Індустріально-педагогічний технікум КІСумДУ

Циклова комісія комп’ютерних дисциплін

# **КУРСОВА РОБОТА**

**з предмету: Конструювання програмного забезпечення**

на тему: «Створення інформаційної системи Відеопрокат»

Виконав студент четвертого курсу,

741 групи

напряму підготовки «Інформатика та обчислювальна техніка»

спеціальності «Розробка програмного забезпечення»

Мицай Владислав Олександрович

(прізвище та ініціали)

Керівник: Бараболіков В.М.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

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

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

м. Конотоп, 2018 рік

Міністерство освіти і науки України

ІНДУСТРІАЛЬНО - ПЕДАГОГІЧНИЙ ТЕХНІКУМ КІ СумДУ

**«Затверджую»**

**Протокол №5 від 23.02.2018**

Голова ц/к комп’ютерних дисциплін

Бараболіков В.М.

**Завдання на курсову роботу**

Створення інформаційної системи Автовокзал

Розробити:

1. Зручний користувальницький інтерфейс;
2. Зазначені методи й уможливити підключення інших методів, оцінити точність кожного методу;
3. Повне тестування ППП;
4. Документацію на ППП;
5. Контрольні приклади.

Дата видачі завдання "\_2 " лютого 2018р.

Керівник: Бараболіков В.М.

Виконавець: студент групи 741 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Термін захисту проекту "26"\_\_квітня\_\_\_2018р.

Конотоп,2018р.

**Зміст**

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

3

5.05010301 5 1 14. вар 7 ПЗ

Розроб.

Мицай В.О.

Перевір.

Бараболіков В.М

Реценз.

Н. Контр.

БараболіковВ.М

Затверд.

Створення інформаційної системи Автовокзал

Літ.

Аркушів

34

ІПТ КІ СумДУ

Вступ ………………………………………………………………….. 4

1.Технічне завдання ………………………………………. 5

2. Моделювання програмного забезпечення …………….. 9

2.1 Діаграма варіантів використання …………………. 9

2.2 Діаграма прецедентів ……………………………. … 12

2.3 Діаграма класів ………………………………………. 14

3. Розробка інтерфейсу користувача……………………... 19

3.1 Загальні рекомендації ………………………..…… 19

3.2 Багато віконні та багато сторінкові додатки ……. 19

3.3 Кольорове оформлення додатків …………..…...... 21

3.4 Шрифти текстів ……………………………..…….. 21

4. Опис використаних класів у програмі ………………… 22

5. Тестування ………………………………………………. 25

6. Публікація проекту ……………………………………… 29

Висновок ……………………………………………………. 35

Додаток А. Лістинг ……………………………………….. 36

**Вступ**

В даний час неможливо уявити собі успішну і ефективну діяльність якої б то не було галузі виробництва без використання сучасних засобів обчислювальної техніки, в першу чергу це стосується комп'ютерів.

У період загальної комп'ютеризації діловодство переноситься з паперових носіїв на електронні, що, безсумнівно, набагато зручніше, ефективніше, швидше для роботи службовців підприємств, організацій, установ. Невід'ємною частиною обліку діяльності організації є наявність баз даних, що зберігають різну інформацію про функціонування підприємства, її співробітників, ділових партнерів, вироблених матеріалах, документах.

Перехід від традиційних методів обробки інформації має ряд важливих переваг. По-перше, це автоматизація, як самого процесу обробки, так і зберігання даних в будь-якої необхідної формі. По-друге, це робота з практично необмеженим обсягом даних. Таким чином, на сьогоднішній є досить актуальною розробка та створення додатків для підприємств, організацій, фірм.

У роботі буде розглянуто процес створення додатку ІС "Автовокзал".

Ця програма має такі переваги, зокрема:

1. зручний і досить простий інтерфейс у використанні;
2. забезпечує ефективну роботу з базою даних;
3. забезпечує діалогове спілкування між комп'ютером і користувачем;
4. економить час робітників.

**Технічне завдання**

## **Засади розробки Проекту**

Інформаційної система Автовокзалу розробляється на вимогу завдання до курсової роботи з предмету КПЗ та з метою полегшення роботи працівників Автовокзалу.

## **2. Мета розробки Проекту**

Метою розробки проекту є:

* ефективно діюча система відображення доступних функцій;
* система моніторингу, реєстрація квитків, видачі, оплати, статистики;
* забезпечення стабільної роботи програми;

## **3. Призначення Проекту**

Полегшення роботи працівників Автовокзалу.

## **4. Вимоги до функціональності проекту**

4.1 Проект створюється у середовищі розробки програмного забезпечення Delphi 7.

* + 1. Система повинна виконувати функції:
* Ведення моніторингу оплати, реєстрації квитків, видачі, оплати, статистики;
* Створювати статистику та підсумки;
* забезпечення підтримки мульти-мовного інтерфейсу
* забезпечення мультиплатформенності проекту;
  + 1. Система повинна забезпечувати такі функції адміністрування:
* виключення невірно введеної інформації;
* забезпечувати автоматичне заповнення картки видачі;
* збір і обробка даних;

4.2 Вимоги до адміністративного інтерфейсу системи:

4.2.1 Інтерфейс системи має бути зрозумілим будь якому користувачі незалежно від рівня володіння комп’ютером;

4.2.2 Інтерфейс повинен бути сприятливий та розрахований на користувача певної кваліфікації;

4.3 Вимоги до адміністрування системи:

4.3.1 Забезпечення віддаленого адміністрування проекту;

4.3.2 Система повинна врахувати можливість одночасної роботи адміністратора та користувача.

## **5. Структура програми і вимоги до її представлення**

5.1 Основною одиницею структури програми є введення даних, перегляд даних, можливість редагувати данні, розрахунок вартості товару.

5.2 Структура програми повинна передбачати такі розділи:

* Місце реєстрації клієнта ;
* Розділ із можливістю додавання рейсів;
* Загальна інформація про рейси, клієнтів.

## **6. Вимоги до операцій з адміністрування та наповнення програми**

6.1 Система адміністрування програми повинна зберігати всю потрібну інформацію та надавати можливість керувати нею.

## **7.** **Вимоги до графічного дизайну програми**

7.1 У графічному дизайні проекту мають використовуватися легкі для сприйняття кольори .

Шрифт та розмір надписів у програмі виконані в одному стилі та зручні для користувача.

Дизайн розділів програми повинен задовольняти таким параметрам:

* Єдиний стиль представлення;

## **8. Вимоги до зручності користування програмою**

8.1 Використовувати простий зручний шрифт .

Не бажано використовувати не чіткі зображення в проекті;

## **9. Вимоги до програмно - технічного забезпечення Проекту**

9.1 Вимоги до складу і параметрів технічних засобів

1. система повинна працювати на IBM сумісних персональних комп'ютерах;
2. мінімальна конфігурація: тип процесора Pentium і вище; об'єм оперативного запам'ятовуючого пристрою 32 Mb і більше;

9.2 Вимоги до інформаційної та програмної сумісності

1. код програми повинен бути реалізований на мові програмування Delphi.
2. В якості інтегрованого середовища розробки програми повинно бути використане середовище BorlandDelphi 7.
3. Програма повинна працювати під управлінням операційних систем WindowsXP та більш нових версій.

## **10. Вимоги до документації Проекту**

Розроблений програмний продукт повинен бути задокументований, коди програми повинні мати всі необхідні коментарі. Розроблена програма повинна включати довідкову інформацію про основні можливості.

До складу даної документації повинні входити: пояснювальна записка , яка має описання розробки програми та керівництво користувача.

## **11. Вимоги до процесу розробки Проекту**

11.1 Розробка повинна бути проведена в три стадії:

1. Розробка технічного завдання;
2. Робоче проектування;
3. Впровадження програмного продукту;

11.2 На стадії розробки технічного завдання повинен бути виконаний етап розробки, закріплення і утвердження справжнього технічного завдання. На стадії робочого проектування повинні бути виконані наступні етапи робіт:

1. розробка програми;
2. розробка програмної документації;
3. перевірка програми.

На стадії впровадження виконуються такі етапи розробки, як підготовка і передача програми.

## **12. Вимоги до передачі прав власності на проект**

Після передачі програмного продукту замовнику, він має право тестувати програму на протязі 7 днів. Після тестування замовник повинен прийняти роботу по даному етапу або в письмовому вигляді розкрити причину відмови прийняття. У випадку обґрунтованої відмови розробник повинен допрацювати додаток.

**2.** **Моделювання програмного забезпечення**

**2.1 Діаграма варіантів використання**

Діаграма варіантів використання описує функціональне призначення системи або, іншими словами, те, що система буде робити в процесі свого функціонування. Вона є вихідним концептуальним поданням або концептуальною моделлю системи в процесі її проектування і розробки.

Суть даної діаграми складається в наступному: проектована система представляється у вигляді безлічі так званих варіантів використання, що надаються системою безлічі акторів або сутностей, що взаємодіють з системою. При цьому актором (actor) або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на моделируемую систему так, як визначить сам розробник.

У свою чергу, варіант використання (use case) служить для опису сервісів, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який чинять системою при діалозі з актором. Варіанти використання визначають функціональні можливості. Кожен з них представляє певний спосіб використання. Таким чином, кожен варіант використання відповідає послідовності дій для того, щоб клієнт міг отримати певний результат. На малюнку поданому нижче, зображена діаграма варіантів використання для ІС Автовокзалу.

Клієнт - все люди, які хочуть скористатися послугами автовокзалу; ІС Автовокзалу - надає необхідну інформацію та базу даних пасажирів; адміністратор - особа, яка займається обслуговуванням бази даних. Клієнти і адміністратори є зовнішніми сутностями. Клієнт звертається в автовокзал, для надання йому послуг, таких як замовлення, видача або отримання якоїсь інформації.

Вибравши потрібну послугу, клієнт проходить процедуру ідентифікації, і якщо потрібно реєструється в базі даних клієнтів. Основним варіантом використання служить "видача квитку". Для отримання квитку, клієнт вказує дату та пункт призначення, тому "видача квитку", включає (include) "перегляд бази марштурів". Після вибору маршруту, клієнту необхідно пройти процедуру ідентифікації, адміністратор перевіряє БД маршрутів на наявність його в базі, отже, видача включає "роботу з базою даних клієнтів". Замовляючи білет, клієнт також може ознайомитись з базою рейсів.

Для цього варіант використання "замовлення квитку" має розширення (extend). Таким чином властивості варіанти використання "замовлення квитку" доповнюються завдяки наявності властивостей у розширеного варіанту використання "видача квитку".

При отриманні квитку, клієнт проходить ідентифікацію у адміністратора, який додає клієнта в БД клієнтів, тим самим варіант використання "видача квитку" включає роботу з БД клієнтів. Після того, як клієнт обрав рейс, йому необхідно заплатити кошти, отже, варіант використання "оплата квитку" включає (include) варіант використання "оплата квитку".

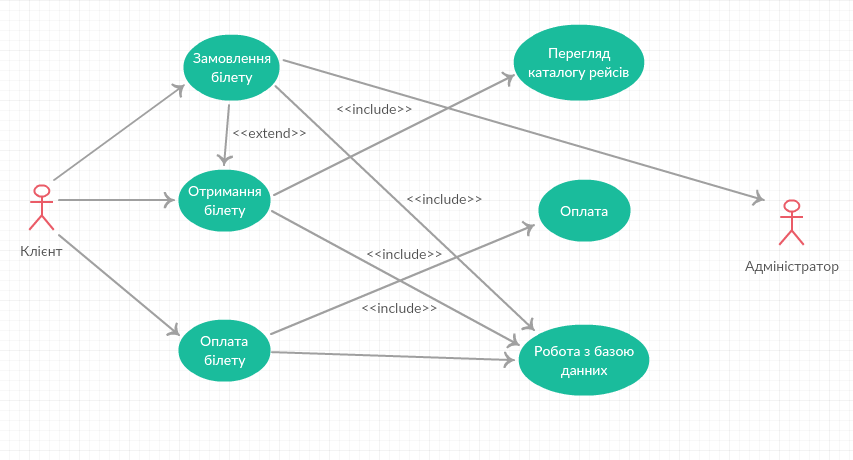


Рис.1 – Діаграма варіантів використання.

**2.2 Діаграма прецедентів**

На малюнку нижче наведена діаграма прецедентів. У цій діаграмі можна виділити наступні суб'єкти і відповідні їм прецеденти:

Продавець - працює з клієнтами: продає квитки (прецедент "Продаж квитків"), консультує клієнтів при здійсненні покупки (прецедент "Консультація клієнтів"), видає квиток (прецедент "Видача квитків") і приймає кошти за квиток (прецедент "Отримання оплати"), відповідає на питання клієнта (прецедент "Запит відомостей").

Клієнт – купує квиток (прецедент "Продаж квитків"), задає питання (прецедент "Запит відомостей") .

Останній суб’єкт не матиме безпосереднього доступу до розроблюваної системі (другорядний суб'єкт), проте саме він є основним джерелом подій, ініціюючих прецеденти, і одержувачем результату роботи прецедентів

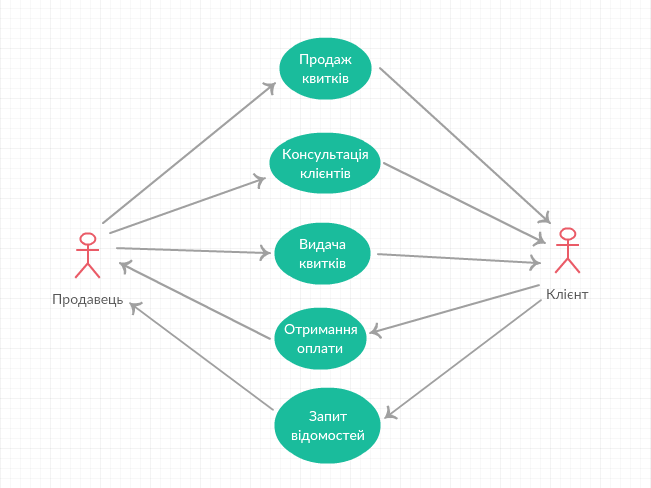


Рис.2 – Діаграма прецедентів.

**2.3 Діаграма класів**

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. Діаграма класів може відбивати, зокрема, різні взаємозв'язки між окремими сутностями предметної області, такими як об'єкти і підсистеми, а також описує їх внутрішню структуру і типи відносин. На даній діаграмі не вказується інформація про тимчасові аспектах функціонування системи. З цієї точки зору діаграма класів є подальшим розвитком концептуальної моделі проектованої системи. Коли говорять про даної діаграмі, мають на увазі статичну структурну модель проектованої системи. Тому діаграму класів прийнято вважати графічним представленому таких структурних взаємозв'язків логічної моделі системи, які не залежать від часу. Діаграма класів складається з безлічі елементів, які в сукупності відображають декларативні знання про предметну область. Ці знання інтерпретуються в базових поняттях мови UML, таких як класи, інтерфейси і відносини між ними і їх складовими компонентами.

**3. Розробка інтерфейсу користувача**

**3.1 Загальні рекомендації**

Загальні рекомендації по розробці графічного інтерфейсу були використані такі:

* Кольори, що використовувались у програмі є легкими для сприйняття, тому сприйняття програми не втомлює користувача при роботі з програмою.
* Шрифт та розмір надписів у програмі відповідно зручні для користувачів.

**3.2 Багато-віконні та багатосторінкові додатки.**

Опис всіх вікон програми можна розглянути в таблиці 2.2.

Таблиця 2.2

| **№ п/п** | **Назва** | **Призначення** | **Примітка**  **(якщо використовується компонента палітри Dialogs чи інше стандартне вікно)** |
| --- | --- | --- | --- |
| **1.** | Form1 | Головна форма програми з якої починається робота. |  |
| **2.** | Reys | На даній формі касир має можливість подивитися перелік всіх рейсів, а також редагувати його. |  |
| **3.** | DobReys | Переходячи на цю форму касир може додати рейс. |  |
| **4.** | Bilet | З цієї форми відбувається продаж білетів. |  | |
| **5.** | Pass | На даній формі показана база даних пасажирів. |  | |
| **6.** | Stat | На цій формі при натисканні на кнопку відображається деяка статистика перевезень. |  | |

**3.3 Кольорове оформлення додатків**

Вікна програми мають світлий колір, такий розподіл кольорів добре впливає на зір користувача.

**3.4 Шрифти**

У даній програмі не має присутніх шрифтів.

**4.Опис використаних класів у програмі**

**4.1 Класи.**

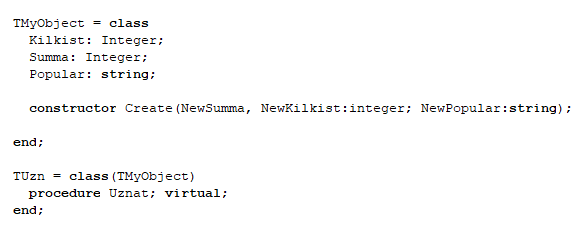


Рис.3. Батьківський клас TMyObject що включає в себе конструктор, 3 поля та один успадкований клас котрий має віртуальний метод

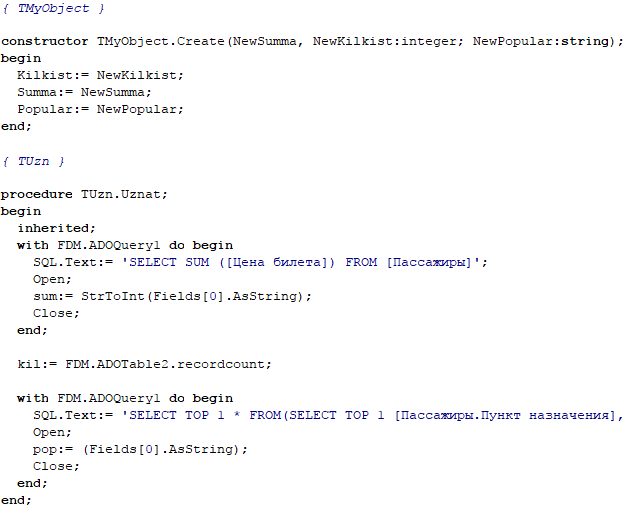


Рис.4. Реалізація методів

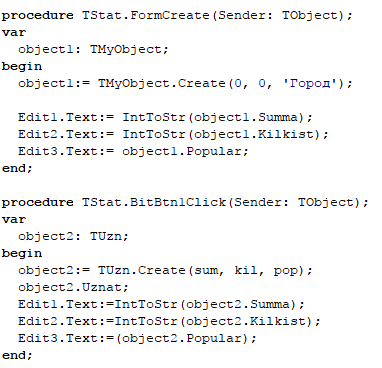


Рис.5. Реалізація методів

**5.Тестування**

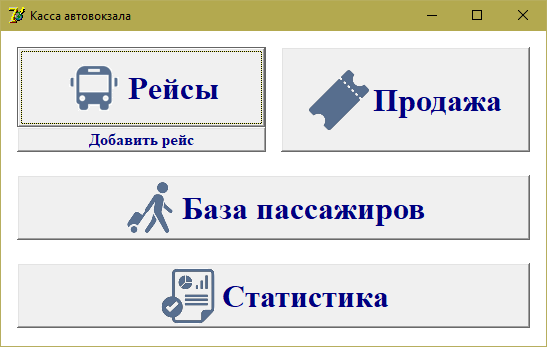


Рис.6. – Запуск програми.

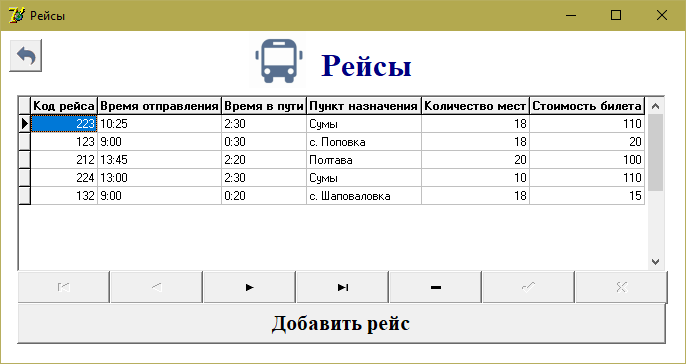


Рис.7. – Вкладка «Рейсы»

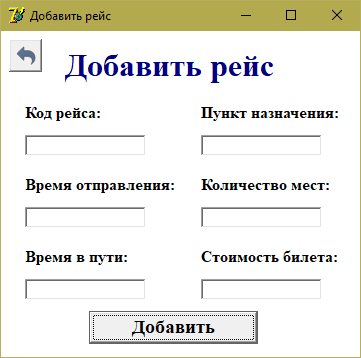


Рис.8. – Форма добавлення рейсу

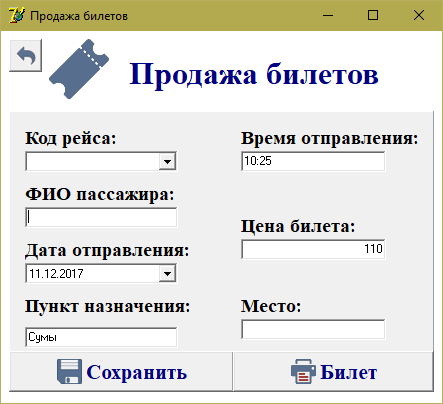


Рис.9. – Форма «Продажа билетов»

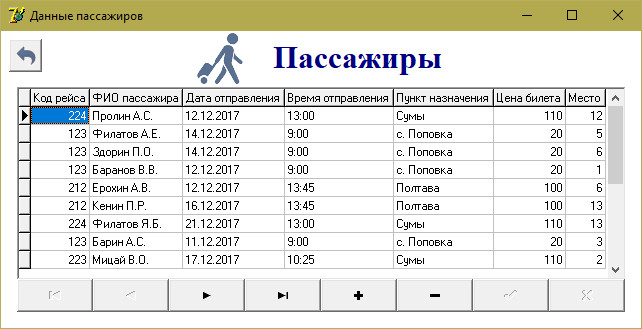


Рис.10. – Форма «База пассажиров»

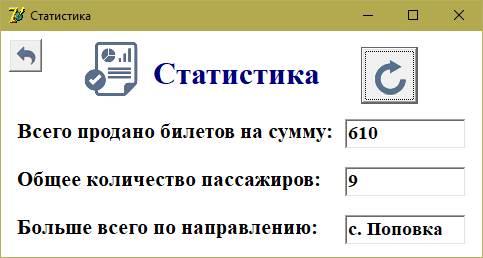


Рис.11. – Форма «Статистика»

**6.Публікація проекту**

**6.1 Розробка системи контролю версій Git**

Проект був опублікований на один з найбільших веб-сервісів для спільної розробки програмного забезпечення – **GitHub**. Так званий як «соціальна мережа для розробників». Є можливість прямого додавання нових файлів в свій репозиторій через веб-інтерфейс сервісу. На сайті є сервіс для швидкої публікації фрагментів коду, та прямо на сайті можна переглядати файли проектів. Завдяки цьому можливо контролювати оновлення та версії проекту. Для того щоб здійснити публікацію проекту потрібно виконати такі дії :

1. Перейти на сайт **GitHub** за посиланням https://github.com/ де потрібно створити аккаунт . Після реєстрації аккаунту потрібно створити новий репозиторій де і буде зберігатися проект, частка коду або його версія, оновлення тощо.

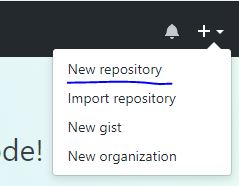


Рис.11 – створення репозиторію на **GitHub.**

1. При створенні репозиторію потрібно вказати його назву, опис (не обов’язково) та рівень доступу. Публічний або приватний. Натиснути кнопку «Create repository». Рис.12

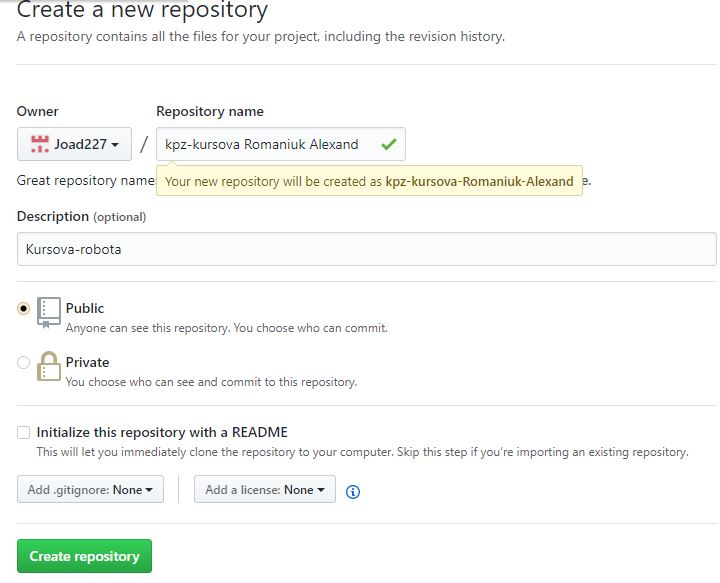


Рис.12 – створення репозиторію на **GitHub.**

Як тільки буде зроблено перші два кроки по створенню репозиторю він з’явиться у профілі. Рис.13

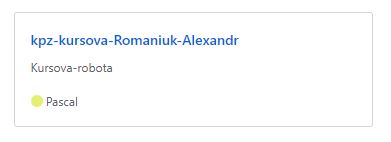


Рис.13 – Пустий репозиторій.

1. Для того щоб розпочати публікацію проекту потрібно встановити. Git Bash. Git Bash - це інтерпретатор bash для Windows, причому інтегрований з Git: він «розуміє» команди Git. Завантажити його можна за посиланням https://gitforwindows.org/ . Інтерфейс Git Bash виглядає як звичайна консоль Windows.
2. Відкриваємо Git Bash та за допомогою ключових команд задаємо ім’я та пошту які вказували при реєстрації. Рис.14

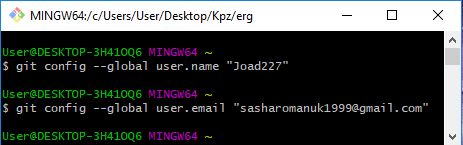


Рис.14 - Git Bash

1. Відкриваємо папку з проектом та пишемо команду у Git Bash «git init» після цієї команди буде створений репозиторій в папці з проектом. Також можна прописати шлях до проекту вручну так як показано на Рис.18. Далі пишемо «git status» для того щоб упевнитися що все зробіли вірно. Будуть показані файли проекту. Рис.15

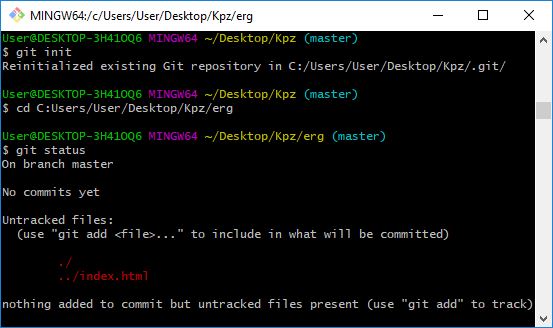


Рис.15 - Git Bash.

1. Далі пишемо команду «git add . » Ця команда означає що всі файли що були у папці будуть завантажені. І щоб упевнитися що всі файли були проіндексовані пишемо команду «git status». Рис.16

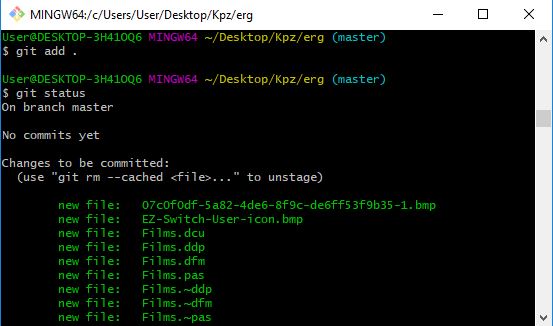


Рис.16 - Git Bash.

1. Наступним кроком потрібно завантажити проект на GitHub. Для цього йдемо на сайт GitHub і входимо в створений раніше рипозиторій, в нашому випадку «[kpz-kursova-Romaniuk-Alexandr](https://github.com/Joad227/kpz-kursova-Romaniuk-Alexandr)». Та копіюємо команду з посиланням Рис.17



Рис.17

1. Вставляємо скопійовану команду у Git Bash. І остання дія , пишемо команду « git push –u origin master ». Після цього проект буде опублікований на GitHub. Рис 18

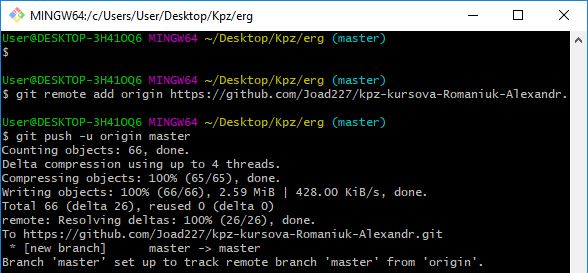


Рис.18 - Git Bash.

1. Перевіряємо наявність файлів проекту на сайті GitHub. Рис. 19.

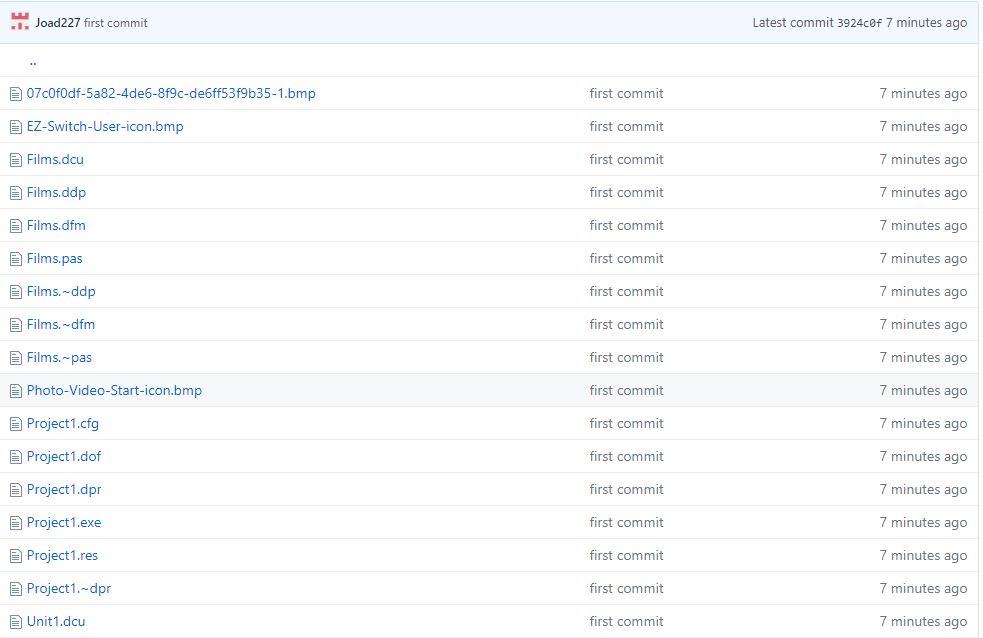


Рис. 19

**Висновок**

В результаті виконаної роботи була створена інформаційна система «Відеопрокат», яка є додатком, розробленим у середовищі MS Access 2016 та Delphi 7. Вона являє собою базу даних, в якій міститься інформація роботу відеопрокату. Додаток має дружній інтерфейс, щоб користувач зміг швидко розібратися, як вона працює, і оцінити функціональний зручність системи. Дані додаються в систему за допомогою спеціальних форм. Система протестована і готова до використання.

Інформаційна система управління «Відеопрокат» складається з трьох взаємопов'язаних модулів: читання даних, введення даних та редагування даних.

Система реалізовує роботу з таблицями, а саме введення, редагування і перегляд даних за допомогою форм. Під введенням мається на увазі введення нових записів у таблиці з використанням форм, редагування – редагування або видалення (для деяких таблиць) вже наявних даних, перегляд – перегляд наявних даних в табличній формі. Програма створена для полегшення та автоматизації роботи відеопрокату.

**Додаток А**

**Лістинг**

**Form1**

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ComCtrls, Menus, StdCtrls, Buttons,ShellApi;

type

TForm1 = class(TForm)

MainMenu1: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

N3: TMenuItem;

PageControl1: TPageControl;

TabSheet1: TTabSheet;

TabSheet2: TTabSheet;

BitBtn1: TBitBtn;

BitBtn2: TBitBtn;

BitBtn3: TBitBtn;

BitBtn4: TBitBtn;

BitBtn5: TBitBtn;

BitBtn6: TBitBtn;

BitBtn7: TBitBtn;

BitBtn8: TBitBtn;

BitBtn9: TBitBtn;

Label1: TLabel;

procedure BitBtn1Click(Sender: TObject);

procedure BitBtn3Click(Sender: TObject);

procedure BitBtn5Click(Sender: TObject);

procedure BitBtn6Click(Sender: TObject);

procedure BitBtn7Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

uses Films, rabotniki, klin, histori, vidach;

{$R \*.dfm}

procedure TForm1.BitBtn1Click(Sender: TObject);

begin

Form2.ShowModal;

end;

procedure TForm1.BitBtn3Click(Sender: TObject);

begin

Form3.ShowModal;

end;

procedure TForm1.BitBtn5Click(Sender: TObject);

begin

klient.ShowModal;

end;

procedure TForm1.BitBtn6Click(Sender: TObject);

begin

vidacha.ShowModal;

//ShellExecute(0, nil, 'kar.xls', nil, nil, SW\_SHOWNORMAL);

end;

procedure TForm1.BitBtn7Click(Sender: TObject);

begin

history.ShowModal;

end;

end.

**Form2**

unit Films;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, Grids, DBGrids, Buttons, DBCtrls, ExtCtrls, StdCtrls, ComCtrls,

ExtDlgs;

type

TForm2 = class(TForm)

GroupBox1: TGroupBox;

DBImage1: TDBImage;

Button1: TButton;

DBRichEdit1: TDBRichEdit;

DBComboBox1: TDBComboBox;

Button2: TButton;

Panel1: TPanel;

DBNavigator1: TDBNavigator;

DBGrid1: TDBGrid;

OpenPictureDialog1: TOpenPictureDialog;

Edit1: TEdit;

Label1: TLabel;

procedure Button1Click(Sender: TObject);

procedure Edit1Change(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form2: TForm2;

implementation

uses dm, Unit1;

{$R \*.dfm}

procedure TForm2.Button1Click(Sender: TObject);

begin

fdm.Tasortiment.Edit;

if OpenPictureDialog1.Execute then DBImage1.Picture.LoadFromFile(OpenPictureDialog1.FileName);

fdm.Tasortiment.Post;

end;

procedure TForm2.Edit1Change(Sender: TObject);

begin

fdm.MyLocatef(Edit1.Text);

end;

end.

**Form3**

unit rabotniki;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, Buttons, DBCtrls, Mask, ExtCtrls, Menus, Grids,

DBGrids;

type

TForm3 = class(TForm)

GroupBox1: TGroupBox;

DBEdit1: TDBEdit;

DBEdit2: TDBEdit;

DBEdit3: TDBEdit;

DBEdit4: TDBEdit;

DBEdit5: TDBEdit;

DBEdit6: TDBEdit;

DBComboBox1: TDBComboBox;

BitBtn1: TBitBtn;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

Label7: TLabel;

MainMenu1: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

N3: TMenuItem;

N4: TMenuItem;

Button1: TButton;

Button2: TButton;

DBGrid1: TDBGrid;

DBNavigator1: TDBNavigator;

Edit1: TEdit;

Label8: TLabel;

Label10: TLabel;

Edit2: TEdit;

procedure N2Click(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

procedure GroupBox1Exit(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Edit1Change(Sender: TObject);

procedure Edit2Change(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form3: TForm3;

implementation

uses dm, Films, Unit1, dol;

{$R \*.dfm}

procedure TForm3.N2Click(Sender: TObject);

begin

Form3.Close;

end;

procedure TForm3.BitBtn1Click(Sender: TObject);

begin

dolznosti.ShowModal;

end;

procedure TForm3.GroupBox1Exit(Sender: TObject);

begin

if fdm.Tsotrudnik.Modified then

fdm.Tsotrudnik.Post;

end;

procedure TForm3.Button1Click(Sender: TObject);

begin

if fdm.Tsotrudnik.Modified then

fdm.Tsotrudnik.Post;

Close;

end;

procedure TForm3.Button2Click(Sender: TObject);

begin

fdm.Tsotrudnik.Append;

DBEdit1.SetFocus;

end;

procedure TForm3.Edit1Change(Sender: TObject);

begin

fdm.MyLocatet(Edit1.Text);

end;

procedure TForm3.Edit2Change(Sender: TObject);

begin

fdm.MyLocatee(Edit2.Text);

end;

end.

**Form vidacha**

unit vidach;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs,ShellApi, StdCtrls;

type

Tvidacha = class(TForm)

Button1: TButton;

Button2: TButton;

Button3: TButton;

Button4: TButton;

procedure FormCreate(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

TProduct = class

Name:string;

Manufacturer:string;

procedure Buy; virtual; abstract;

constructor Create(NewName,NewManufacturer:string);

end;

Tvideo = class(TProduct)

procedure Buy; override;

end;

Twater = class(TProduct)

procedure Buy; override;

end;

Tchocolate = class(TProduct)

procedure Buy; override;

end;

var

vidacha: Tvidacha;

a: array [1..3] of TProduct;

implementation

{$R \*.dfm}

procedure Tvidacha.FormCreate(Sender: TObject);

begin

a[1]:= Tvideo.Create('Ôiëüì','12');

a[2]:= Twater.Create('Íàïié','22');

a[3]:= Tchocolate.Create('Øîêîëàä','32');

end;

{ TProduct }

constructor TProduct.Create(NewName, NewManufacturer: string);

begin

Name:= NewName;

Manufacturer := NewManufacturer;

end;

{ Tvideo }

procedure Tvideo.Buy;

var Count,vName:string;

Price:integer;

begin

vName:= InputBox('Âêàæiòü íàéìåíóâàííÿ ôiëüìó ','íàéìåíóâàííÿ:','Ôiëüì..');

Count:= InputBox('Âêàæiòü êiëüêiñòü ãîäèí îðåíäè ','Êiëüêiñòü ãîäèí:','24');

Price:= StrToInt(Count) \* 2;

ShowMessage('Íàéìåíóâàííÿ: Ôiëüì' +vName + #13 + 'Öiíà â ãðí: '+ IntToStr(Price));

end;

{ Twater }

procedure Twater.Buy;

var Count:string;

Price:integer;

begin

Count:= InputBox('Âêàæiòü êiëüêiñòü ïëÿøîê','êiëüêiñòü:','2');

Price:= StrToInt(Count) \* 5 ;

ShowMessage('Íàéìåíóâàííÿ: Íàïié'+ #13 + 'Ö³íà â ãðí: '+ IntToStr(Price));

end;

{ Tchocolate }

procedure Tchocolate.Buy;

var Count:string;

Price:integer;

begin

Count:= InputBox('Âêàæiòü êiëüêiñòü øîêîëàäîê','êiëüêiñòü:','2');

Price:= StrToInt(Count) \* 10 ;

ShowMessage('Íàéìåíóâàííÿ: Øîêîëàä'+ #13 + 'Ö³íà â ãðí: '+ IntToStr(Price));

end;

procedure Tvidacha.Button1Click(Sender: TObject);

begin

a[1].Buy;

end;

procedure Tvidacha.Button2Click(Sender: TObject);

begin

a[2].Buy;

end;

procedure Tvidacha.Button3Click(Sender: TObject);

begin

a[3].Buy;

end;

procedure Tvidacha.Button4Click(Sender: TObject);

begin

ShellExecute(0, nil, 'kar.xls', nil, nil, SW\_SHOWNORMAL);

end;

end.