MISSION IMPOSSIBLE – GESTIRE UNA SCUOLA E' UNA MISSIONE

PROJECT SCOPE:

Il progetto ha lo scopo di **CREARE** un software di gestione per una scuola, con funzionalità di gestire studenti, classi e voti degli studenti nelle materie, prevedendo operazioni di inserimento, modifica, visualizzazione, cancellazione.

Il progetto è stato sviluppato con l'ottica di perseguire la persistenza dei dati, in modo che i dati inseriti siano sempre disponibili.

1 MODALITA' DI SVILUPPO

Metodologia SCRUM

""Per un mondo più snello e agile"

METODOLOGIA SCRUM

- Suddivisione dei tempi di lavoro tramite «Sprint»[1]
- Suddivisione del carico di lavoro tramite backlog
- Utilizzo di software e piattaforme per la gestione del team e del lavoro (Flying Donut e GANTT-PROJECT)

L'uso della metodologia SCRUM prevede la suddivisione in ruoli all'interno di un team interfunzionale che si «passano il lavoro come la palla nel rugby.

FLYING DONUT

Software Scrum e Kanban per la gestione dei progetti

2 LAVORO PRELIMINARE

ANALISI DEI REQUISITI – SWOT – DIAGRAMMI – TIMELINE

ANALISI DEI REQUISITI

Di cosa ha bisogno il progetto?

Requisiti funzionali e requisiti non funzionali

ANALISI DEI REQUISITI

Requisiti funzionali

1. Gestione degli studenti
2. Gestione delle classi
3. Gestione delle valutazioni
4. Gestione del corpo docente
5. Gestione delle materie

Requisiti non funzionali

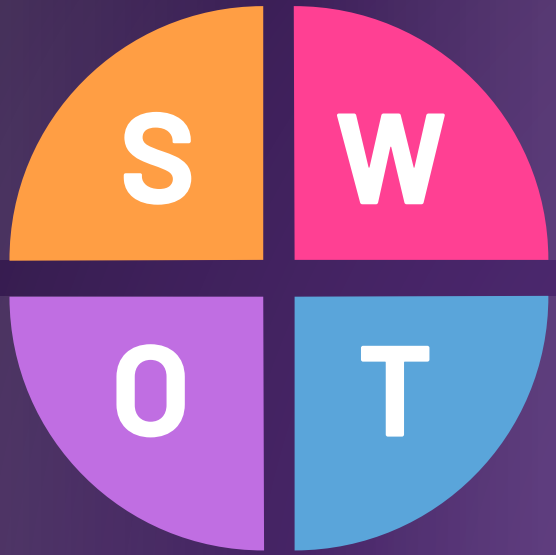
1. Scalabilità
2. Usabilità
3. Documentazione

SWOT ANALYSIS

STRENGTHS

Funzionalità complete

Facilità d'uso



WEAKNESSES

Competenza tecnica

Implementazione con altri prodotti già in uso

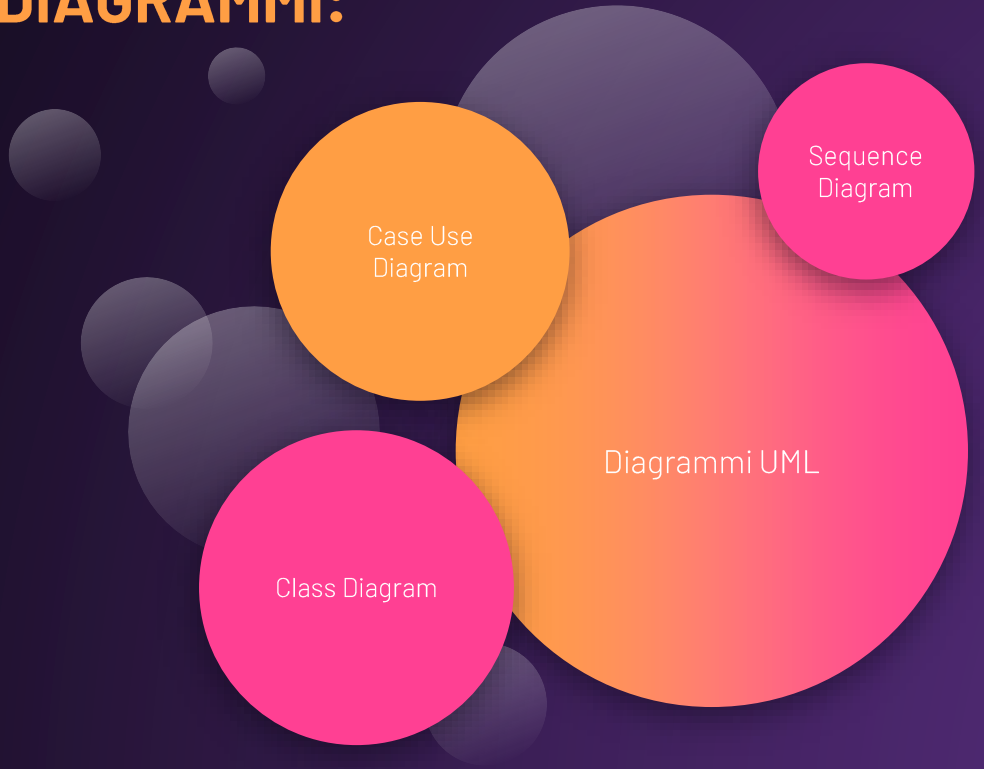
Scalabilità

OPPORTUNITIES

Concorrenza

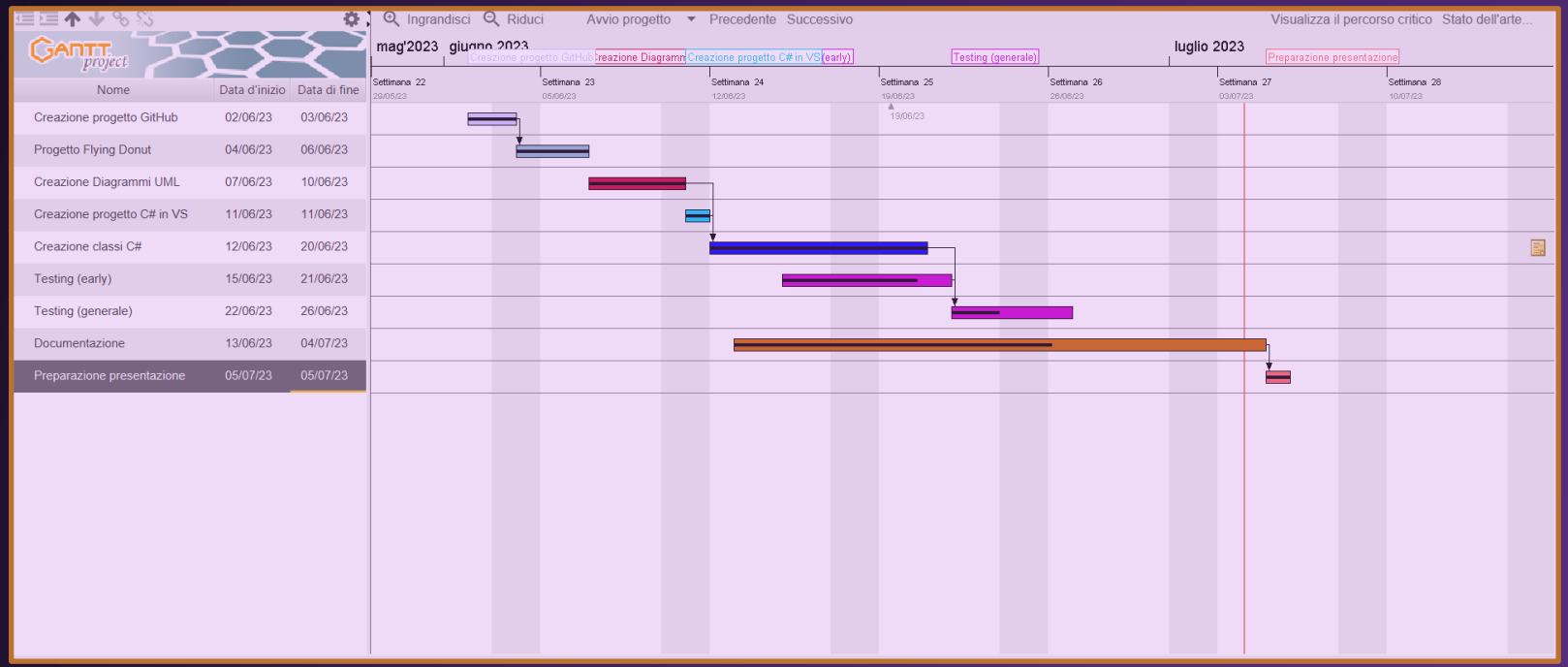
THREATS

DIAGRAMMI:



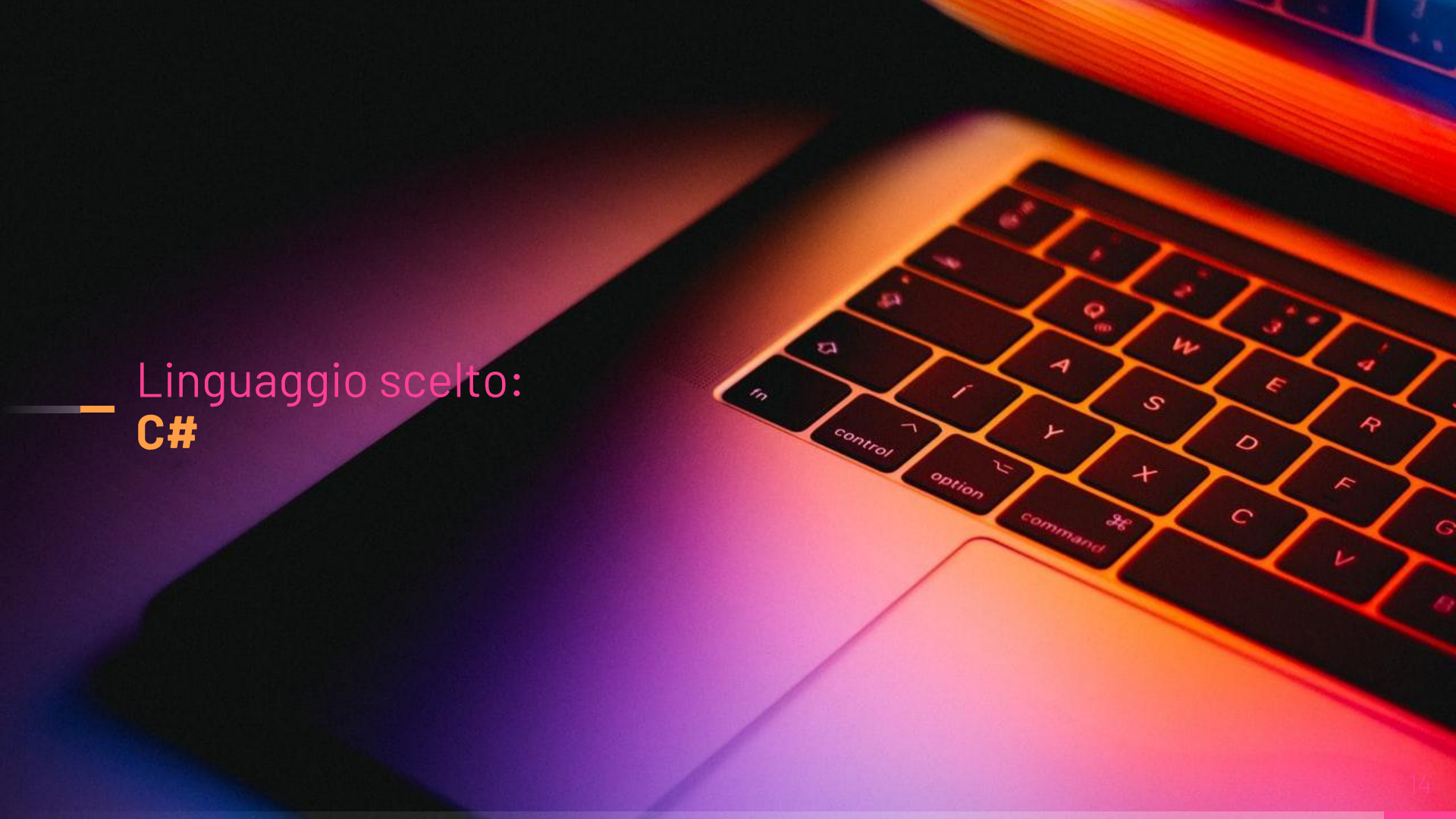
L’utilizzo di diagrammi UML ha aiutato a definire le funzionalità necessarie, le entità e come interagiscono tra loro all’interno del software.

TIMELINE: GANTTPROJECT



3 SCRITTURA DEL SOFTWARE

LINGUAGGIO - CARATTERISTICHE - CLASSI



Linguaggio scelto:
C#

CARATTERISTICHE DEL CODICE

Divisione in classi

La divisione in classi permette una maggiore scalabilità del codice per l'aggiunta di nuove funzionalità e una migliore gestione delle eventuali modifiche.

Newtonsoft Json

L'utilizzo del plugin JSON della Newtonsoft (anche noto come JSON.NET) permette una più semplice manipolazione e serializzazione/ deserializzazione di dati in formato JSON all'interno di applicazioni .NET.

NEWTONSOFT JSON

Newtonsoft.Json è una libreria utilizzata per la gestione dei dati JSON in applicazioni .NET. Offre una serie di funzionalità per lavorare con dati JSON, semplificando il processo di serializzazione e deserializzazione degli oggetti .NET da e verso JSON.
In questo progetto è servita per garantire la persistenza dei dati.

```
Product product = new Product();
product.Name = "Apple";
product.Expiry = new DateTime(2008, 12, 28);
product.Sizes = new string[] { "Small" };

string json = JsonConvert.SerializeObject(product);
// {
//   "Name": "Apple",
//   "Expiry": "2008-12-28T00:00:00",
//   "Sizes": [
//     "Small"
//   ]
// }
```

SCRITTURA DEL CODICE: DETTAGLI

Persistenza dati

Viene garantita la persistenza dei dati con il file Json.

Utilizzo dei controlli – Valutazione massima

Nell'inserimento delle valutazioni, si è aggiunto un controllo per impedire l'inserimento di una valutazione con punteggio maggiore di 100.

Utilizzo dei controlli - ID

Nell'inserimento degli ID definiti dal codice, si è aggiunto un controllo per impedire l'inserimento di lettere e permettere solo numeri interi.

Menu

Per le scelte si è implementato un intuitivo menu

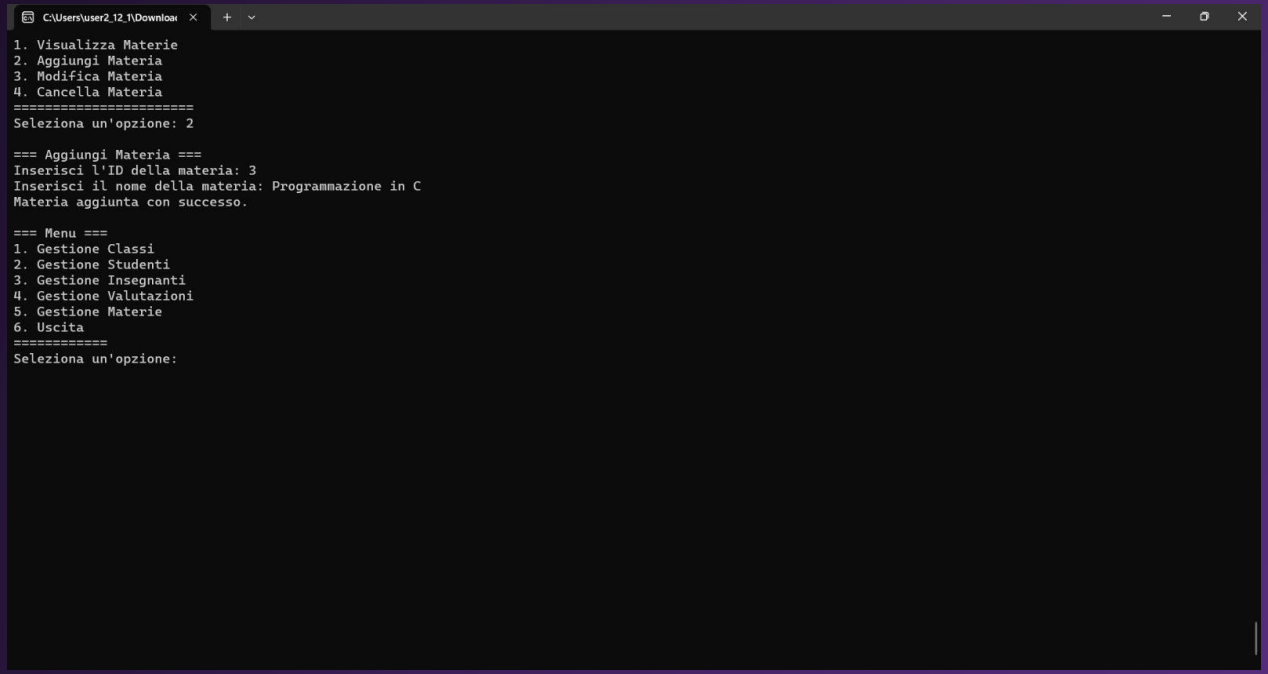
Modificatori di accesso

Si è scelto di usare INTERNAL al posto di PUBLIC per il mantenimento della riservatezza da parte esterna ma permettendo l'accesso da dentro l'assembly.

CLASSI

PROGRAM	STUDENTE	INSEGNANTE	MATERIA	CLASSE	DATA
Classe principale, quella che gestisce il menù, il main e il salvataggio del file JSON attraverso Liste.	Classe che gestisce l'inserimento, la modifica, la visualizzazione e la cancellazione degli studenti	Classe che gestisce l'inserimento, la modifica, la visualizzazione e la cancellazione degli insegnanti.	Classe che gestisce l'inserimento, la modifica, la visualizzazione e la cancellazione delle materie.	Classe che gestisce l'inserimento, la modifica, la visualizzazione e la cancellazione delle classi.	Classe che gestisce le liste Classi, Studenti, Insegnanti e Valutazioni per il JSON.

RISULTATO?


C:\Users\user2_12_1\Downloa
1. Visualizza Materie
2. Aggiungi Materia
3. Modifica Materia
4. Cancella Materia
==========
Seleziona un'opzione: 2
=== Aggiungi Materia ===
Inserisci l'ID della materia: 3
Inserisci il nome della materia: Programmazione in C
Materia aggiunta con successo.
=== Menu ===
1. Gestione Classi
2. Gestione Studenti
3. Gestione Insegnanti
4. Gestione Valutazioni
5. Gestione Materie
6. Uscita
==========
Seleziona un'opzione:

4 TESTING

PROCESSI DI TESTING – PROGRAMMI – APPLICAZIONE

TESTING:

Modalità:

Per controllare che il codice funzioni come si deve, si sono eseguiti unit test specifici per i metodi interessati.

Implementazione:

I test sono implementati utilizzando il framework di testing "Microsoft.VisualStudio.TestTools.UnitTesting".

I test utilizzano la simulazione dell'input e dell'output della console tramite l'utilizzo di "StringReader" e "Console.SetIn" per i test interattivi.

".

AAA:

ARRANGE, ACT e ASSERT.

I test controllano le asserzioni utilizzando il metodo "Assert" del framework di testing per confrontare i risultati attesi con i risultati ottenuti.
Ogni test è indipendente dagli altri e non dipende dall'ordine di esecuzione.

RISULTATO?

```
[TestMethod]
● | 0 riferimenti
public void AggiungiStudente_AddStudent()
{
    // Arrange
    List<Studente> studentiTest = new List<Studente>();

    // Simulazione dell'input dell'utente
    string input = "1\nMario\nRossi\n1995-07-15\n2022\nSeconda\n";

    // Simulazione della console input/output
    using (StringReader sr = new StringReader(input))
    {
        Console.SetIn(sr);

        // Act
        AggiungiStudente(studentiTest);

        // Assert
        Assert.AreEqual(1, studentiTest.Count);
        Assert.AreEqual(1, studentiTest[0].IDStudente);
        Assert.AreEqual("Mario", studentiTest[0].Nome);
        Assert.AreEqual("Rossi", studentiTest[0].Cognome);
        Assert.AreEqual(new DateOnly(1995, 7, 15), studentiTest[0].DataNascita);
        Assert.AreEqual(2022, studentiTest[0].AnnoCorso);
        Assert.AreEqual("Seconda", studentiTest[0].NomeClasse);
    }
}
1 riferimento | ● 1/1 superati
public static void AggiungiStudente(List<Studente> studenti)
{
    Console.WriteLine("=== Aggiungi Studente ===");
    Console.Write("Inserisci l'ID dello studente: ");
    int id;
    while (!int.TryParse(Console.ReadLine(), out id))
    {
        Console.WriteLine("Inserisci un ID valido (numero intero).");
        Console.Write("Inserisci l'ID dello studente: ");
    }

    Console.Write("Inserisci il nome dello studente: ");
    string nome = Console.ReadLine();
    Console.Write("Inserisci il cognome dello studente: ");
    string cognome = Console.ReadLine();
    Console.Write("Inserisci la data di nascita dello studente: ");
    DateOnly dataNascita = DateOnly.Parse(Console.ReadLine());
    Console.Write("Inserisci l'anno di corso dello studente: ");
    int annoCorso = int.Parse(Console.ReadLine());
    Console.Write("Inserisci la classe dello studente: ");
    string nomeClasse = Console.ReadLine();

    Studente nuovoStudente = new Studente(id, nome, cognome, dataNascita, annoCorso, nomeClasse);
    studenti.Add(nuovoStudente);

    Console.WriteLine("Studente aggiunto con successo.");
}
```

PRESENTAZIONE DEL TEAM

Carlotta Paola Marino
SCRUM MASTER

Michael Furnari
DIAGRAMMI UML

Riccardo Gentile
PROGRAMMATORE

Andrea Lanzavecchia
TESTER/DOCUMENTATORE

Samuele Zambarelli
PROGRAMMATORE

GRAZIE DELL'ATTENZIONE!