Dica Ionela Valentina
Grupa 3123a

Caiet

Practica:

Ziua 1 (6 ore)-23.07:
1. Protectia muncii.
2. Prezentarea temei pentru practica, stabilirea sarcinilor de lucru, prezentarea echipamentului, detalierea si distribuirea sarcinilor pe etape.
2. Crearea aplicatiei in C# propriu-zisa in consola si impartirea proiectul pe mai multe nivele: nivelul stocarii datelor si librarii.
3. Am inceput cu clasa de baza “Utilizator” pentru care am implementat constructorii si metodele necesare operarii cu datele disponibile.
4. Pentru nivelul stocarii datelor, m-am ocupat de administrarea datelor dintr-un fisier csv, unde am implementat constructorul unui admin care sa opereze cu un fisier, apoi am implementat metode de adaugare, cautare, stergere din si in fisier si o metoda care extrage toate datele din fisier sub forma unui tablou de obiecte.
5. In main am facut testele necesare verificarii rularii corespunzatoare.
6. Am creat o interfata destinata utilizatorului si am adaugat cateva butoane, label-uri, textbox-uri care mai apoi sa indeplineasca functiile necesare.
7. Am adaugat pentru rularea in consola o secventa de cod care ascunde consola si afiseaza un message box, oferindu-i utilizatorului aplicatiei optiunea de a alege in ce mod isi doreste sa foloseasca aplicatia: in consola sau in interfata grafica.

Ziua 2 (6 ore)-24.07:
1. Am setat proprietatile si evenimentele pentru toate controalele folosite, incercand sa dau un aspect estetic interfetei grafice.
2. Am testat repetat comportamentul aplicatiei in urma validarilor implementate si am verificat functionarea corecta a aplicatiei.
3. Am adaugat un alt formular pentru citirea si salvarea unui nou utilizator in fisier, care se inchide imediat dupa apasarea butonului Ok a mesajului de confirmare a salvarii.
4. Am implementat alte functii, ca: reimprospatarea DataGrid ului la fiecare modificare, Cautare, Modificarea unui anumit utilizator selectat din DataGrid.
5. Am reluat codul punand in comentarii explicatii si eliminand posibilele greseli sau copieri.

Ziua 3 (6 ore)-25.07:
1. M-am folosit de libraria System.IO.Ports pentru a implementa functia necesara pentru deschiderea portului serial. Am realizat conexiunea dintre program si modem. Am citit documentatia modemului pentru a gasi comenzi utile pentru aplicatia mea.
2. M-am folosit de aplicatia PuTTY pentru a vedea cum primesc datele cand numarul cartelei din modem este apelat. Pe baza acestor teste am facut prelucrari in cod asupra datelor primite astfel incat sa pot extrage numarul de telefon.
3. Cu ajutorul functiilor si claselor implementate in zilele anterioare, am implementat o metoda de cautare a utilizatorilor, care sa afiseze utilizatorul care a sunat sau, in cazul in care utilizatorul nu se afla in fisierul .csv, sa se afiseze un mesaj corespunzator.

4. Datele primite in ceea ce priveste numarul de telefon difera de la un apelant la altul(“+40, 07, 0040”), deci am ales ca format universal “0040” si am facut o metoda de validare si formatare a numarului.
5. Am adaugat comentarii ajutatoare pentru a-mi aminti ce am facut in momentul unei revizuiri.

Ziua 4 (6 ore)-26.07:
1. Am reusit sa adaug partea de ascultare a modemului si in interfata, impreuna cu verificarea daca utilizatorul exista in fisier.
2. M-am ocupat de alte aspecte in ceea ce priveste interfata grafica (estetica, erori).

Ziua 5 (6 ore)-29.07:
1. Documentatie asupra standardului “Wake-on-Lan” si ce anume este necesar pentru a putea implementa in programul meu.
2. Implementare functie SendWakeOnLan si testare in consola, impreuna cu configurarea unui alt calculator pentru a avea WoL(Wake-on-Lan) enabled.
3. Am incercat sa creez metodele necesare pentru ascultare mesaje, insa fara succes, deoarece intrau in conflict cu apelurile si modemul nu raspundea mereu la comenzi.
4. Am incercat sa imbunatatesc preluarea datelor primite atunci cand se verifica daca sunt mesaje necitite.

Ziua 6(6 ore)-30.07:
1. Incercare de debugging, gasire a problemelor existente si rezolvarea lor.
2. Modificari majore asupra codului care intr-un final tot nu functionau.
3. Am realizat ca raspunsul la comenzile date depinde si de operatorul de retea. Am incercat sa remediez acest lucru prin cod (trimiterea repetata a aceleiasi comenzi).

Ziua 7(6 ore)-31.07:
1. Am reluat documentatia modemului si am incercat sa gasesc o alta modalitate de a gestiona mesajele.
2. In loc de a verifica ciclic modemul daca au venit mesaje, am ales sa folosesc o comanda care trimite o „notificare” de fiecare data cand e trimis un mesaj nou (doar in timpul rularii programului, deoarece mesajele necitite ar putea realiza o comanda trimisa cu cateva zile in urma daca ar fi considerate mesajele necitite).
3. Am implementat o metoda care sa preia datele notificarii si dupa sa caute mesajul in functie de index-ul furnizat de notificare si sa il afiseze, impreuna cu numarul de telefon.
4. Am implementat metode de extragere a numarului de telefon si al mesajului.
5. Am adaugat si o alta clasa TestBench pentru a verifica daca continutul mesajului (de tip TB1, TB2, etc) este in fisier si, dupa verificarea existentei numarului de telefon printre utilizatori, pornirea calculatorului cu adresa MAC corespunzatoare TB-ului.
6. Verificarea functionalitatii programului in diferite conditii si modificari, daca a fost cazul.
7. Restructurarea codului pentru a fi mai fluent si pentru o logica mai buna.

Ziua 8(6 ore)-01.08:
1. Documentatie asupra fisierelor xml, app.config, app.settings si modul in care sunt prelucrate datele din acesta in C#.
2. Modificarea tipului de fisier din care sunt preluate datele (fisierul csv este inlocuit de fisierul xml), incercarea introducerii datelor de intrare in fisierul app.config.
3. Incercare de extragere si prelucrare a datelor de intrare din fisierul app.config

Ziua 9(6 ore)-02.08:
1. Finalizarea partii de prelucrare a datelor dintr-un fisier xml.
2. Inceputul transformarii aplicatiei consola intr-un serviciu windows, care va rula pe fundal. Eliminarea interfetei grafice si a message-box-ului afisat la rularea in consola.
3. Documentatie asupra crearii unui serviciu windows
4. Crearea unui proiect nou de tip worker service. Acesta va rula in fundal sarcinile de care am nevoie.
Ziua 10(6 ore)-05.08:
1. Incercarea crearii unui proiect de tip Installer in cadrul solutiei care va instala serviciul windows.
2. Documentatie asupra proiectelor de tip Installer si alegerea unui template util pe care sa il pot modifica corespunzator.
3. Incercarea integrarii installerului in solutia mea, insa fara succes la instalarea propriu-zisa a serviciului.

Ziua 11(6 ore)-06.08:
1. Incercarea gasirii problemei in cee ace priveste instalarea esuata a serviciului.
2. Constatarea tipului de proiect gresit (.NET Framework). Acesta fiind o versiune mai veche, nu putea oferi aceleasi functionalitati, deci am creat un alt proiect .NET 6.0 care sa se potriveasca cu nevoile aplicatiei.
3. Restructurarea fisierelor proiectului astfel incat sa fie mai bine organizate.

Ziua 12(6 ore)-07.08:
1. Installer-ul instaleaza serviciul cu success, insa in momentul utilizarii serviciului, acesta se opreste singur la cateva secunde de la pornirea sa.
2. Incercarea rezolvarii problemei date prin diferite metode, insa fara success.

Ziua 13 (6 ore)-08.08:
1. Retestari pentru a identifica problema.
2. Identificarea problemei: modul in care aplicatia era tinuta deschisa. La inceput, proiectul fiind de tip consola, preveneam inchiderea programului prin Console.Readline() (era asteptata introducerea unui caracter de la tastatura), ceea ce pentru un serviciu windows nu functioneaza, nu exista o consola, serviciul ruleaza pe fundal, deci aplicatia pur si simplu se inchidea.
3.Rezolvarea problemei si comentarea tuturor functiilor specific consolei (inutile pentru un serviciu windows).

Ziua 14 (6 ore)-09.08:
1. Includerea in fisierul xml a datelor portului serial (vor fi citite din fisier).
2. Includerea crearii unui fisier de tip text cu activitatea cartelei (in momentul primirii unui apel, mesaj, se va scrie in fisier numarul si informatiile corespunzatoare, daca este autorizat sau nu si daca s-a deschis calculatorul dorit cu succes).
3. Executabilul crea fisierul, insa serviciul nu, din cauza ca nu avea permisiunea de a crea fisiere si foldere.

Ziua 15(6 ore)-12.08:
1. Rezolvarea problemei cu fisierul text.
2. Documentatie asupra plantUML.
3. Crearea unei scheme pentru logica proiectului cu ajutorul PlantUML.
4. Crearea unui mini-ghid de creare a executabilului si a installer-ului.