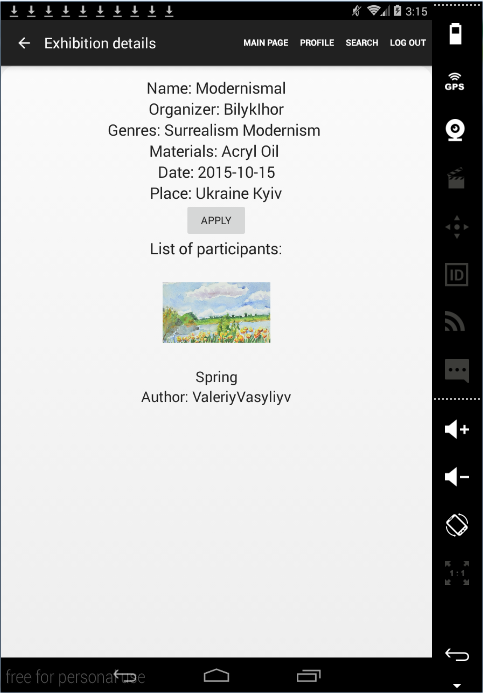
**Рисунок 4.9** – Інформація про картину

Тепер настав час шукати місце у виставках для нашої картини. Натисніть кнопку «SEARCH». Оберіть жанри «Modernism» та матеріал «Oil». Відкриється форма, яку зображено на рисунку 4.10.



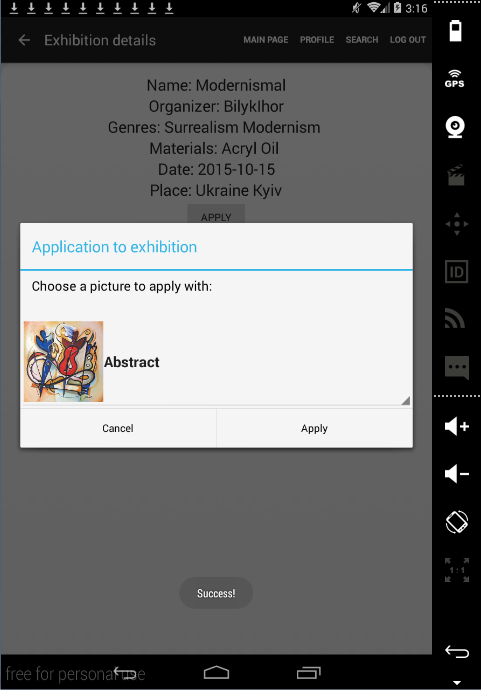
**Рисунок 4.10** – Форма пошуку

Натисніть на виставку «MODERNISMAL». Відкриється форма, яку зображено на рисунку 4.11.



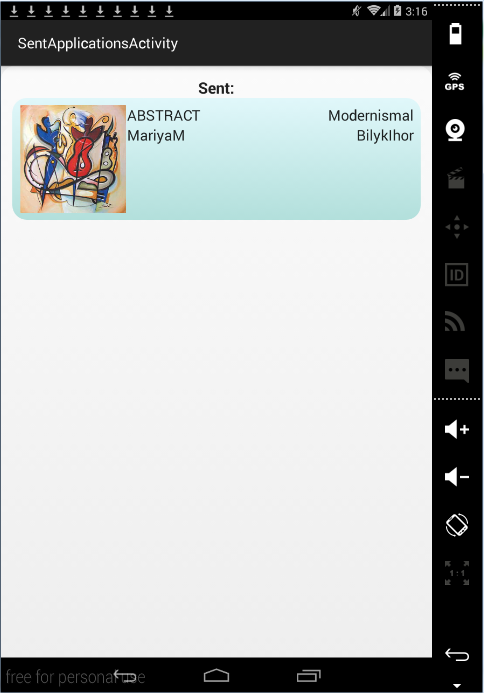
**Рисунок 4.11** – Інформація про виставку

Натисніть на кнопку «APPLY». Відкриється форма, яку зображено на рисунку 4.12.



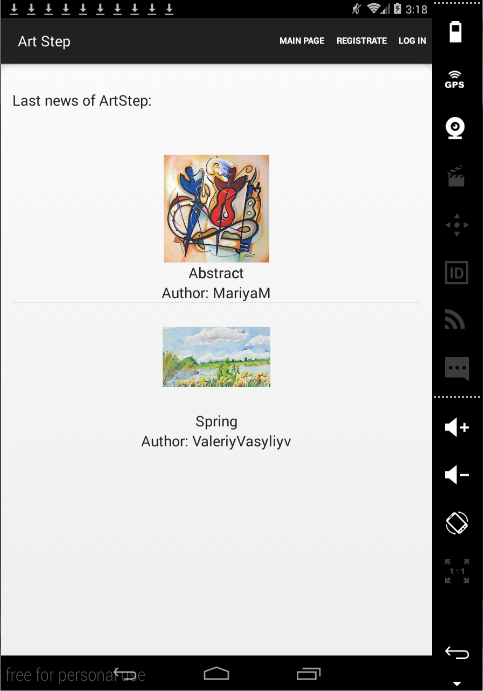
**Рисунок 4.12** – Форма подачі заявки

Оберіть нашу картину «ABSTRACT» та натисніть кнопку «APPLY». Заявку успішно подано. Давайте переглянемо її. Перейдіть на сторінку «PROFILE», звідти – на вкладку «APPLICATIONS» та на сторінку «SENT». Відкриється форма, яку зображено на рисунку 4.13.



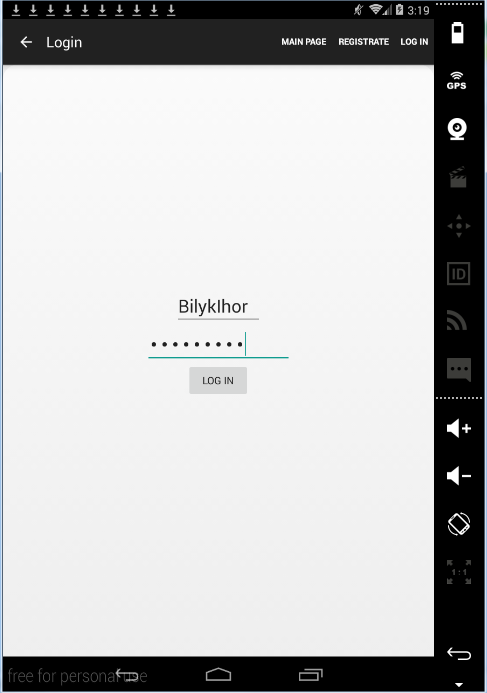
**Рисунок 4.13** – Форма перегляду надісланих заявок

Тепер справа за організатором виставок. Спочатку вийдемо з системи. Для цього натисніть кнопку в меню «LOG OUT». Відкриється форма, яку зображено на рисунку 4.14.



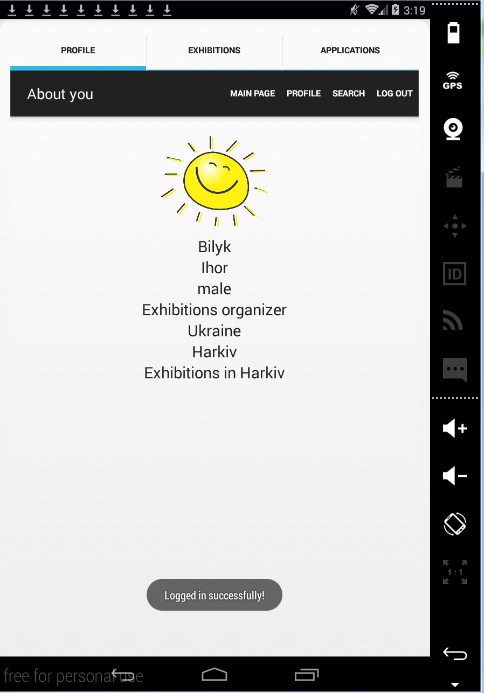
**Рисунок 4.14** –Головна сторінка ArtStep

Увійдемо в систему. Для цього натисніть кнопку «LOG IN». Відкриється форма, яку зображено на рисунку 4.15.



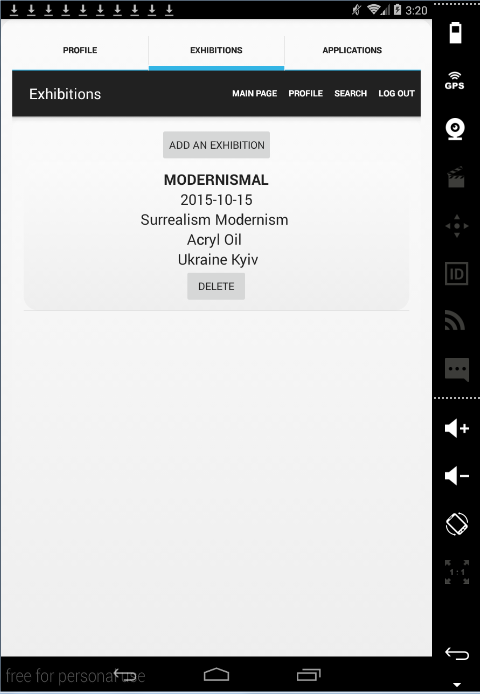
**Рисунок 4.15** – Форма авторизації ArtStep

У базі даних вже є користувач організатор виставок. Введіть у поле «Your login» «BilykIhor», а в поле «Your password» - «passBilyk». Натисніть кнопку «LOG IN». Відкриється форма, яку зображено на рисунку 4.16.



**Рисунок 4.16** – Форма профілю ArtStep

Перейдемо на вкладку «EXHIBITIONS». Відкриється форма, яку зображено на рисунку 4.17.



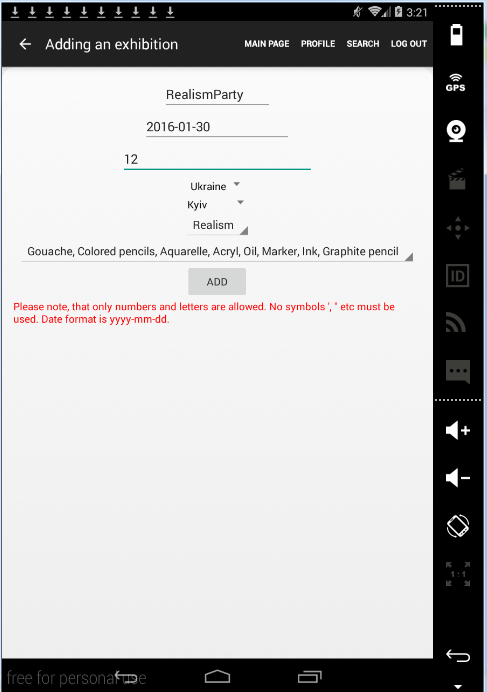
**Рисунок 4.17** – Форма виставок ArtStep

Переглянемо деталі виставки «MODERNISMAL». Для цього натиснемо на неї. Відкриється форма, яку зображено на рисунку 4.18.



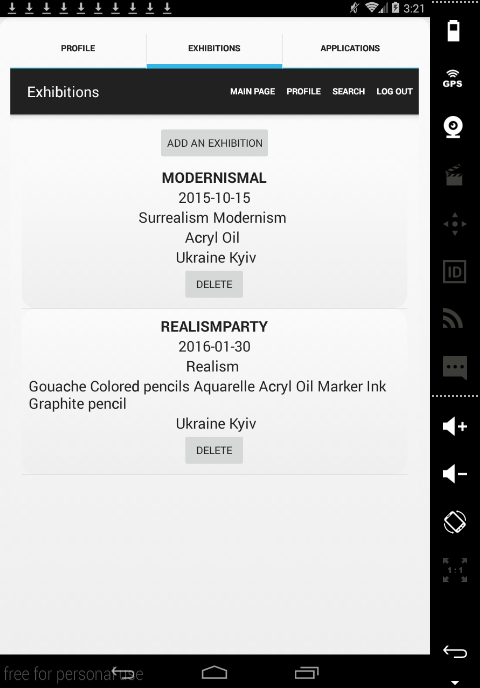
**Рисунок 4.18** – Форма перегляду виставки ArtStep

Додамо нову виставку. Для цього повернемося назад та натиснемо кнопку «ADD AN EXHIBITION». Відкриється форма, яку зображено на рисунку 4.19.



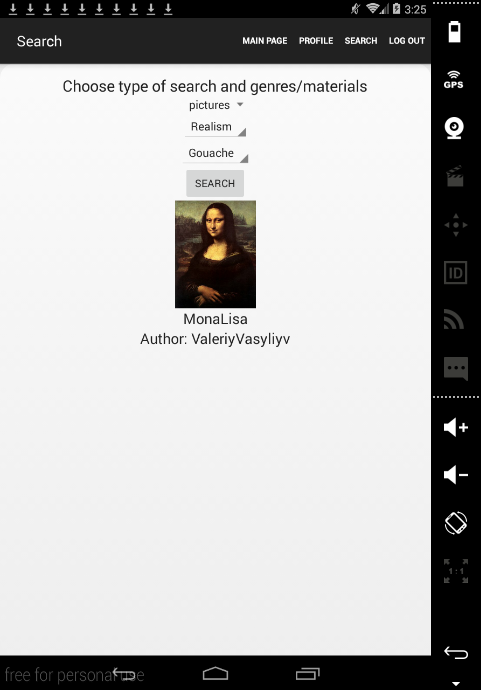
**Рисунок 4.19** – Форма додавання виставки ArtStep

Заповнимо поля та натиснемо кнопку «ADD». Переглянемо виставки знову, перейшовши на вкладку «EXHIBITIONS», отримаємо форму, зображену на рисунку 4.20.



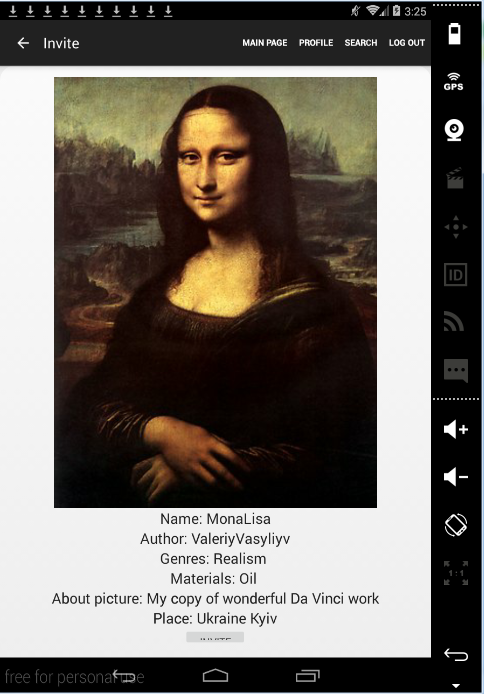
**Рисунок 4.20** – Форма перегляду виставок ArtStep

Тепер знайдемо учасників для нашої виставки. Для цього натиснемо кнопку «SEARCH» та оберемо критерії пошуку. Відкриється форма, яку зображено на рисунку 4.21.



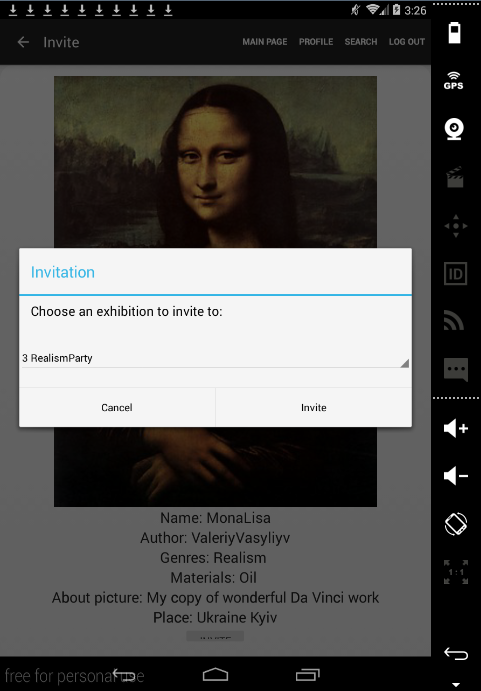
**Рисунок 4.21** – Форма пошуку ArtStep

Переглянемо детальніше знайдену картину, натиснувши на неї. Відкриється форма, яку зображено на рисунку 4.22.



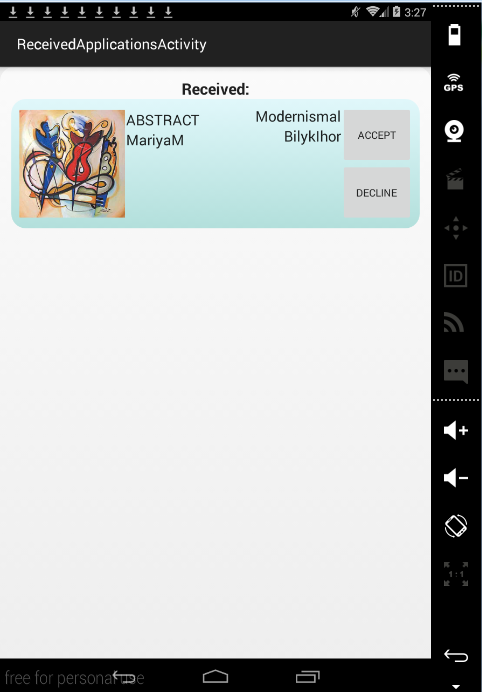
**Рисунок 4.22** – Форма перегляду картини ArtStep

Натиснемо кнопку «INVITE» для запрошення. Відкриється форма, яку зображено на рисунку 4.23.



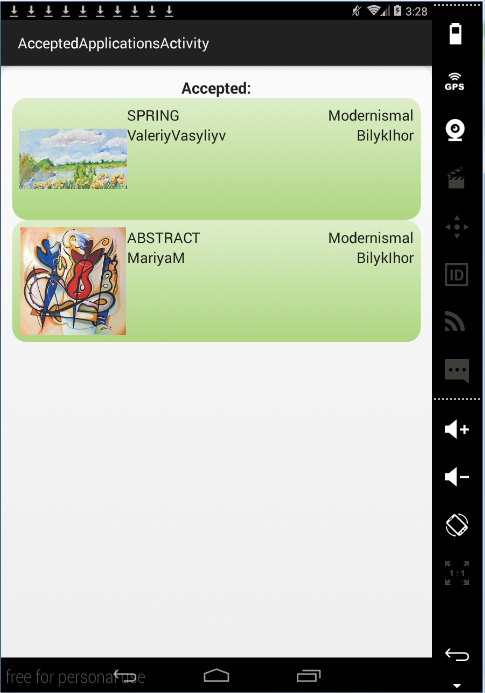
**Рисунок 4.23** – Форма запрошення ArtStep

Оберемо виставку «RealismParty» та запросимо картину. Тепер перейдемо до перегляду заявок. Для цього перейдемо по «PROFILE» - «APPLICATIONS» - «RECEIVED». Відкриється форма, яку зображено на рисунку 4.24.



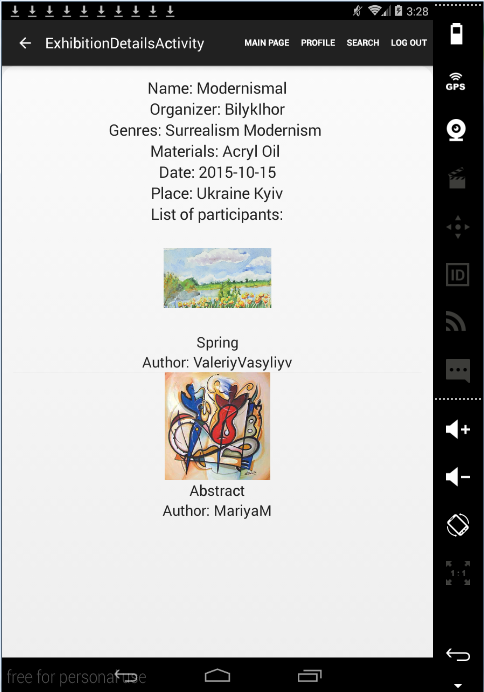
**Рисунок 4.24** – Перегляд отриманих заявок ArtStep

Натиснемо кнопку «ACCEPT». Після цього перейдемо назад та оберемо вкладку «ACCEPTED». Відкриється форма, яку зображено на рисунку 4.25.



**Рисунок 4.25** – Перегляд прийнятих заявок ArtStep

Тепер переглянемо інформацію про виставку «MODERNISMAL» для того, щоб побачити список учасників. Для цього перейдемо «PROFILE» - «EXHIBITIONS» - «MODERNISMAL». Відкриється форма, яку зображено на рисунку 4.26.



**Рисунок 4.26** – Перегляд учасників виставки ArtStep

### 4.2 Методика випробувань

Було виявлено тестові сценарії для всіх варіантів використання системи планування робіт на підприємстві.

Таблиця 4.1 – Опис тестових сценаріїв до системи планування робіт на підприємстві і результати їх виконання

|  |  |  |
| --- | --- | --- |
| Назва тесту | Тест 1. Тест автентифікації в системі під роллю співробітника технологічного відділу | |
| Функція\Use case | Автентифікація актора у системі | |
| **Дія** | **Очікуваний результат** | **Результат тесту**  **• пройдений**  **• провалений**  **• заблокований** |
| Передумова: | | |
| Перейти за адресою системи планування робіт на підприємстві | Система відкрита, користувачу доступні сторінки «Головна», «About», «Контакти», а також кнопки реєстрації та входу. | пройдений |
| Натиснути кнопку «Вхід» | Користувач бачить сторінку з формою автентифікації. | пройдений |
| Кроки тесту: | | |
| Заповнити поле «Логін» значенням «testaccount» | Дані успішно введені. | пройдений |
| Заповнити поле «Пароль» значенням «Test1234» | Дані успішно введені, користувач бачить кружечки замість введених символів. | пройдений |
| Натиснути кнопку «Увійти» | Успішна автентифікація під обліковим записом тестового співробітника технологічного відділу. Користувач бачить головну сторінку з привітанням «Привіт Іванов Іван Іванович!» (посилання на кабінет користувача) і кнопкою «Log off» замість кнопок реєстрації та входу. | пройдений |
| Післяумова: | | |
| Натиснучти кнопку «Log off» | Система переходить у початковий стан: користувач бачить головну сторінку, кнопки «Реєстрація» та «Вхід» доступні. | пройдений |
|  |  |  |
| Назва тесту | Тест 2. Тест автентифікації у системі під роллю головного інженера | |
| Функція\Use case | Автентифікація актора у системі | |
| **Дія** | **Очікуваний результат** | **Результат тесту**  **• пройдений**  **• провалений**  **• заблокований** |
| Передумова: | | |
| Перейти за адресою системи планування робіт на підприємстві | Система відкрита, користувачу доступні сторінки «Головна», «About», «Контакти», а також кнопки реєстрації та входу. | пройдений |
| Натиснути кнопку «Вхід» | Користувач бачить сторінку з формою автентифікації. | пройдений |
| Кроки тесту: | | |
| Заповнити поле «Логін» значенням «testengineer» | Дані успішно введені. | пройдений |
| Заповнити поле «Пароль» значенням «Test1234» | Дані успішно введені. | пройдений |
| Натиснути кнопку «Увійти» | Успішна автентифікація під обліковим записом тестового головного інженера. Користувач бачить головну сторінку з привітанням «Привіт Петренко Петро Петрович!» (посилання на кабінет користувача) і кнопкою «Log off» замість кнопок реєстрації та входу. | пройдений |
| Післяумова: | | |
| Натиснучти кнопку «Log off» | Система переходить у початковий стан: користувач бачить головну сторінку, кнопки «Реєстрація» та «Вхід» доступні. | пройдений |
|  |  |  |
| Назва тесту | Тест 3. Вихід із системи | |
| Функція\Use case | Вихід із системи | |
| **Дія** | **Очікуваний результат** | **Результат тесту**  **• пройдений**  **• провалений**  **• заблокований** |
| Передумова: | | |
| Перейти за адресою системи планування робіт на підприємстві | Система відкрита, користувачу доступні сторінки «Головна», «About», «Контакти», а також кнопки реєстрації та входу. | пройдений |
| Увійти у систему під одним із тестових облікових записів (Тести 1-2) | Користувач автентифікований у системі. | пройдений |
| Кроки тесту: | | |
| Натиснучти кнопку «Log off» | Система переходить у початковий стан: користувач бачить головну сторінку, кнопки «Реєстрація» та «Вхід» доступні. | пройдений |
|  |  |  |
| Назва тесту | Тест 4. Тест додавання виробу | |
| Функція\Use case | Ведення виробів | |
| **Дія** | **Очікуваний результат** | **Результат тесту**  **• пройдений**  **• провалений**  **• заблокований** |
| Передумова: | | |
| Перейти за адресою системи планування робіт на підприємстві | Система відкрита, користувачу доступні сторінки «Головна», «About», «Контакти», а також кнопки реєстрації та входу. | пройдений |
| Автентифікуватися у системі під обліковим записом тестового співробітника технологічного відділу (Тест 1) | Користувач автентифікований у системі під обліковим записом співробітника технологічного відділу. | пройдений |
| Перейти за посиланням на особистий кабінет користувача на головній панелі (посилання «Привіт Іванов Іван Іванович!») | Користувач бачить сторінку особистого кабінету співробітника технологічного відділу. | пройдений |
| Натиснути кнопку «Ведення виробів» | Відкрита сторінка ведення виробів. Користувач бачить список наявних у системі виробів. | пройдений |
| Кроки тесту: | | |
| Натиснути кнопку «Додати новий виріб» | Користувачу доступна форма додавання нового виробу. | пройдений |
| Заповнити обов’язкове поле «Назва» | Дані успішно введені. | пройдений |
| Заповнити обов’язкове поле «Дедлайн» | Дані успішно введені. | пройдений |
| Заповнити поле «Опис» | Дані успішно введені. | пройдений |
| Натиснути кнопку «Зберегти» | Відкрита сторінка ведення виробів. Новий виріб додано у кінець списку. | пройдений |
| Післяумова: | | |
| Натиснути на кнопку «Головна» | Відкрита головна сторінка системи. | пройдений |
|  |  |  |
| Назва тесту | Тест 5. Тест додавання операції | |
| Функція\Use case | Ведення операцій | |
| **Дія** | **Очікуваний результат** | **Результат тесту**  **• пройдений**  **• провалений**  **• заблокований** |
| Передумова: | | |
| Перейти за адресою системи планування робіт на підприємстві | Система відкрита, користувачу доступні сторінки «Головна», «About», «Контакти», а також кнопки реєстрації та входу. | пройдений |
| Автентифікуватися у системі під обліковим записом тестового співробітника технологічного відділу (Тест 1) | Користувач автентифікований у системі під обліковим записом співробітника технологічного відділу. | пройдений |
| Перейти за посиланням на особистий кабінет користувача на головній панелі (посилання «Привіт Іванов Іван Іванович!») | Користувач бачить сторінку особистого кабінету співробітника технологічного відділу. | пройдений |
| Натиснути кнопку «Ведення операцій» | Відкрита сторінка ведення операцій. Користувач бачить список наявних у системі операцій. | пройдений |
| Кроки тесту: | | |
| Натиснути кнопку «Додати нову операцію» | Користувачу доступна форма додавання нової операції. | пройдений |
| Заповнити обов’язкове поле «Назва» | Дані успішно введені. | пройдений |
| Заповнити поле «Опис» | Дані успішно введені. | пройдений |
| Натиснути кнопку «Зберегти» | Відкрита сторінка ведення операцій. Нову операцію додано у кінець списку. | пройдений |
| Післяумова: | | |
| Натиснути на кнопку «Головна» | Відкрита головна сторінка системи. | пройдений |
|  |  |  |
| Назва тесту | Тест 6. Тест додавання операції до технологічної карти виробу | |
| Функція\Use case | Ведення технологічних карт | |
| **Дія** | **Очікуваний результат** | **Результат тесту**  **• пройдений**  **• провалений**  **• заблокований** |
| Передумова: | | |
| Перейти за адресою системи планування робіт на підприємстві | Система відкрита, користувачу доступні сторінки «Головна», «About», «Контакти», а також кнопки реєстрації та входу. | пройдений |
| Автентифікуватися у системі під обліковим записом тестового співробітника технологічного відділу (Тест 1) | Користувач автентифікований у системі під обліковим записом співробітника технологічного відділу. | пройдений |
| Перейти за посиланням на особистий кабінет користувача на головній панелі (посилання «Привіт Іванов Іван Іванович!») | Користувач бачить сторінку особистого кабінету співробітника технологічного відділу. | пройдений |
| Натиснути кнопку «Ведення виробів» | Відкрита сторінка ведення виробів. Користувач бачить список наявних у системі виробів. | пройдений |
| Кроки тесту: | | |
| Натиснути кнопку «Технологічна карта» навпроти обраного виробу | Відкрита сторінка ведення технологічної карти обраного виробу. Користувач бачить список операцій у даній технологічній карті. | пройдений |
| Натиснути кнопку «Додати операцію» | Користувачу доступна форма додавання операції до технологічної карти виробу. | пройдений |
| Обрати операцію у полі «Операція» | Поле заповнено. | пройдений |
| Заповнити поле «Опис» | Поле заповнено. | пройдений |
| Натиснути кнопку «Зберегти» | Відкрита сторінка ведення технологічної карти. Операцію додано у кінець списку. | пройдений |
| Післяумова: | | |
| Натиснути на кнопку «Головна» | Відкрита головна сторінка системи. | пройдений |
|  |  |  |
| Назва тесту | Тест 7. Тест додавання нової дільниці | |
| Функція\Use case | Ведення дільниць | |
| **Дія** | **Очікуваний результат** | **Результат тесту**  **• пройдений**  **• провалений**  **• заблокований** |
| Передумова: | | |
| Перейти за адресою системи планування робіт на підприємстві | Система відкрита, користувачу доступні сторінки «Головна», «About», «Контакти», а також кнопки реєстрації та входу. | пройдений |
| Автентифікуватися у системі під обліковим записом тестового головного інженера (Тест 2) | Користувач автентифікований у системі під обліковим записом головного інженера. | пройдений |
| Перейти за посиланням на особистий кабінет користувача на головній панелі (посилання «Привіт Петренко Петро Петрович!») | Користувач бачить сторінку особистого кабінету головного інженера. | пройдений |
| Натиснути кнопку «Ведення дільниць» | Відкрита сторінка ведення дільниць. Користувач бачить список наявних у системі дільниць. | пройдений |
| Кроки тесту: | | |
| Натиснути кнопку «Додати нову дільницю» | Користувачу доступна форма додавання нової дільниці. | пройдений |
| Заповнити обов’язкове поле «Назва» | Дані успішно введені. | пройдений |
| Заповнити поле «Опис» | Дані успішно введені. | пройдений |
| Натиснути кнопку «Зберегти» | Відкрита сторінка ведення дільниць. Нову дільницю додано у кінець списку. | пройдений |
| Післяумова: | | |
| Натиснути на кнопку «Головна» | Відкрита головна сторінка системи. | пройдений |
|  |  |  |
| Назва тесту | Тест 8. Тест додавання нового приладу | |
| Функція\Use case | Ведення приладів | |
| **Дія** | **Очікуваний результат** | **Результат тесту**  **• пройдений**  **• провалений**  **• заблокований** |
| Передумова: | | |
| Перейти за адресою системи планування робіт на підприємстві | Система відкрита, користувачу доступні сторінки «Головна», «About», «Контакти», а також кнопки реєстрації та входу. | пройдений |
| Автентифікуватися у системі під обліковим записом тестового головного інженера (Тест 2) | Користувач автентифікований у системі під обліковим записом головного інженера. | пройдений |
| Перейти за посиланням на особистий кабінет користувача на головній панелі (посилання «Привіт Петренко Петро Петрович!») | Користувач бачить сторінку особистого кабінету головного інженера. | пройдений |
| Натиснути кнопку «Ведення приладів» | Відкрита сторінка ведення приладів. Користувач бачить список наявних у системі приладів. | пройдений |
| Кроки тесту: | | |
| Натиснути кнопку «Додати новий прилад» | Користувачу доступна форма додавання нового приладу. | пройдений |
| Заповнити обов’язкове поле «Назва» | Дані успішно введені. | пройдений |
| Заповнити поле «Опис» | Дані успішно введені. | пройдений |
| Обрати дільницю у полі «Дільниця» | Поле заповнено. | пройдений |
| Натиснути кнопку «Зберегти» | Відкрита сторінка ведення приладів. Новий прилад додано у кінець списку. | пройдений |
| Післяумова: | | |
| Натиснути на кнопку «Головна» | Відкрита головна сторінка системи. | пройдений |
|  |  |  |
| Назва тесту | Тест 9. Тест додавання операції до списку операцій виконуваних на приладі | |
| Функція\Use case | Визначення операцій виконуваних на приладах | |
| **Дія** | **Очікуваний результат** | **Результат тесту**  **• пройдений**  **• провалений**  **• заблокований** |
| Передумова: | | |
| Перейти за адресою системи планування робіт на підприємстві | Система відкрита, користувачу доступні сторінки «Головна», «About», «Контакти», а також кнопки реєстрації та входу. | пройдений |
| Автентифікуватися у системі під обліковим записом тестового головного інженера (Тест 2) | Користувач автентифікований у системі під обліковим записом головного інженера. | пройдений |
| Перейти за посиланням на особистий кабінет користувача на головній панелі (посилання «Привіт Петренко Петро Петрович!») | Користувач бачить сторінку особистого кабінету головного інженера. | пройдений |
| Натиснути кнопку «Ведення приладів» | Відкрита сторінка ведення приладів. Користувач бачить список наявних у системі приладів. | пройдений |
| Кроки тесту: | | |
| Натиснути кнопку «Виконувані операції» навпроти обраного приладу | Відкрита сторінка визначення операцій виконуваних на приладі. Користувач бачить список операцій, виконуваних для даного приладу. | пройдений |
| Натиснути кнопку «Додати операцію» | Користувачу доступна форма додавання операції до списку виконуваних операцій на приладі. | пройдений |
| Обрати операцію у полі «Операція» | Поле заповнено. | пройдений |
| Заповнити поле «Час виконання» | Дані успішно введені. | пройдений |
| Натиснути кнопку «Зберегти» | Відкрита сторінка визначення операцій виконуваних на приладі. Нову операцію додано у кінець списку. | пройдений |
| Післяумова: | | |
| Натиснути на кнопку «Головна» | Відкрита головна сторінка системи. | пройдений |
|  |  |  |
| Назва тесту | Тест 10. Тест побудови розкладу виробництва | |
| Функція\Use case | Аналіз розкладу виробництва | |
| **Дія** | **Очікуваний результат** | **Результат тесту**  **• пройдений**  **• провалений**  **• заблокований** |
| Передумова: | | |
| Перейти за адресою системи планування робіт на підприємстві | Система відкрита, користувачу доступні сторінки «Головна», «About», «Контакти», а також кнопки реєстрації та входу. | пройдений |
| Увійти у систему під одним із облікових записів (Тести 1-2) | Користувач автентифікований у системі. | пройдений. |
| Перейти за посиланням на особистий кабінет користувача на головній панелі | Користувач бачить сторінку особистого кабінету. | пройдений |
| Кроки тесту: | | |
| Натиснути кнопку «Аналіз розкладу виробництва» | Користувачу доступна сторінка для аналізу розкладу виробництва, де він бачить список виконуваних робіт на кожному приладі по дільницях. | пройдений |
| Післяумова: | | |
| Натиснути на кнопку «Головна» | Відкрита головна сторінка системи. | пройдений |

## Висновок

У даній роботі була вивчена тема побудови розкладу виробництва на підприємстві. Було поставлено за мету – розробити автоматизовану систему для ведення основної виробничої інформації підприємства та побудови розкладу робіт на приладах дільниць підприємства.

Виконуючи дане завдання, я:

* освоїв принцип представлення бізнес-процесів обраного предметного середовища за допомогою інструментів мови UML;
* удосконалив навички написання алгоритмів та реалізації застосувань з кросплатформним графічним інтерфейсом;
* закріпив навички роботи з реляційними базами даних;
* засвоїв створення програмного забезпечення за технологією TDD;
* продемонстрував навички проведення тестування розробленого програмного продукту;
* написав технічну документацію до створеного програмного продукту.

У підсумку можна сказати, що я досяг поставленої переді мною мети та виконав завдання розрахунково-графічної роботи з дисципліни «Технологія створення програмних продуктів», але тема повністю не вичерпана, і можна продовжувати удосконалювати програмний продукт, реалізовуючи нові функції.

# Список використаної літератури

1. Офіційна інтернет-сторінка розробників веб-застосувань за допомогою технології ASP.NET. [Електронний ресурс] Режим доступу: [http://www.asp.net//](http://developer.android.com/)
2. Приклади розробки діаграм UML від SparxSystems. [Електронний ресурс] Режим доступу: <http://www.sparxsystems.com/uml-tutorial.html>
3. Вивчення BPMN. [Електронний ресурс] Режим доступу: <http://www.bpmn.org/>
4. Вікіпедія – найбільша Інтернет-енциклопедія, стаття про UML. [Електронний ресурс] Режим доступу: <http://en.wikipedia.org/wiki/Unified_Modeling_Language>
5. Офіційний сайт розробників застосунків за допомогою IDE Microsoft Visual Studio. [Електронний ресурс] Режим доступу: [https://www.visualstudio.com/](http://developer.android.com/tools/studio/index.html)
6. Веб-сервери: IIS Express. [Електронний ресурс] Режим доступу: <http://www.iis.net/learn/extensions/introduction-to-iis-express/>
7. Офіційна сторінка СУБД Microsoft SQL Server. [Електронний ресурс] Режим доступу: https://www.microsoft.com/en-us/server-cloud/products/sql-server/
8. Електронний ресурс розробників клієнтської частини веб-застосунків. [Електронний ресурс] Режим доступу: http://www.w3schools.com/
9. Офіційний інтернет-сайт компанії DHTMLX для створення діаграм для веб-застосунків. [Електронний ресурс] Режим доступу:

[http:// http://dhtmlx.com/](http://developer.android.com/tools/testing/testing_android.html)

1. Офіційний інтернет-сайт розробників програмного забезпечення за допомогою технологій компанії Microsoft. [Електронний ресурс] Режим доступу: https://msdn.microsoft.com

# Додаток А

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ з/п** | **Назва етапів виконання розрахунково-графічної роботи** | **Строк виконання етапів проекту** | **Примітка** |
|  | *Постановка задачі* | *28.02.2016* |  |
|  | *Розробка бізнес-логіки* | *15.03.2016* |  |
|  | *Створення діаграм* | *17.03.2016* |  |
|  | *Створення моделі бази даних* | *05.04.2016* |  |
|  | *Налаштування веб-сервера* | *26.04.2016* |  |
|  | *Розробка програмного забезпечення* | *29.04.2016* |  |
|  | *Налагодження програми* | *15.05.2016* |  |
|  | *Оформлення роботи* | *27.05.2016* |  |

# ДОДАТОК Б

**ОСНОВНІ ЧАСТИНИ КОДУ ПРОГРАМИ**

public class Schedule<T> : List<T> where T : MachineSchedule, new()

{

/// <summary>

/// Default constructor

/// </summary>

public Schedule()

{

}

/// <summary>

/// Constructor based on machines

/// </summary>

/// <param name="machines"></param>

public Schedule(IEnumerable<Machine> machines)

{

var instance = new T();

foreach (var machine in machines)

{

Add(instance.Create(machine) as T);

}

}

/// <summary>

///

/// </summary>

/// <param name="machineSchedules"></param>

public Schedule(IEnumerable<T> machineSchedules)

{

AddRange(machineSchedules);

}

public string GetJson()

{

var data = new List<object>();

var links = new List<object>();

var currentId = 1;

var linkId = 10;

foreach (var machine in this.OrderBy(x => x.Machine.DepartmentId).ThenBy(x => x.Machine.Id))

{

if (machine.Tasks.Count != 0)

{

var parentId = currentId++;

var startTime = machine.StartTime;

var previousId = 0;

var duration = 0d;

foreach (var task in machine.Tasks)

{

data.Add(new

{

id = currentId,

text = task.TaskName,

start\_date = dateString(startTime),

duration = string.Format("{0}", task.Duration/60),

parent = parentId.ToString(),

progress = 0,

open = true

});

startTime = startTime.AddSeconds(task.Duration);

if (previousId != 0)

{

links.Add(new

{

id = linkId.ToString(),

source = previousId.ToString(),

target = currentId.ToString(),

type = "0"

});

++linkId;

}

previousId = currentId;

duration += task.Duration;

currentId++;

}

data.Add(new

{

id = parentId,

text = string.Format("{0} (дільниця \"{1}\")", machine.Machine.Name, machine.Machine.DepartmentName),

start\_date = dateString(machine.StartTime),

duration = string.Format("{0}", duration / 60),

progress = 1,

open = true

});

}

}

return new JavaScriptSerializer().Serialize(new {data = data, links = links});

}

private static string dateString(DateTime date)

{

return string.Format("{0:D2}-{1:D2}-{2} {3:D2}:{4:D2}", date.Day, date.Month, date.Year, date.Hour, date.Minute);

}

/// <summary>

///

/// </summary>

/// <param name="tasks"></param>

/// <param name="machines"></param>

/// <returns></returns>

public static Schedule<MachineSchedule> BuildSchedule(IEnumerable<ScheduleTask> tasks, IEnumerable<Machine> machines)

{

#region Validate arguments

if (tasks == null)

{

throw new ArgumentNullException();

}

if (machines == null)

{

throw new ArgumentNullException();

}

var tasksList = (tasks as List<ScheduleTask>) ?? tasks.ToList();

var machinesList = (machines as List<Machine>) ?? machines.ToList();

if (tasksList.Count == 0 || machinesList.Count == 0)

{

return new Schedule<MachineSchedule>(machinesList);

}

#endregion

tasksList.Sort((x, y) =>

{

var result = x.ExtremeTime.CompareTo(y.ExtremeTime);

return result == 0 ? x.Duration.CompareTo(y.Duration) : result;

});

var initSchedule = initialSchedule(tasksList, machinesList);

// Initial schedule has been found

if (initSchedule != null)

{

if (initSchedule.AppointedTasks.Count == tasksList.Count)

{

return initSchedule.Convert();

}

// Algorithm A1.1

var success = true;

var sortedSet = new List<InitialMachineSchedule>(initSchedule);

foreach (var currentTask in tasksList)

{

if (initSchedule.AppointedTasks.Contains(currentTask.TechnologyId))

{

continue;

}

InitialMachineSchedule min = null;

foreach (var machine in sortedSet)

{

if (currentTask.CompatibleDepartments.Contains(machine.Machine.DepartmentId) &&

(min == null || machine.EndTime < min.EndTime))

{

min = machine;

}

}

if (min == null)

{

success = false;

break;

}

if (min.Tasks.Count == 0)

{

min.StartTime = currentTask.ExtremeTime;

min.EndTime = currentTask.Deadline;

min.Tasks.AddFirst(currentTask);

continue;

}

var newEndTime = min.EndTime.AddSeconds(currentTask.Duration);

if (newEndTime > currentTask.Deadline)

{

success = false;

break;

}

min.Tasks.AddLast(currentTask);

min.EndTime = min.EndTime.AddSeconds(currentTask.Duration);

}

if (success)

{

return (new InitialSchedule(sortedSet).Convert());

}

}

// Algorithm A2.1

// Sorg by (d, l)

tasksList.Sort((x, y) =>

{

var result = x.Deadline.CompareTo(y.Deadline);

return result == 0 ? x.Duration.CompareTo(y.Duration) : result;

});

var machineSchedules = new Schedule<MachineSchedule>(machinesList);

var n = tasksList.Count;

// Building schedule

for (var i = n - 1; i >= 0; --i)

{

// Find unallowable with minimal start time

var currentTask = tasksList[i];

MachineSchedule targetMachine = null;

MachineSchedule lastMachine = null;

foreach (var schedule in machineSchedules)

{

if (currentTask.CompatibleDepartments.Contains(schedule.Machine.DepartmentId))

{

if (lastMachine == null || lastMachine.StartTime < schedule.StartTime)

{

lastMachine = schedule;

}

if (!(currentTask.Deadline > schedule.StartTime) &&

(targetMachine == null || schedule.StartTime < targetMachine.StartTime))

{

targetMachine = schedule;

}

}

}

if (lastMachine == null)

{

continue;

}

// If founded

if (targetMachine != null)

{

targetMachine.Tasks.AddFirst(currentTask);

targetMachine.StartTime = currentTask.ExtremeTime;

continue;

}

// Else take machine with max start time, find allowable task with max duration

ScheduleTask longestTask = null;

var index = i;

for (var j = i; j >= 0; --j)

{

if (tasksList[j].CompatibleDepartments.Contains(lastMachine.Machine.DepartmentId) &&

!(tasksList[j].Deadline < lastMachine.StartTime) &&

(longestTask == null || tasksList[j].Duration > longestTask.Duration))

{

longestTask = tasksList[j];

index = j;

}

}

lastMachine.Tasks.AddFirst(longestTask);

lastMachine.StartTime = lastMachine.StartTime.AddSeconds(-longestTask.Duration);

// Remove appointed task

tasksList.RemoveAt(index);

}

return new Schedule<MachineSchedule>(machineSchedules);

}

/// <summary>

/// Initial schedule

/// </summary>

/// <param name="tasks"></param>

/// <param name="machines"></param>

/// <returns></returns>

private static InitialSchedule initialSchedule(List<ScheduleTask> tasks, List<Machine> machines)

{

var schedule = new InitialSchedule(machines);

var engaged = new List<int>();

foreach (var currentTask in tasks)

{

int? target = null;

foreach (var id in engaged)

{

if (!currentTask.CompatibleDepartments.Contains(schedule[id].Machine.DepartmentId) || schedule[id].EndTime > currentTask.ExtremeTime)

{

continue;

}

if (target != null)

{

return null;

}

target = id;

}

if (target != null)

{

var t = (int)target;

schedule[t].Tasks.AddLast(currentTask);

schedule[t].EndTime = schedule[t].EndTime.AddSeconds(currentTask.Duration);

schedule.AppointedTasks.Add(currentTask.TechnologyId);

continue;

}

if (engaged.Count == machines.Count)

{

return schedule;

}

// engage next processor

int? index = null;

for (var i = 0; i < schedule.Count; i++)

{

if (!engaged.Contains(i) && currentTask.CompatibleDepartments.Contains(schedule[i].Machine.DepartmentId))

{

index = i;

break;

}

}

if (index == null)

{

continue;

}

var forEngage = schedule[(int)index];

forEngage.StartTime = currentTask.ExtremeTime;

forEngage.Tasks.AddLast(currentTask);

forEngage.EndTime = currentTask.Deadline;

schedule.AppointedTasks.Add(currentTask.TechnologyId);

engaged.Add((int)index);

}

return schedule;

}

/// <summary>

/// Initial machine schedule

/// </summary>

private class InitialMachineSchedule : MachineSchedule

{

public DateTime EndTime { get; set; }

public InitialMachineSchedule()

{

EndTime = DateTime.MinValue;

}

private InitialMachineSchedule(Machine machine) : base(machine)

{

EndTime = DateTime.MinValue;

}

public override MachineSchedule Create(Machine machine)

{

return new InitialMachineSchedule(machine);

}

}

/// <summary>

/// Initial schedule

/// </summary>

private class InitialSchedule : Schedule<InitialMachineSchedule>

{

/// <summary>

///

/// </summary>

public List<int> AppointedTasks { get; set; }

/// <summary>

/// Constructor

/// </summary>

/// <param name="machines"></param>

public InitialSchedule(IEnumerable<Machine> machines) : base(machines)

{

AppointedTasks = new List<int>();

}

/// <summary>

///

/// </summary>

/// <param name="machineSchedules"></param>

public InitialSchedule(IEnumerable<Schedule<T>.InitialMachineSchedule> machineSchedules) : base(machineSchedules)

{

AppointedTasks = new List<int>();

}

/// <summary>

///

/// </summary>

/// <returns></returns>

public Schedule<MachineSchedule> Convert()

{

var result = new Schedule<MachineSchedule>();

result.AddRange(this.Cast<MachineSchedule>());

return result;

}

}

private class EndTimeComparer : IComparer<InitialMachineSchedule>

{

public int Compare(InitialMachineSchedule x, InitialMachineSchedule y)

{

return x.EndTime == y.EndTime ? x.Machine.Id.CompareTo(y.Machine.Id) : x.EndTime.CompareTo(y.EndTime);

}

}

private class StartTimeComparer : IComparer<MachineSchedule>

{

public int Compare(MachineSchedule x, MachineSchedule y)

{

return x.StartTime == y.StartTime ? x.Machine.Id.CompareTo(y.Machine.Id) : x.StartTime.CompareTo(y.StartTime);

}

}

}

public class Repository

{

private static string \_connectionString;

public static string DefaultConnection

{

get

{

if (string.IsNullOrEmpty(\_connectionString))

{

\_connectionString = ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString;

}

return \_connectionString;

}

}

public static List<IdentityRole<string, IdentityUserRole>> GetAvailableRoles()

{

var result = new List<IdentityRole<string, IdentityUserRole>>();

using (var connection = new SqlConnection(DefaultConnection))

{

connection.Open();

using (var cmd = new SqlCommand())

{

cmd.CommandText = @"select Id, Name from AspNetRoles";

cmd.Connection = connection;

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

result.Add(new IdentityRole()

{

Id = reader.GetFieldValue<string>(0),

Name = reader.GetFieldValue<string>(1)

});

}

}

}

connection.Close();

}

return result;

}

public static List<Item> GetItems(Item instance)

{

var result = new List<Item>();

using (var connection = new SqlConnection(DefaultConnection))

{

connection.Open();

using (var cmd = new SqlCommand())

{

var text = instance is Machine

? @"select p.Id, p.Name, p.Description, p.C\_Date, p.C\_User, u.UserName, d.Id, d.Name from {0}s p

left join AspNetUsers u on p.C\_User = u.Id left join Departments d on d.Id = p.DepartmentId"

: instance is Product

? @"select p.Id, p.Name, p.Description, p.C\_Date, p.C\_User, u.UserName, p.Deadline from {0}s p

left join AspNetUsers u on p.C\_User = u.Id"

: @"select p.Id, p.Name, p.Description, p.C\_Date, p.C\_User, u.UserName from {0}s p

left join AspNetUsers u on p.C\_User = u.Id";

cmd.CommandText = string.Format(text, instance.InheritorName);

cmd.Connection = connection;

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

var item = instance.Create(

reader.GetFieldValue<int>(0),

reader.GetFieldValue<string>(1),

reader.GetFieldValue<string>(2),

reader.IsDBNull(3) ? DateTime.MinValue : reader.GetFieldValue<DateTime>(3),

reader.IsDBNull(4) ? string.Empty : reader.GetFieldValue<string>(4),

reader.IsDBNull(5) ? string.Empty : reader.GetFieldValue<string>(5)

);

if (instance is Machine && !reader.IsDBNull(6))

{

var machine = (Machine)item;

machine.DepartmentId = reader.GetFieldValue<int>(6);

machine.DepartmentName = reader.GetFieldValue<string>(7);

}

if (instance is Product && !reader.IsDBNull(6))

{

var product = (Product)item;

product.Deadline = reader.GetFieldValue<DateTime>(6);

}

result.Add(item);

}

}

}

connection.Close();

}

return result;

}

public static bool DeleteItem(int id, string inheritorName)

{

var result = false;

using (var connection = new SqlConnection(DefaultConnection))

{

connection.Open();

using (var cmd = new SqlCommand())

{

cmd.CommandText = string.Format(@"delete {0}s where Id = @idParam", inheritorName);

cmd.Connection = connection;

cmd.Parameters.AddWithValue("idParam", id);

if (cmd.ExecuteNonQuery() > 0)

{

result = true;

}

}

connection.Close();

}

return result;

}

public static void UpdateItems(IEnumerable<Item> items)

{

var itemsArray = (items as List<Item>) ?? items.ToList();

if (itemsArray.Count == 0)

{

return;

}

var instance = itemsArray[0];

var isMachine = instance is Machine;

var isProduct = instance is Product;

var stringBuilder = new StringBuilder();

using (var connection = new SqlConnection(DefaultConnection))

{

connection.Open();

using (var cmd = new SqlCommand())

{

stringBuilder.AppendLine("BEGIN TRANSACTION UpdateItems");

var i = 0;

foreach (var p in itemsArray)

{

++i;

var nameParam = string.Format("nameParam{0}", i);

var descriptionParam = string.Format("descriptionParam{0}", i);

var idParam = string.Format("idParam{0}", i);

var departmentParam = string.Empty;

if (isMachine)

{

departmentParam = string.Format("departmentParam{0}", i);

var id = ((Machine)p).DepartmentId;

cmd.Parameters.AddWithValue(departmentParam, id == 0 ? (object)DBNull.Value : id);

}

var deadlineParam = string.Empty;

if (isProduct)

{

deadlineParam = string.Format("deadlineParam{0}", i);

var deadline = ((Product) p).Deadline;

cmd.Parameters.AddWithValue(deadlineParam, deadline ?? (object) DBNull.Value);

}

stringBuilder.AppendLine(

string.Format("update {0}s set Name = @{1}, Description = @{2}{3} where Id = @{4}",

instance.InheritorName,

nameParam, descriptionParam,

isMachine

? string.Format(", DepartmentId = @{0}", departmentParam)

: isProduct

? string.Format(", Deadline = @{0}", deadlineParam)

: string.Empty,

idParam));

cmd.Parameters.AddRange(new[]

{

new SqlParameter(nameParam, p.Name ?? string.Empty),

new SqlParameter(descriptionParam, p.Description ?? string.Empty),

new SqlParameter(idParam, p.Id)

});

}

stringBuilder.AppendLine("COMMIT TRANSACTION UpdateItems");

cmd.CommandText = stringBuilder.ToString();

cmd.Connection = connection;

cmd.ExecuteNonQuery();

}

connection.Close();

}

}

public static void CreateItem(Item item, string userId)

{

var machine = item as Machine;

var product = item as Product;

using (var connection = new SqlConnection(DefaultConnection))

{

connection.Open();

using (var cmd = new SqlCommand())

{

cmd.CommandText = string.Format(

@"insert {0}s (Name, Description, C\_Date, C\_User{1}) values (@nameParam, @descrParam, @dateParam, @userParam{2})",

item.InheritorName,

machine != null ? ", DepartmentId"

: product != null ? ", Deadline"

: string.Empty,

machine != null ? ", @departmentParam"

: product != null ? ", @deadlineParam"

: string.Empty);

cmd.Connection = connection;

cmd.Parameters.AddWithValue("nameParam", item.Name);

cmd.Parameters.AddWithValue("descrParam", item.Description ?? string.Empty);

cmd.Parameters.AddWithValue("dateParam", DateTime.Now);

cmd.Parameters.AddWithValue("userParam", userId);

if (machine != null)

{

cmd.Parameters.AddWithValue("departmentParam", machine.DepartmentId == 0 ? (object)DBNull.Value : machine.DepartmentId);

}

if (product != null)

{

cmd.Parameters.AddWithValue("deadlineParam", product.Deadline ?? (object) DBNull.Value);

}

cmd.ExecuteNonQuery();

}

connection.Close();

}

}

public static List<ScheduleTask> GetScheduleTasks()

{

var result = new List<ScheduleTask>();

using (var connection = new SqlConnection(DefaultConnection))

{

connection.Open();

using (var cmd = new SqlCommand())

{

cmd.CommandText =

@"select tc.Id, tc.Duration, p.Id, p.Name, p.Deadline, t.Id, t.Name, tc.Description, c.DepartmentId

from Technologies tc

inner join Tasks t on tc.TaskId = t.Id

inner join Products p on p.Id = tc.ProductId

inner join Compatibilities c on c.TaskId = tc.TaskId";

cmd.Connection = connection;

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

var technologyId = reader.GetFieldValue<int>(0);

var item = result.Find(x => x.TechnologyId == technologyId);

if (item == null)

{

item = new ScheduleTask(technologyId,

reader.GetFieldValue<double>(1),

reader.GetFieldValue<int>(2),

reader.GetFieldValue<string>(3),

reader.GetFieldValue<DateTime>(4),

reader.GetFieldValue<int>(5),

reader.GetFieldValue<string>(6),

reader.GetFieldValue<string>(7)

);

result.Add(item);

}

item.CompatibleDepartments.Add(reader.GetFieldValue<int>(8));

}

}

}

connection.Close();

}

return result;

}

public static Technologies GetTechnologies(int productId)

{

var result = new Technologies();

result.ProductId = productId;

using (var connection = new SqlConnection(DefaultConnection))

{

connection.Open();

using (var cmd = new SqlCommand())

{

cmd.CommandText = string.Format(

@"select Id, TaskId, Description, Duration from Technologies where ProductId = @idParam");

cmd.Parameters.AddWithValue("idParam", productId);

cmd.Connection = connection;

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

result.Add(new Technology()

{

Id = reader.GetFieldValue<int>(0),

TaskId = reader.GetFieldValue<int>(1),

Description = reader.GetFieldValue<string>(2),

Duration = reader.GetFieldValue<double>(3),

ProductId = productId

});

}

}

}

using (var cmd = new SqlCommand())

{

cmd.CommandText = @"select Name from Products where Id = @idParam";

cmd.Parameters.AddWithValue("idParam", productId);

cmd.Connection = connection;

using (var reader = cmd.ExecuteReader())

{

if (reader.Read())

{

result.ProductName = reader.GetFieldValue<string>(0);

}

}

}

connection.Close();

}

return result;

}

public static void UpdateTechnologies(IEnumerable<Technology> technologies)

{

var stringBuilder = new StringBuilder();

using (var connection = new SqlConnection(DefaultConnection))

{

connection.Open();

using (var cmd = new SqlCommand())

{

stringBuilder.AppendLine("BEGIN TRANSACTION UpdateTechnologies");

var i = 0;

foreach (var t in technologies)

{

++i;

var taskParam = string.Format("taskParam{0}", i);

var descriptionParam = string.Format("descriptionParam{0}", i);

var durationParam = string.Format("durationParam{0}", i);

var idParam = string.Format("idParam{0}", i);

stringBuilder.AppendLine(string.Format("update Technologies set TaskId = @{0}, Description = @{1}, Duration = @{2} where Id = @{3}",

taskParam, descriptionParam, durationParam, idParam));

cmd.Parameters.AddRange(new[]

{

new SqlParameter(taskParam, t.TaskId),

new SqlParameter(descriptionParam, t.Description ?? string.Empty),

new SqlParameter(idParam, t.Id),

new SqlParameter(durationParam, t.Duration)

});

}

stringBuilder.AppendLine("COMMIT TRANSACTION UpdateTechnologies");

cmd.CommandText = stringBuilder.ToString();

cmd.Connection = connection;

cmd.ExecuteNonQuery();

}

connection.Close();

}

}

public static void CreateTechnology(Technology technology)

{

using (var connection = new SqlConnection(DefaultConnection))

{

connection.Open();

using (var cmd = new SqlCommand())

{

cmd.CommandText = @"insert Technologies (ProductId, TaskId, Description, Duration) values (@productParam, @taskParam, @descrParam, @durParam)";

cmd.Connection = connection;

cmd.Parameters.AddWithValue("productParam", technology.ProductId);

cmd.Parameters.AddWithValue("taskParam", technology.TaskId);

cmd.Parameters.AddWithValue("descrParam", technology.Description ?? string.Empty);

cmd.Parameters.AddWithValue("durParam", technology.Duration);

cmd.ExecuteNonQuery();

}

connection.Close();

}

}

public static bool DeleteTechnology(int id)

{

var result = false;

using (var connection = new SqlConnection(DefaultConnection))

{

connection.Open();

using (var cmd = new SqlCommand())

{

cmd.CommandText = "delete Technologies where Id = @idParam";

cmd.Connection = connection;

cmd.Parameters.AddWithValue("idParam", id);

if (cmd.ExecuteNonQuery() > 0)

{

result = true;

}

}

connection.Close();

}

return result;

}

public static List<Compatibility> GetCompatibilities(int departmentId)

{

var result = new List<Compatibility>();

using (var connection = new SqlConnection(DefaultConnection))

{

connection.Open();

using (var cmd = new SqlCommand())

{

cmd.CommandText = @"select t.Id, t.Name, t.Description, t.C\_Date, t.C\_User, u.UserName, c.Id from Tasks t

left join AspNetUsers u on t.C\_User = u.Id

left join Compatibilities c on c.TaskId = t.Id and c.DepartmentId = @departmentParam";

cmd.Parameters.AddWithValue("departmentParam", departmentId);

cmd.Connection = connection;

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

var task = new Task(reader.GetFieldValue<int>(0), reader.GetFieldValue<string>(1),

reader.GetFieldValue<string>(2), reader.GetFieldValue<DateTime>(3),

null, string.Empty);

if (!reader.IsDBNull(4))

{

task.CreationUserId = reader.GetFieldValue<string>(4);

task.CreationUserLogin = reader.GetFieldValue<string>(5);

}

result.Add(new Compatibility

{

Task = task,

IsCompatible = !reader.IsDBNull(6)

});

}

}

}

connection.Close();

}

return result;

}

public static void ChangeCompatibility(int departmentId, int taskId, bool compatible)

{

using (var connection = new SqlConnection(DefaultConnection))

{

connection.Open();

using (var cmd = new SqlCommand())

{

cmd.CommandText = compatible

? "insert Compatibilities (DepartmentId, TaskId) values (@departmentParam, @taskParam)"

: "delete Compatibilities where DepartmentId = @departmentParam and TaskId = @taskParam";

cmd.Connection = connection;

cmd.Parameters.AddWithValue("departmentParam", departmentId);

cmd.Parameters.AddWithValue("taskParam", taskId);

cmd.ExecuteNonQuery();

}

connection.Close();

}

}

public static Department GetDepartment(int id)

{

Department result = null;

using (var connection = new SqlConnection(DefaultConnection))

{

connection.Open();

using (var cmd = new SqlCommand())

{

cmd.CommandText = @"select d.Name, d.Description, d.C\_Date, d.C\_User, u.UserName from Departments d

left join AspNetUsers u on d.C\_User = u.Id where d.Id = @idParam";

cmd.Parameters.AddWithValue("idParam", id);

cmd.Connection = connection;

using (var reader = cmd.ExecuteReader())

{

if (reader.Read())

{

result = new Department(id, reader.GetFieldValue<string>(0), reader.GetFieldValue<string>(1),

reader.GetFieldValue<DateTime>(2), null, string.Empty);

if (!reader.IsDBNull(3))

{

result.CreationUserId = reader.GetFieldValue<string>(3);

result.CreationUserLogin = reader.GetFieldValue<string>(4);

}

}

}

}

connection.Close();

}

return result;

}

}