GRIGLIA VALUTAZIONE-AUTOVALUTAZIONE STATO AVANZAMENTO PROGETTI PCTO 4IC – a.s 2020/2021

TITOLO PROGETTO:	SHORT-STREETS

PIANO OPERATIVO:

ORE TOTALI	110	di cui progetto:	15 (14 %)	implementazione:	85 (77 %)	documentazione:	10 (9%)

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.	22
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
	F2.D3: Unione algoritmi peso-volume e percorso minimo	20
F3: Mappe	F3.D1: utilizzare una mappa di un piccolo quartiere sulla quale indicare i punti in cui il corriere deve andare.	12
	F2.D1: dopo aver individuato il percorso più' agevole, evidenziare il percorso che il corriere deve seguire sulla mappastessa	16

(*) 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 cuisi 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 : _____21/04/2021_____ **PROGETTO COMPLETATO AL** _70_ % (**TEST coverage**____60__%)

ORE SVOLTE	85 (77 %)	di cui progetto:	13 (87%)	implementazione:	64 (75 %)	documentazione:	8 (80%)

(*) percentuali riferite alle rispettive ore del PIANO OPERATIVO

SITUAZIONE PROGETTO

FUNZIONALITA' PRINCIPALI	dettaglio funzionalità	% implementata	bilancio ore			
			previste	svolte	restanti	+/- %
F1	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.	95	20	25	0	+25
F2	F2.D1: calcolare il percorso minimo che il rider deve effettuare in modo davelocizzare e facilitare la consegna.	100	22	18	4	-18
	F2.D2: Implementazione di un algoritmo peso-volume che rappresentaun vincolo in più nel calcolo del percorso minimo	100	18	15	3	-17
	F2.D3: Unione degli algoritmi peso-volume e percorso minimo	90	20	15	5	-25
F3	F3.D1: utilizzare una mappa di un piccolo quartiere sulla quale indicare i punti in cui il corriere deve andare.	95	10	7	3	-30
	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

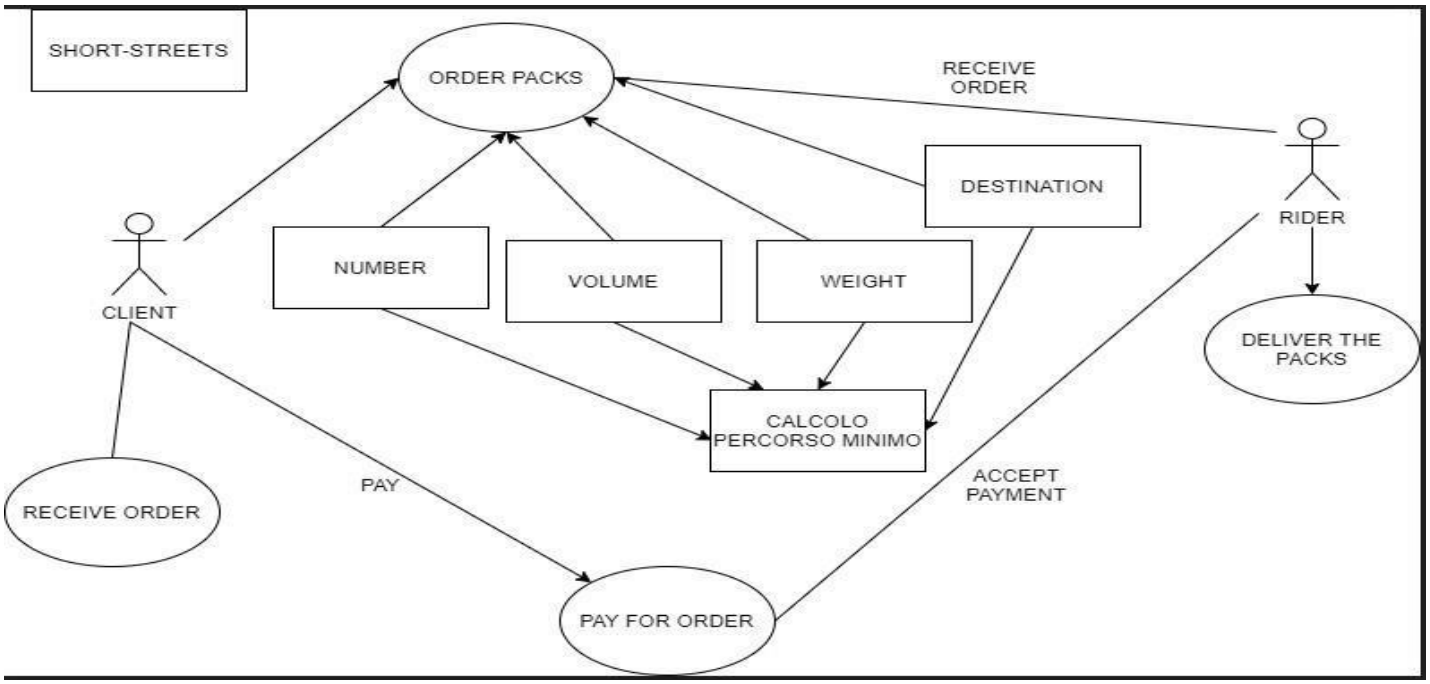
STUDENTE	SINTESI ATTIVITA' SVOLTA	ore svolte	RISULTATI		
			Liv. comp.	% compl.	**VALUTAZIONE**
Bissola Mattia	Testing del software	28	3	50	
Giuggioli Daniel	Interfaccia grafica, classi per la gestione dei pacchi, utilizzo di una mappa su cui visualizzare i punti presso cuisi devono effettuare le consegne	34	3	70	
Majid Zaccaria	Testing del software	28	3	50	
Salvi Alessandro	Sviluppo dell'algoritmo che consente al rider di trasportare contemporaneamente determinatipacchi secondo determinate caratteristiche, quali il peso e il volume.	32	3	90	
Sonzogni Nicolò	Sviluppo dell'algoritmo per il calcolo del percorso minimo che il rider dovrà compiere.	37	3	90	

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

Use Case Diagram

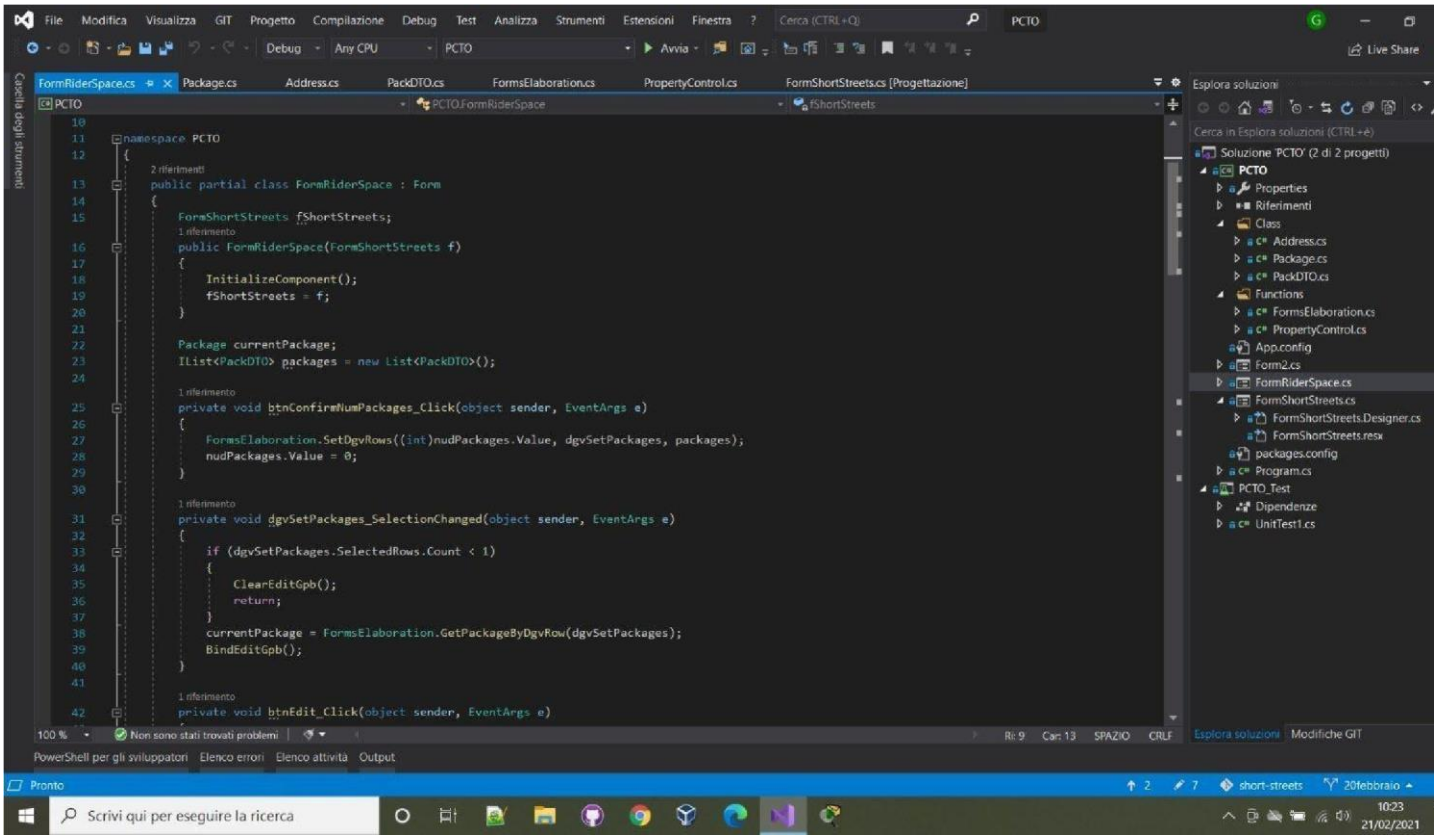
TITOLO PROGETTO	SHORT-STREETS



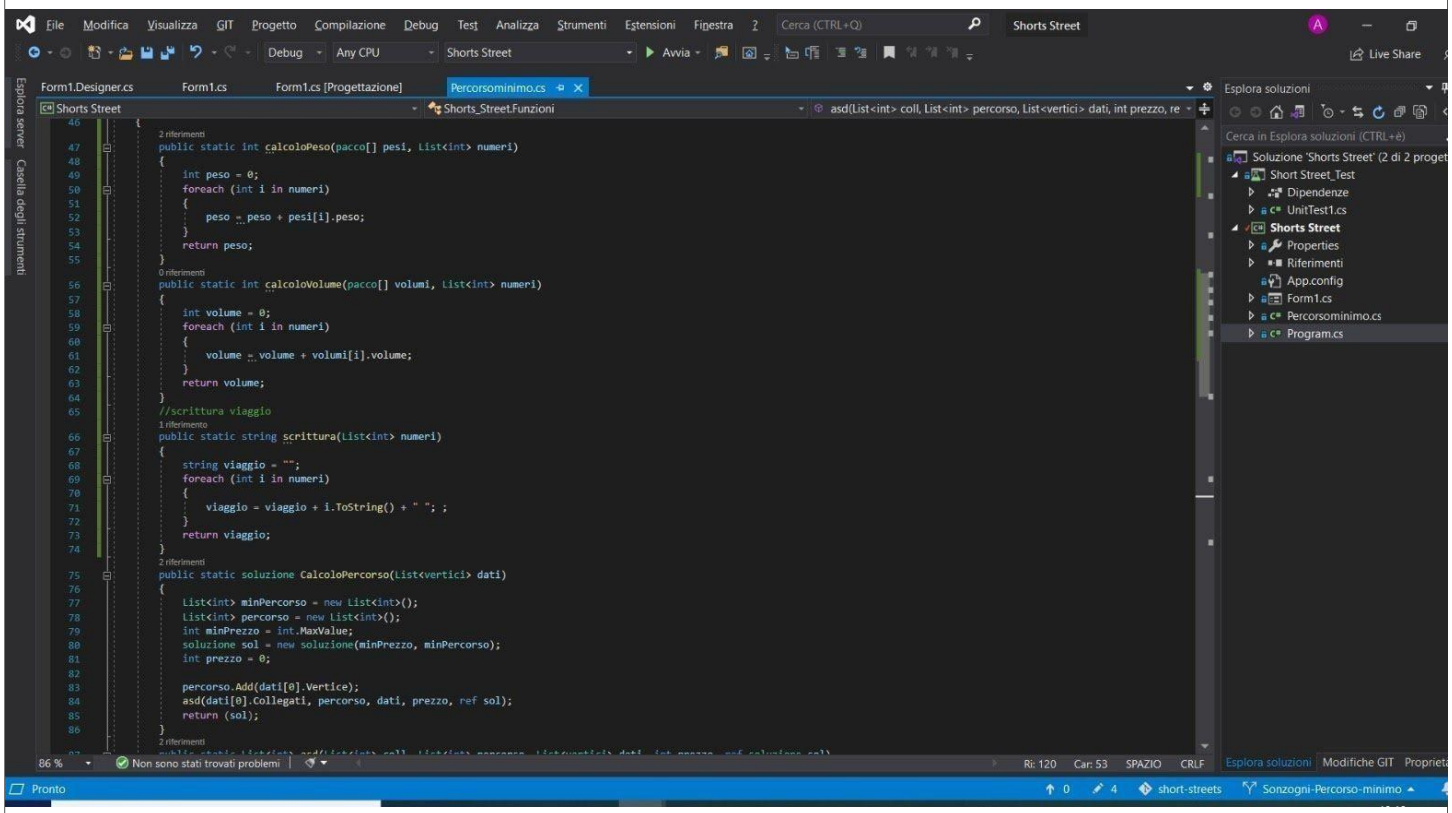
SCREENSHOTS

TITOLO PROGETTO	SHORT-STREETS

F1.D1



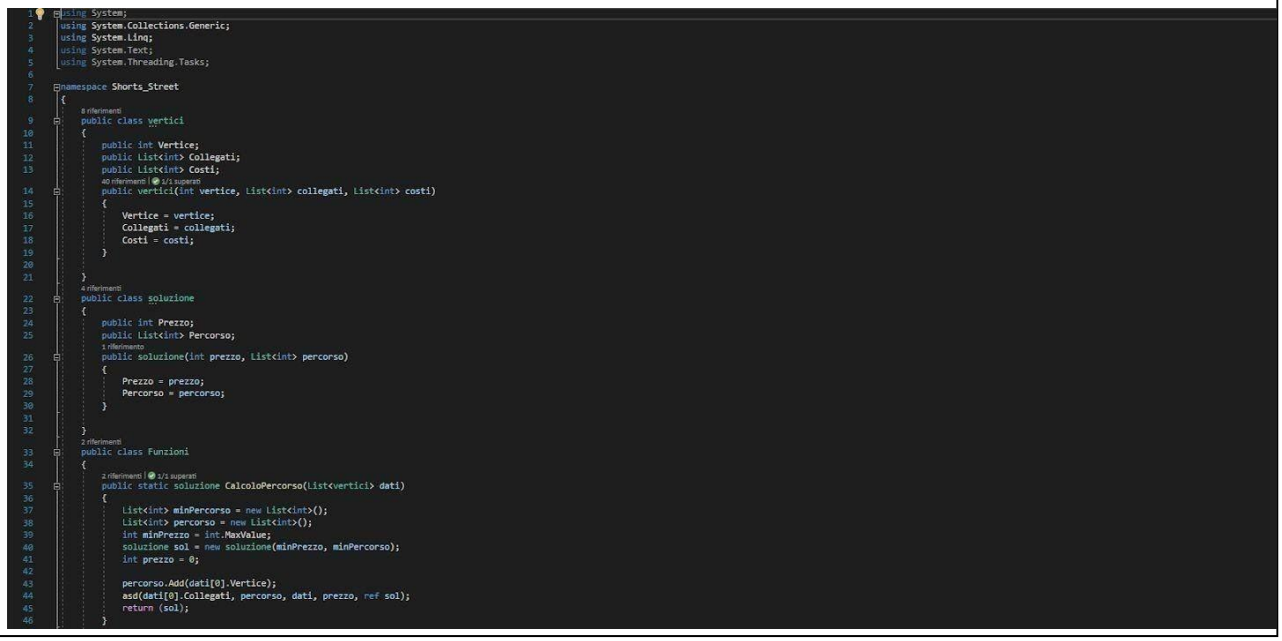
F1.D2



```
{
    2 riferimenti
    public static int calcoloPeso(pacco[] pesi, List<int> numeri)
    {
        int peso = 0;
        foreach (int i in numeri)
        {
            peso = peso + pesi[i].peso;
        }
        return peso;
    }
    0 riferimenti
    public static int calcoloVolume(pacco[] volumi, List<int> numeri)
    {
        int volume = 0;
        foreach (int i in numeri)
        {
            volume = volume + volumi[i].volume;
        }
        return volume;
    }
    //scrittura viaggio
    1 riferimento
    public static string scrittura(List<int> numeri)
    {
        string viaggio = "";
        foreach (int i in numeri)
        {
            viaggio = viaggio + i.ToString() + " "; ;
        }
        return viaggio;
    }
    2 riferimenti
    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);
    }
    2 riferimenti
```

F1.D3



```
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;
        4 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);
        }
```