GRIGLIA VALUTAZIONE-AUTOVALUTAZIONE STATO AVANZA

Short-Streets

TITOLO PROGETTO:

15 (15%)

di cui progetto: 10 (10%)

ORE TOTALI PIANO OPERATIVO:

100

FUNZIONALITA' PRINCIPALI	dettaglio funzionalità	costo in ore (*)
F1: inserimento dati	F1.D1: indicare il numero di pacchi da consegnare e alcune loro caratteristiche come il peso e il volume.	20
F2: calcolo percorso minimo	F2.D1: calcolare il percorso minimo che il rider deve effettuare in modo da velocizzare e facilitare la consegna.	22
	F2.D2: implementazione di un algoritmo peso-volume che rappresenta un vincolo in più nel calcolo del percorso minimo.	18
F3: parte grafica	F3.D1: utilizzare una mappa di un piccolo quartiere sulla quale indicare i punti in cui il corriere deve andare.	10
	F3.D2: dopo aver individuato il percorso più' agevole, evidenziare il percorso che il corriere deve seguire sulla mappa stessa.	15

(*) con riferimento ai tempi previsti per l'implementazione

STUDENTE	SINTESI COMPITI PERSONALI	LIV. COMPITO (1-2-3) (*)
Bissola Mattia	Testing del software	3
Giuggioli Daniel	Interfaccia grafica, classi per la gestione dei pacchi, utilizzo di una mappa su cui visualizzare i punti presso cui si devono effettuare le consegne e il percorso più breve calcolato dagli algoritmi	3
Majid Zaccaria	Testing del software	3
Salvi Alessandro	Sviluppo dell'algoritmo che consente al rider di trasportare contemporaneamente determinati pacchi secondo determinate caratteristiche, quali il peso e il volume e unione dell'algoritmo peso-volume con quello del percorso minimo	3
Sonzogni Nicolò	Sviluppo dell'algoritmo per il calcolo del percorso minimo che il rider dovrà compiere e unione dell'algoritmo peso-volume con quello del percorso minimo. Utilizzo di una mappa su cui visualizzare i punti presso cui si devono effettuare le consegne e il percorso più breve calcolato dagli algoritmi	3

(*) 3: complesso .. 1:semplice

SITUAZIONE AL GIORNO : 20-02-21 **PROGETTO COMPLETATO AL 40%** (**TEST coverage** **8%**)

ORE SVOLTE 35 (35%) di cui progetto: 10 (67%) implementazione: 28 (37%) documentazione: 3 (30%)

(*) percentuali riferite alle rispettive ore del PIANO OPERATIVO

SITUAZIONE PROGETTO

FUNZIONALITA' PRINCIPALI	dettaglio funzionalità	% implementata	bilancio ore previste	svolte	restanti	+/- %
F1: inserimento dati	F1.D1: indicare il numero di pacchi da consegnare e alcune loro caratteristiche come il peso, il volume e la destinazione a cui il pacco deve giungere.	70	20	17	3	-15 %
F2: calcolo percorso minimo	F2.D1: calcolare il percorso minimo che il rider deve effettuare in modo da velocizzare e facilitare la consegna.	100	22	18	4	-18 %
	F2.D2: implementazione di un algoritmo peso-volume che rappresenta un vincolo in più nel calcolo del percorso minimo.	100	18	15	3	-17 %
F3: parte grafica	F3.D1: utilizzare una mappa di un piccolo quartiere sulla quale indicare i punti in cui il corriere deve andare.	0	10	0	10	-100 %
	F3.D2: dopo aver individuato il percorso più' agevole, evidenziare il percorso che il corriere deve seguire sulla mappa stessa.	0	15	0	15	-100 %

(*) con riferimento ai tempi previsti per l'implementazione

STUDENTE	SINTESI ATTIVITA' SVOLTA	ore svolte	RISULTATI Liv. comp.	% compl.	**VALUTAZIONE**
Bissola Mattia	Testing del software.	12	3	20%	
Giuggioli Daniel	Interfaccia grafica, classi per la gestione dei pacchi.	15	3	50%	
Majid Zaccaria	Testing del software.	12	3	20%	

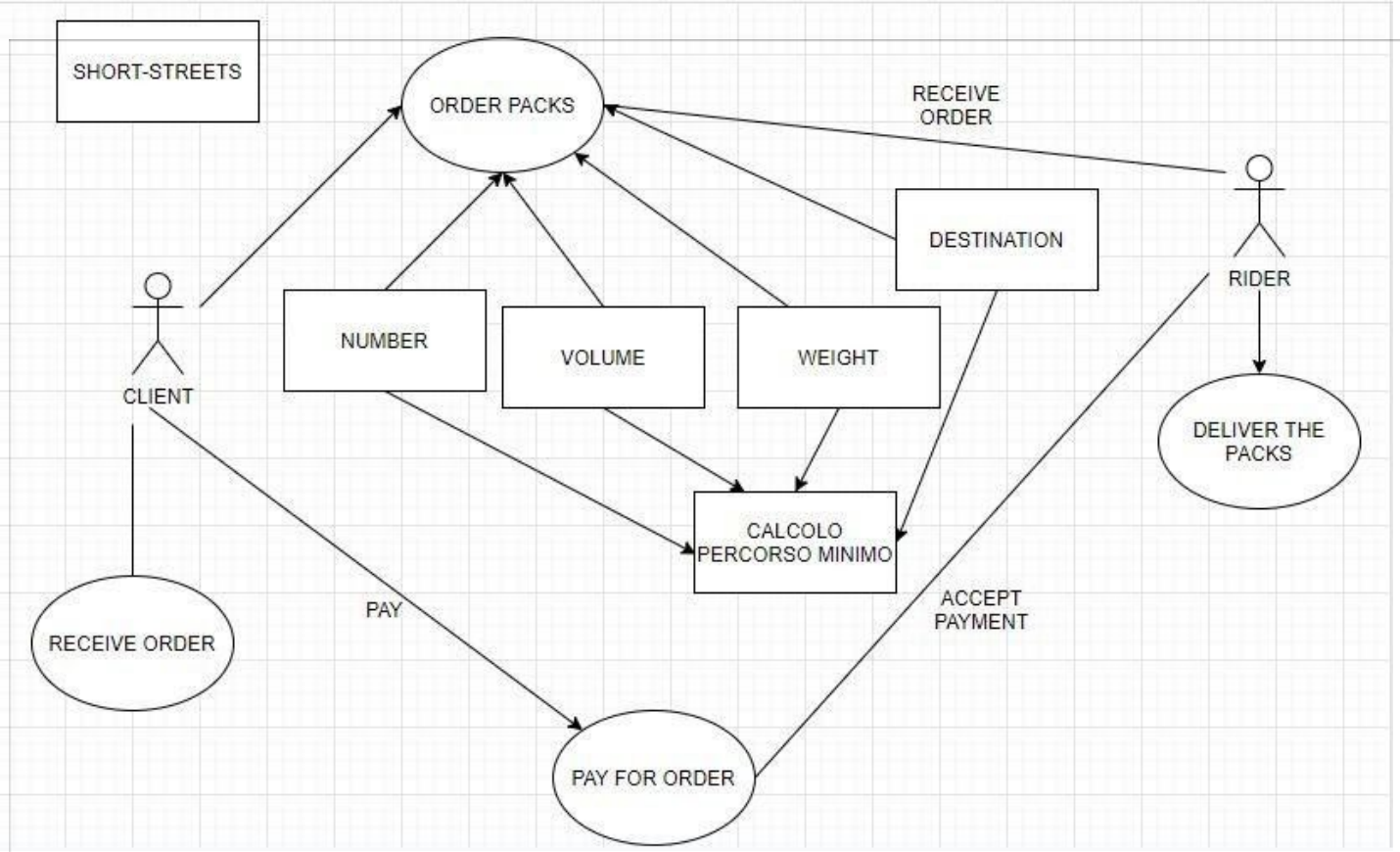
Salvi Alessandro	Sviluppo dell'algoritmo che consente al rider di trasportare contemporaneamente determinati pacchi secondo determinate caratteristiche, quali il peso e il volume.	15	3	75%
Sonzogni Nicolò	Sviluppo dell'algoritmo per il calcolo del percorso minimo che il rider dovrà compiere.	18	3	75%

Legenda risultati autovalutazione STATO AVANZAMENTO PROGETTO

VALUTAZIONE	RISULTATO PROGETTO	LIVELLO ATTIVITA'	% COMPITI PERSONALI COMPLETATI
POSITIVO	>= 75%	3 (alto)	>= 50%
POSITIVO	>= 75%	2 (medio)	>= 75%
POSITIVO	>= 75%	1 (base)	> 90%
BASE	>=75%	1,2,3	MINORE DEL PREVISTO PER IL LIVELLO
BASE	MINORE DEL 75%	1,2,3	MAGGIORI O UGUALI AL PREVISTO
NEGATIVO	MINORE DEL 75%	1,2,3	MINORE DEL PREVISTO PER IL LIVELLO

U se Case Diagram

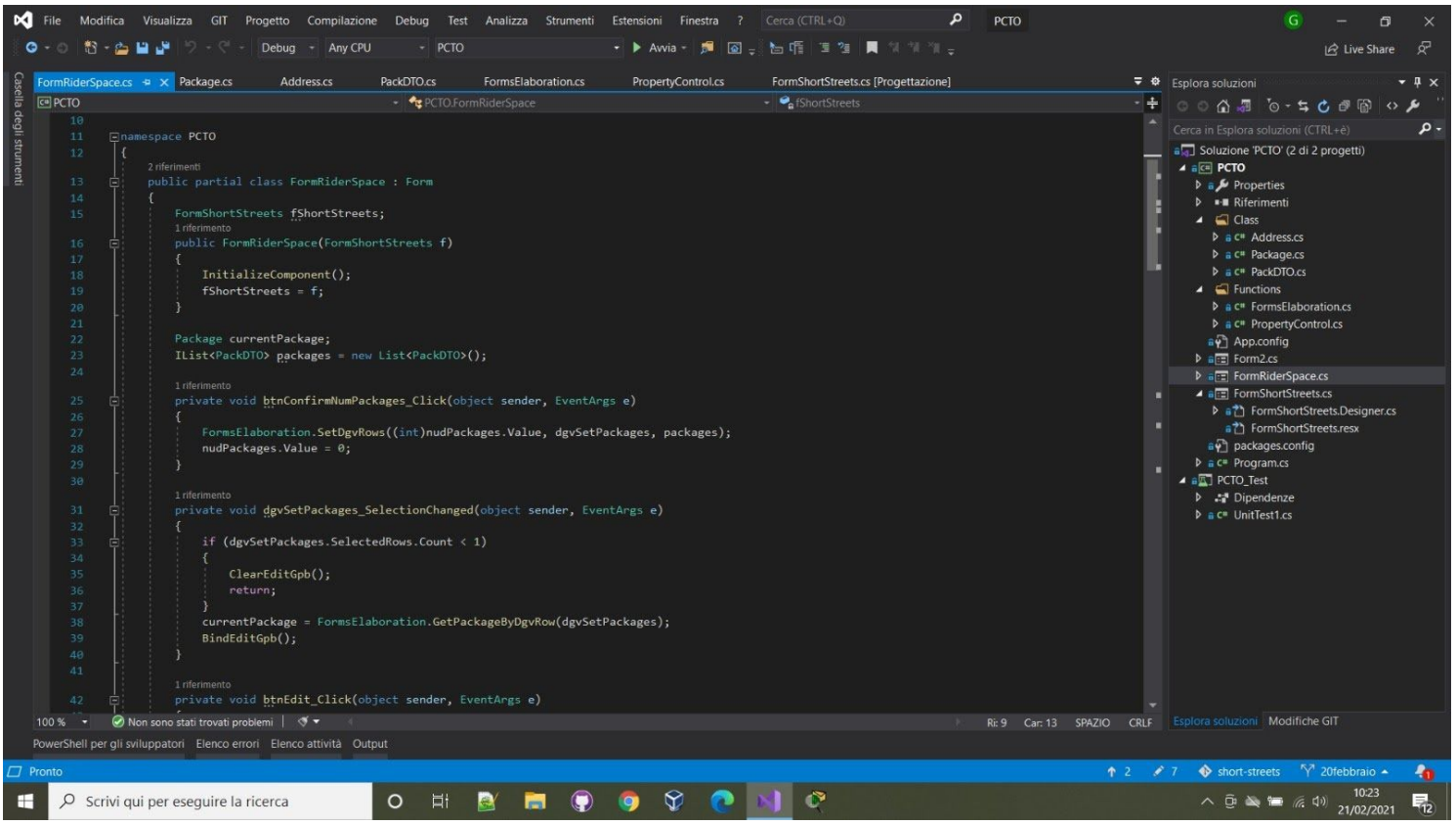
TITOLO PROGETTO	Short-Streets



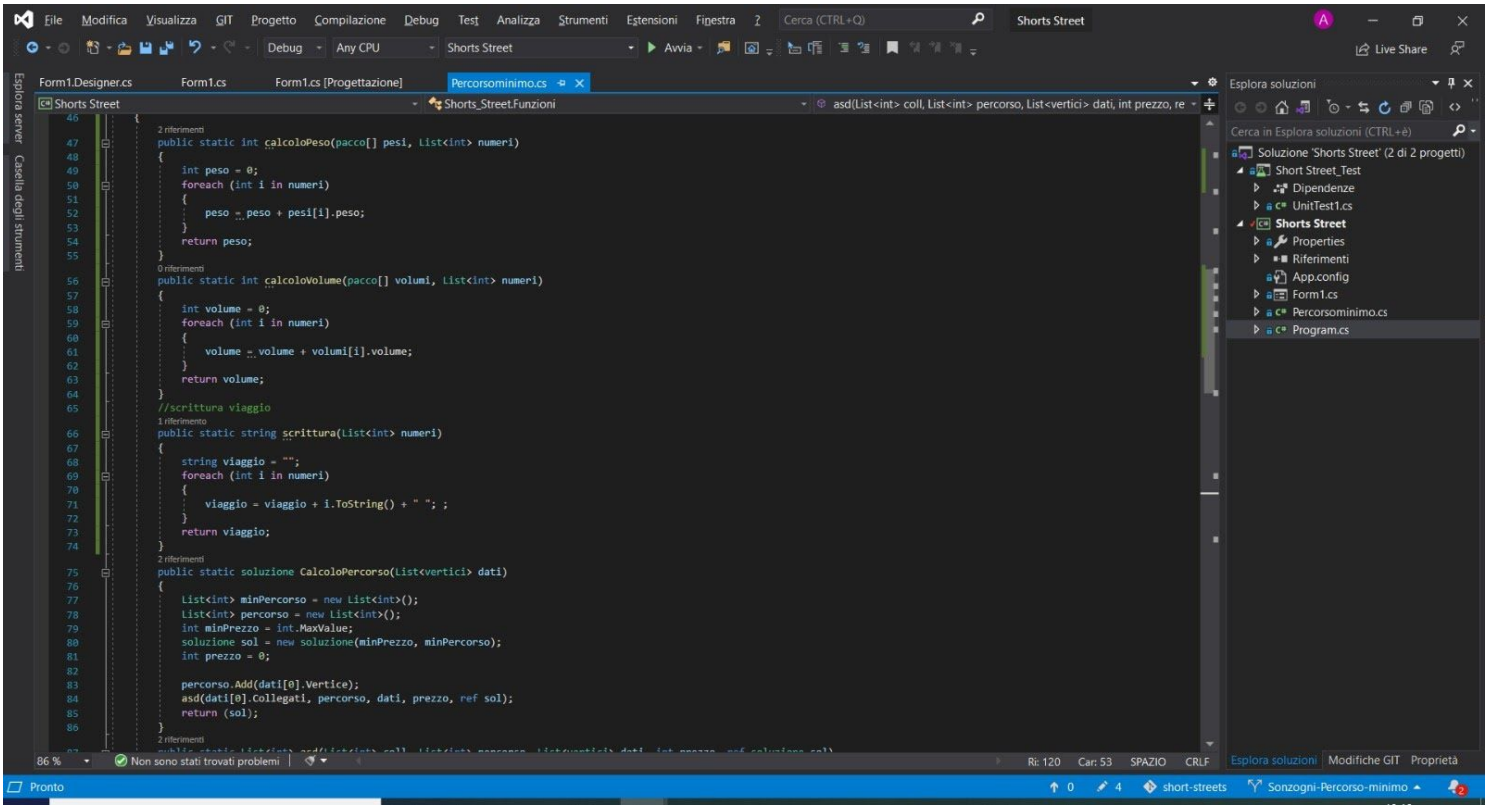
SCREENSHOTS

TITOLO PROGETTO	Short-Streets

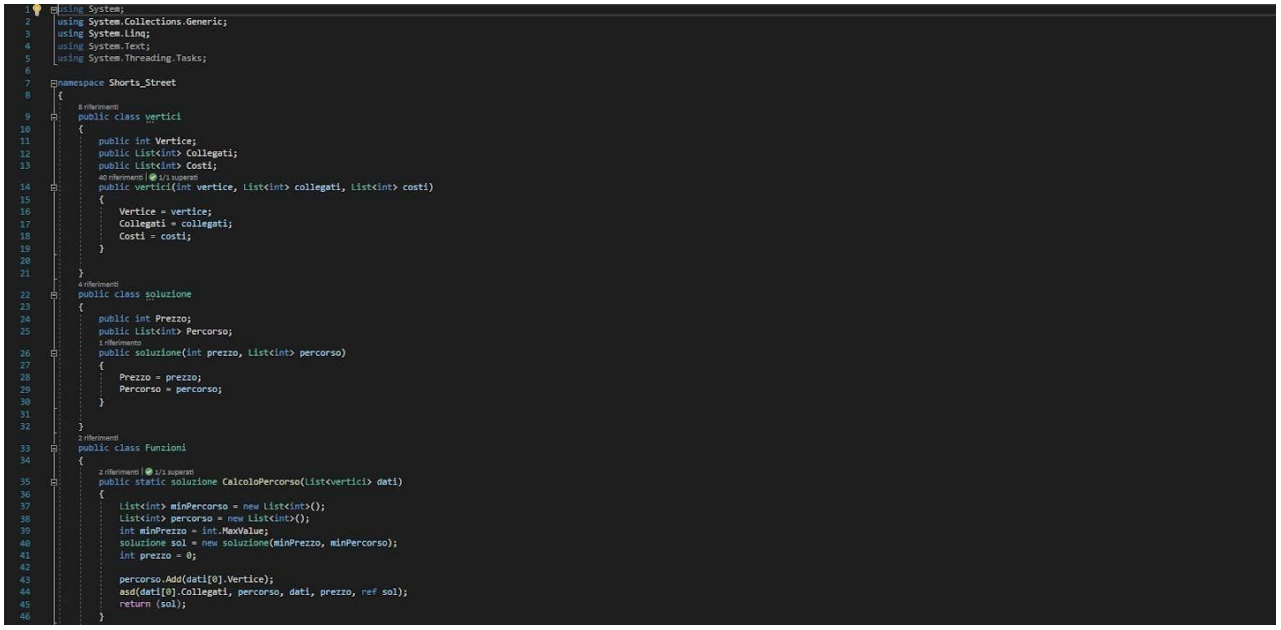
F1.D1:



F1.D2:



F2.D1:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Shorts_Street
{
    8 riferimenti
    public class vertici
    {
        public int Vertice;
        public List<int> Collegati;
        public List<int> Costi;
        40 riferimenti | 1/1 superati
        public vertici(int vertice, List<int> collegati, List<int> costi)
        {
            Vertice = vertice;
            Collegati = collegati;
            Costi = costi;
        }

    }
    4 riferimenti
    public class soluzione
    {
        public int Prezzo;
        public List<int> Percorso;
        1 riferimento
        public soluzione(int prezzo, List<int> percorso)
        {
            Prezzo = prezzo;
            Percorso = percorso;
        }

    }
    2 riferimenti
    public class Funzioni
    {
        2 riferimenti | 1/1 superati
        public static soluzione CalcoloPercorso(List<vertici> dati)
        {
            List<int> minPercorso = new List<int>();
            List<int> percorso = new List<int>();
            int minPrezzo = int.MaxValue;
            soluzione sol = new soluzione(minPrezzo, minPercorso);
            int prezzo = 0;

            percorso.Add(dati[0].Vertice);
            asd(dati[0].Collegati, percorso, dati, prezzo, ref sol);
            return (sol);
        }
```

```
public static List<int> asd(List<int> coll, List<int> percorso, List<vertici> dati, int prezzo, ref soluzione sol)
        {
            foreach (int i in coll)
            {
                if (!percorso.Contains(dati[i - 1].Vertice))
                {
                    percorso.Add(i);
                    var temp = dati[i - 1].Collegati;
                    percorso = asd(temp, percorso, dati, prezzo, ref sol);
                }
            }
            if (percorso.Count == dati.Count)
            {
                prezzo = 0;
                if (dati[0].Collegati.Contains(percorso[percorso.Count - 1]))
                {
                    int posizione = 0;
                    int x = 0;
                    foreach (int i in percorso)
                    {
                        vertici elemento = dati.Where(y => y.Vertice == i).FirstOrDefault();
                        if (i == percorso[percorso.Count - 1])
                        {
                            foreach (int a in dati[0].Collegati)
                            {
                                if (a == percorso[percorso.Count - 1])
                                {
                                    prezzo = prezzo + elemento.Costi[0];
                                    break;
                                }
                            }
                            if (sol.Prezzo > prezzo)
                            {
                                sol.Prezzo = prezzo;
                                sol.Percorso.Clear();
                                foreach (int g in percorso)
                                {
                                    sol.Percorso.Add(g);
                                }
                            }
                            break;
                        }
                        else
                        {
                            foreach (int a in elemento.Collegati)
                            {
                                if (a == percorso[x + 1])
                                {
                                    prezzo = prezzo + elemento.Costi[posizione];
                                    posizione = 0;
                                    break;
                                }
                                posizione++;
                            }
                        }
                        x++;
                    }
                }
                percorso.RemoveRange(percorso.Count - 1, 1);
            }
            else
                percorso.RemoveRange(percorso.Count - 1, 1);

            return (percorso);
        }
```