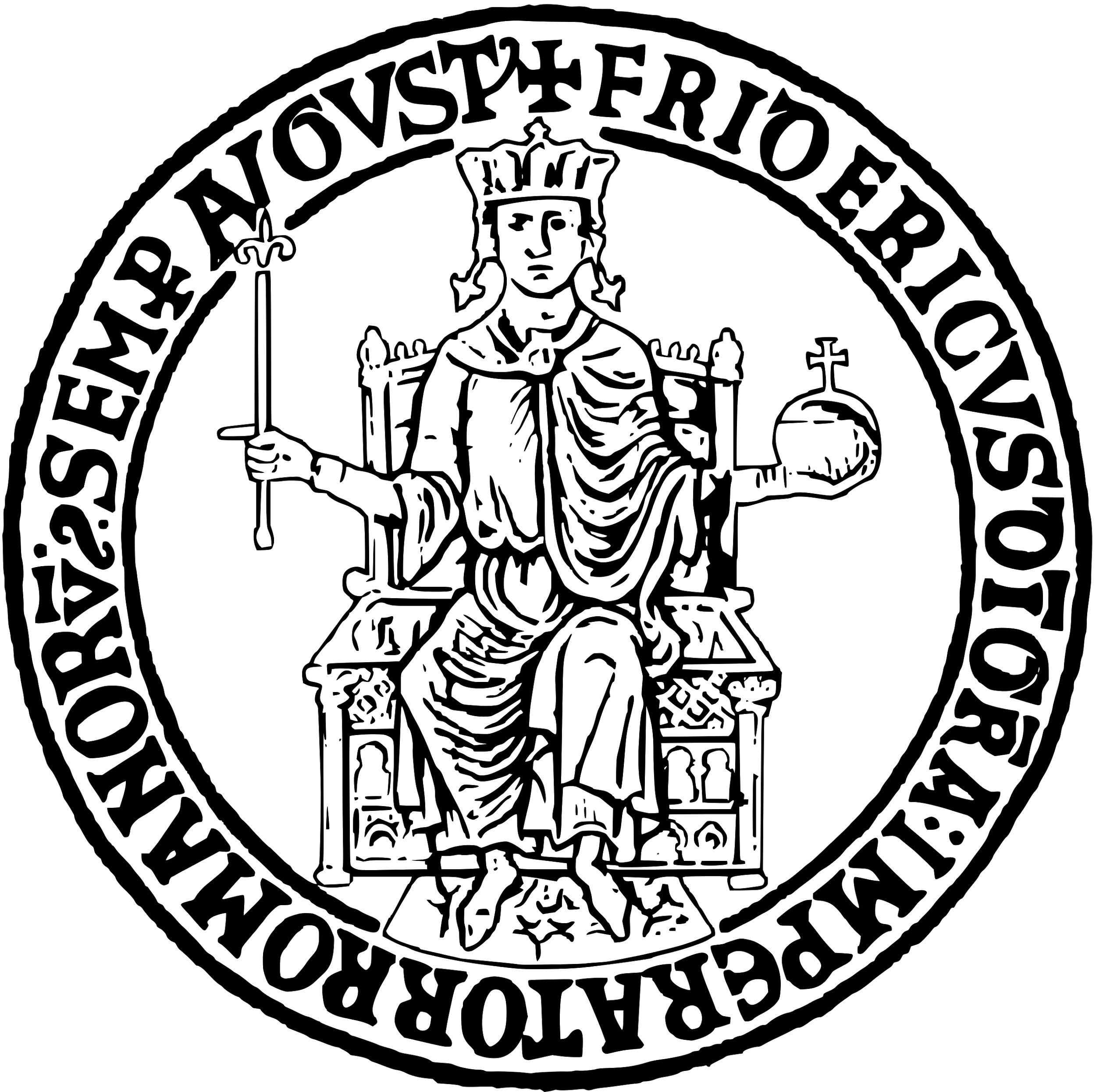
Università degli studi di Napoli Federico II  
Scuola Politecnica e delle Scienze di Base

Dipartimento di Ingegneria Elettrica

e Tecnologie dell’Informazione

Anno accademico 2020/2021

Progettazione e sviluppo di una base di dati relazionale per la gestione di progetti.

Esposito Sara Trinchillo Giusi Ornella Beniamino Gentile

N86003374 N86003950 N86003004

**Indice**

1. **Descrizione del progetto**

**1.1** Analisi progetto

1. **Progettazione concettuale**
2. Analisi di un class diagram
3. Class Diagram
4. Alcune precisazioni sul class diagram
5. Ristrutturazione del class diagram
6. Chiavi primarie
7. Attributi multipli
8. Attributi strutturati
9. Attributi derivati
10. Gerarchie di specializzazione
11. **Dizionari dei dati**
12. Dizionario delle classi
13. Dizionario delle associazioni
14. Dizionario dei vincoli

**Capitolo 1**

**Descrizione del progetto**

1. **Analisi del problema**

Si progetti ed implementi una base di dati relazionale, ovvero uno schema concettuale di dati strutturati, che possa essere d’aiuto alla gestione e memorizzazione di progetti realizzati da un’azienda.   
 Bisogna tener traccia dei dipendenti della suddetta azienda ricordando che ognuno di essi ha un ruolo, in particolare per ogni progetto ci sarà un unico project manager.

Il progetto avrà una specifica tipologia, che può essere: “Ricerca di base”, “Ricerca Industriale”, “Ricerca sperimentale”, “Sviluppo Sperimentale”, ed uno o più ambiti, tra cui: Economia, Medicina, Informatica.  
 Per ogni dipendente ci sarà un identificativo reale: il codice fiscale.  
 Al fine di collaborare alla realizzazione del progetto ogni gruppo potrà partecipare a dei meeting telematici o fisici.  
 Il project manager deve poter accedere ai registri delle presenze dei propri colleghi, alle loro skill e al loro salario al fine dell’organizzazione dei gruppi e della valutazione aziendale di ogni singolo membro.

**Capitolo 2**

**Progettazione concettuale**

1. **Class Diagram**

Al fine di semplificare la lettura si è scelto di:

● Esplicitare le enumerazioni al lato destro nella Fig.1 e in basso nella Fig.2.

● Non impostare la lunghezza degli attributi di tipo VARCHAR a meno che non sia di fondamentale importanza, come nel caso del codice fiscale.

● Le cardinalità delle associazioni indicate con \* specificano che quella classe può partecipare da un minimo di zero a un massimo di molti, ossia (0,\*).

● Dare un solo nome alle associazioni invece di due (uno per direzione).

1. **Analisi di un class diagram**

Un class diagram è un diagramma che descrive la struttura di un sistema. Per apprendere appieno cosa è un class diagram bisogna capire da cosa è formato.  
Le **classi** rappresentano delle entità. Il nome della classe indica la categoria di entità descritta. Ogni classe è corredata da un insieme di **attributi**, che descrivono in maniera univoca le caratteristiche o lo stato degli oggetti della classe, e **metodi** che descrivono il comportamento di un’entità.

Gli attributi sono definiti da un nome, un tipo e una molteplicità.  
Il simbolo grafico che rappresenta le classi è un rettangolo suddiviso in tre scomparti, rispettivamente dedicati al nome della classe, agli attributi e metodi.

Due classi possono essere legate da **relazioni**, esistono diversi tipi di relazione:

* **associazioni:** indicano che due classi o più (nel caso di associazione ternaria) condividono una relazione statica. Le associazioni sono identificate nel class diagram da una o più rette consecutive che collegano due o più classi.
* **specializzazioni** indicano che una delle due classi è una generalizzazione dell’altra (superclasse), ovvero che eredita le caratteristiche (attributi e metodi) della superclasse e le fa proprie.   
  Le specializzazioni sono identificate da una freccia (⎯ᐅ ) con la punta verso la classe padre.  
  Esistono diversi tipi di specializzazione:

*Totale*: la classe padre è obbligatoriamente una delle classi figlie.

*Parziale*: la classe padre può non essere nessuna delle classi figlie.

*Disgiunt*a: la classe padre non può essere più classi figlie insieme.

*Overlap*: la classe padre può essere più classi figlie insieme.

1. **Alcune precisazioni sul class diagram**

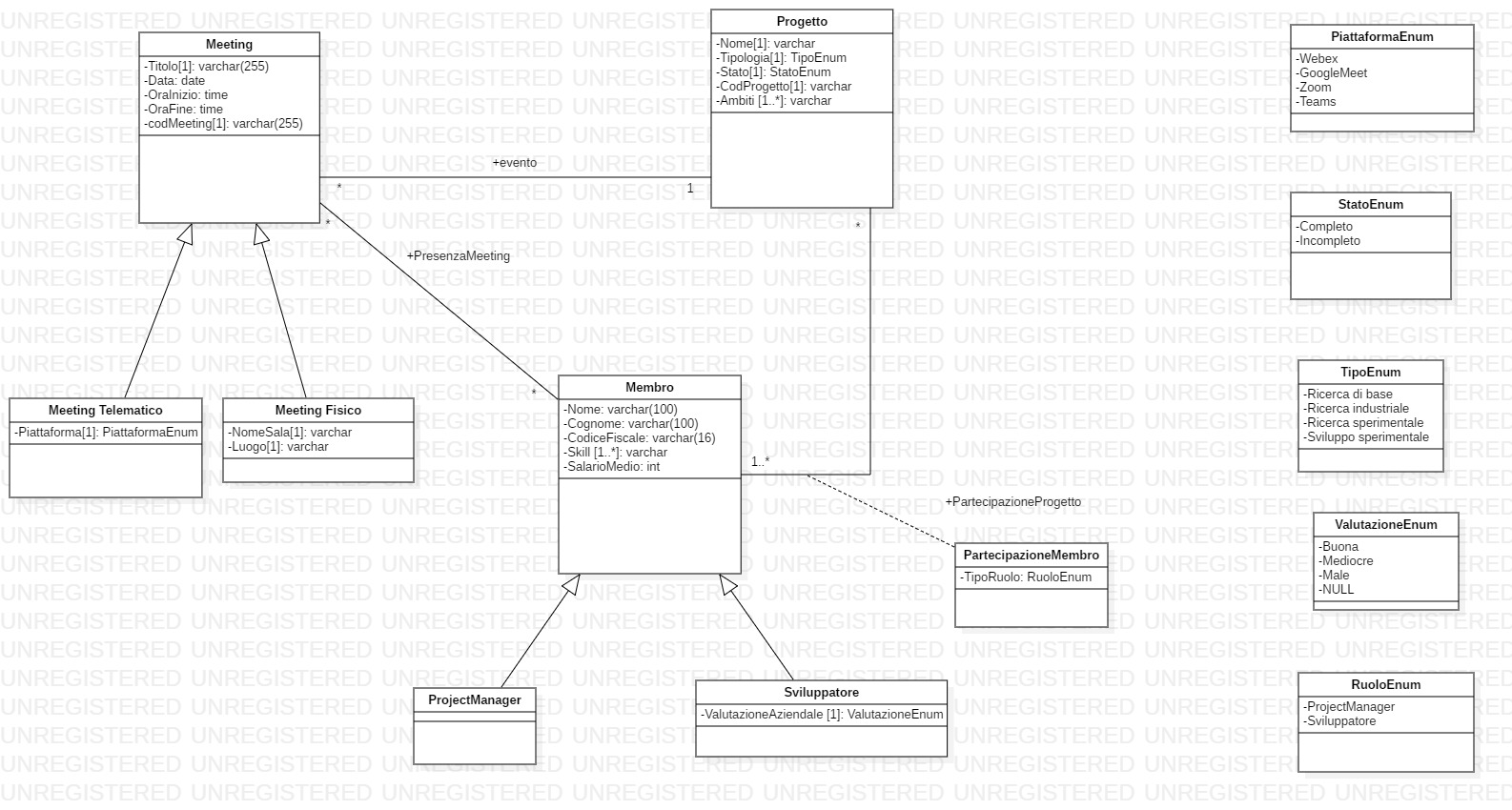
Dal class diagram (Fig.1) si può osservare che sono state create tre classi principali per le entità **membro**, **progetto**, **meeting**, due classi di specializzazione di meeting: **meeting telematico** e **meeting fisico e** due classi di specializzazione di membro: **ProjectManager** e **Sviluppatore**.

Tra meeting e membro è stata inserita l’associazione **PresenzaMeeting**, questa associazione è chiamata così perché esplicita:

* quanti membri partecipano ai meeting con cardinalità (2,\*) poiché un meeting con meno di due membri non ha senso di esistere.
* a quanti meeting un membro ha preso parte con cardinalità (0,\*) perché se un membro lavora a un progetto da solo (cosa che è possibile fare secondo l’associazione PartecipazioneProgetto) non ha bisogno di partecipare a meeting.

L’associazione **PartecipazioneProgetto** ha il compito di collegare la classe membro e la classe progetto con cardinalità:

* (1,\*) perché un progetto può essere formato da un solo membro.
* (\*) perché un membro può partecipare a nessuno (essere in vacanza o periodo di stop) o più progetti.



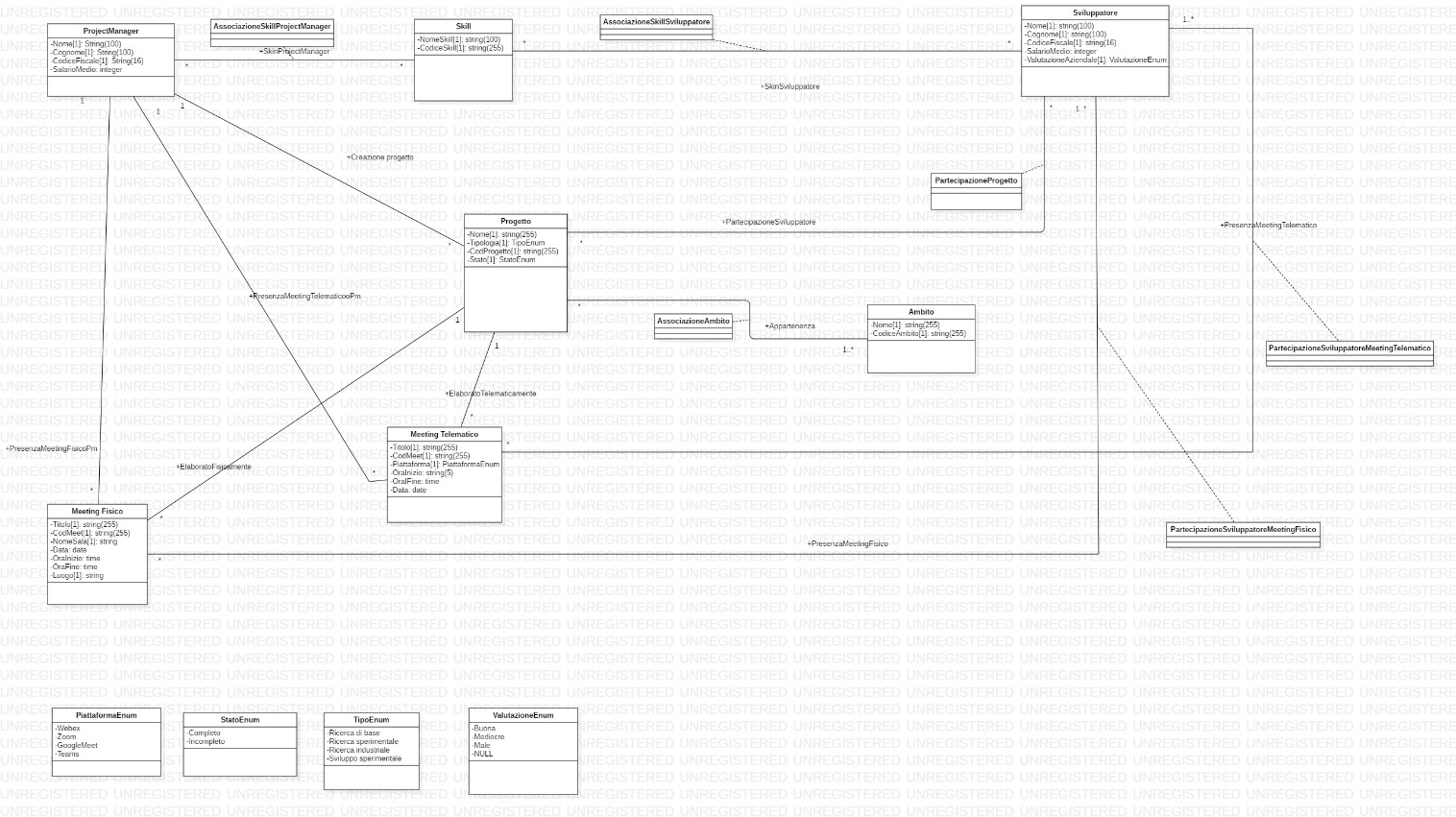
**Fig.1**

1. **Ristrutturazione del class diagram**

Al fine di rendere il class diagram idoneo alla traduzione in schemi relazionali e di migliorare l’efficienza dell’implementazione si procede alla ristrutturazione dello stesso.

È necessario analizzare le chiavi primarie e fare in modo che siano univoche e più sintetiche possibili. Anche gli attributi derivati, se possibile, devono essere omessi.

Al termine del procedimento di ristrutturazione il class diagram (Fig.2) non conterrà attributi strutturati, attributi multipli, gerarchie di specializzazione e associazioni.



**Fig.2**

1. **Chiavi Primarie**

Nel class diagram ristrutturato si è deciso, al fine di agevolare l’identificazione, di aggiungere un attributo destinato a essere la **chiave primaria** della classe qualora la chiave primaria fosse composta.  
Ad esempio per le classi “progetto” e “meeting” sono state aggiunte le chiavi primarie “CodMeet” e “CodProgetto”.

1. **Attributi Multipli**

È necessario gestire gli **attributi multipli** in quanto questi prevedono la possibilità di contenere più valori. A causa dell’impossibilità di stimare il numero di diversi attributi associati creeremo una nuova classe contenente l’attributo.  
 Per esempio la classe “Membro” contiene l’attributo “Skill”, quest’ultimo è un attributo multiplo in quanto a un membro potrebbero essere associate più skills. Per evitare di avere un attributo di cui non si sa con precisione a che valore riferirsi, si crea una classe “**Skill**” apposita.  
 Un altro attributo multiplo proviene dalla classe “Progetto”: ambito. Un progetto, infatti, può trattare più ambiti. Anche questa volta creiamo una classe per “**Ambito**”. Facendo ciò però causeremo un’associazione molti a molti, poiché un progetto può includere più ambiti e un ambito è trattato da più progetti.

1. **Attributi Derivati**

Gli attributi derivati sono attributi derivabili o calcolabili da altri.  
Un esempio potrebbe essere la data di nascita e l’età, è chiaro che l’età è un attributo derivabile dalla data di nascita. Possiamo rimuovere un attributo derivato nel class diagram, ma possiamo anche mantenere tale attributo per migliorare la comprensibilità.   
Nel class diagram mostrato non ci sono attributi derivati

1. **Attributi Strutturati**

Gli attributi strutturati, come gli attributi multipli, devono essere rimossi nel class diagram ristrutturato in quanto questi creano dei problemi quando si desidera conoscere i singoli campi dell’attributo.  
 Nel nostro Class Diagram non abbiamo attributi strutturati, ma un esempio poteva essere la residenza di un membro.  
E’ un attributo strutturato in quanto composto dalla via, dal numero civico, dal cap e dalla città di residenza.

Si potrebbe risolvere il problema due modi: il primo è di creare tanti attributi per tutti i campi dell’attributo strutturato, il secondo è creare una nuova classe “Residenza” avente gli attributi che corrispondono ai campi del vecchio attributo strutturato.

1. **Gerarchie di Specializzazione**

Nel primo class diagram si hanno due tipi di specializzazioni. Esistono più modi per eliminare le specializzazioni, si è scelto di accorpare la classe padre nelle classi figlie per la generalizzazione “Meeting”, infatti non troviamo più la classe “Meeting” ma i suoi attributi sono stati trasferiti nelle due classi figlie: Meeting Telematico e Meeting Fisico. Lo stesso è avvenuto per la classe Membro e le sue specializzazioni, sono diventate due classi: Sviluppatore e ProjectManager.

**Capitolo 3**

1. **Dizionario delle classi**

|  |  |  |
| --- | --- | --- |
| **Classe** | **Descrizione** | **Attributo** |
| **ProjectManager** | È la classe che descrive un impiegato di un’azienda nel ruolo di Project Manager. Ha il compito dell’organizzazione aziendale. | **CodiceFiscale**: (String(16)) *Chiave primaria* cheidentifica univocamente i project manager.  **Nome**: (String(100)) nome associato al project manager.  **Cognome**: (String(100)) cognome associato al project manager.  **SalariaMedio**: (integer) il salario medio che riceve il project manager. |
| **Sviluppatore** | È la classe che descrive un impiegato di un’azienda nel ruolo di sviluppatore. Ha il compito di svolgere le mansioni che gli vengono affidate. | **CodiceFiscale**: (String(16)) *Chiave primaria* cheidentifica univocamente uno sviluppatore.  **Nome**: (String(100)) nome associato allo sviluppatore.  **Cognome**: (String(100)) cognome associato allo sviluppatore.  **SalariaMedio**: (integer) il salario medio che lo sviluppatore riceve.  **ValutazioneAziendale**: (ValutazioneEnum) la valutazione che un project manager associa ad uno sviluppatore. |
| **Progetto** | È la classe che descrive un progetto realizzato o da realizzare dai membri di un’azienda. | **CodProgetto**: (String(255)) *Chiave* *primaria* cheidentifica univocamente un progetto*.*  **Nome**: (String(255)) nome dato al progetto. **Tipologia**: (TipoEnum) indica il tipo di progetto.  **Stato**: (StatoEnum) indica se il progetto è stato portato a termine o meno. |
| **Meeting Fisico** | È la classe che identifica un meeting tra i membri di un’azienda, di tipo fisico. | **CodMeet**: (String(255)) *Chiave* *primaria* cheidentifica univocamente un meeting fisico.  **Titolo**: (String(255)) nome assegnato ad un meeting fisico come titolo.  **NomeSala**: (String) nome della sala dove si svolge il meeting.  **Data**: (Date) il giorno, il mese e l’anno in cui si svolge il meeting.  **OraInizio**: (Time) l’orario di inizio del meeting.  **OraFIne**: (Time) l’ora di fine del meeting.  **Luogo**: (String) il luogo in cui si svolge il meeting. |
| **Meeting Telematico** | È la classe che identifica un meeting tra i membri di un’azienda, di tipo telematico. | **CodMeet**: (String(255)) *Chiave* *primaria* che identifica univocamente un meeting telematico.  **Titolo**: (String(255)) nome assegnato ad un meeting telematico come titolo.  **Piattaforma**: (PiattaformaEnum) il sito/app in cui si svolge il meeting telematico.  **Data**: (Date) il giorno, il mese e l’anno in cui si svolge il meeting.  **OraInizio**: (Time) l’orario di inizio del meeting.  **OraFIne**: (Time) l’ora di fine del meeting. |
| **Skill** | È la classe che identifica le abilità di un membro (Project manager o sviluppatore). | **CodiceSkill**: (String (255)) *Chiave* *primaria* cheidentifica univocamente una skill.  **NomeSkill**: (String(100)) il nome di una skill. |
| **Ambito** | È la classe che identifica l’ambito di un progetto. | **CodiceAmbito:** (String (255)) *Chiave primaria* che identifica univocamente un ambito.  **NomeAmbito**: (String(255)) il nome di un ambito. |

1. **Dizionario delle associazioni**

|  |  |  |
| --- | --- | --- |
| **Associazioni** | **Descrizione** | **Classi coinvolte** |
| **CreazioneProgetto** | Esprime la possibilità per un project manager di creare un progetto. | **Progetto[\*]** ruolo **è creato**: indica il progetto che viene creato dal project manager.  **ProjectManager[1]** ruolo **crea** :indica il project manager che crea il progetto. |
| **Partecipazione**  **ProgettoSviluppatore** | Esprime la possibilità per uno sviluppatore di partecipare a un progetto. | **Sviluppatore[\*]**ruolo **lavora ad**: indica lo sviluppatore che lavora ad un progetto.  **Progetto[\*]** ruolo **viene realizzato**: indica il progetto che viene realizzato da uno o più sviluppatori. |
| **PresenzaMeetingTelematico** | Esprime la possibilità da parte di uno sviluppatore di partecipare a un meeting in via telematica. | **Sviluppatore[1..\*]** ruolo **partecipa**: indica lo sviluppatore che presenza a un meeting telematico.  **MeetingTelematico[\*]** ruolo **ha partecipanti**: indica il meeting telematico che ha come partecipanti degli sviluppatori. |
| **PresenzaMeetingFisico** | Esprime la possibilità da parte di uno sviluppatore di partecipare ad un meeting fisico. | **Sviluppatore[1..\*]** ruolo **partecipa**: indica lo sviluppatore che presenza a un meeting fisico.  **MeetingFisico[\*]** ruolo **ha partecipanti**: indica il meeting fisico che ha come partecipanti degli sviluppatori. |
| **PresenzaMeetingFisicoPm** | Esprime la possibilità da parte di un project manager di partecipare ad un meeting in via telematica. | **ProjectManager[1]** ruolo **organizzatore**: indica il project manager che organizza un meeting fisico.  **MeetingFisico[\*]** ruolo **è organizzato**: indica il meeting fisico che viene organizzato da un solo project manager. |
| **PresenzaMeetingTelematicoPm** | Esprime la possibilità da parte di un project manager di partecipare ad un meeting fisico. | **ProjectManager[1]** ruolo **organizzatore**: indica il project manager che organizza un meeting  telematico.  **MeetingTelematico[\*]** ruolo **è organizzato**: indica il meeting telematico che viene organizzato da un solo project manager. |
| **ElaboratoTelematicamente** | Esprime la possibilità di elaborare un progetto durante un meeting telematico. | **MeetingTelematico[\*]** ruolo **permette elaborazione**: indica il meeting telematico dove viene elaborato un progetto.  **Progetto[1]** ruolo **viene elaborato**: indica il progetto che viene elaborato durante un meeting telematico. |
| **ElaboratoFisicamente** | Esprime la possibilità di elaborare un progetto durante un meeting fisicamente. | **MeetingFisico[\*]** ruolo **permette elaborazione**: indica il meeting fisico dove viene elaborato un progetto.  **Progetto[1]** ruolo **viene elaborato**: indica il progetto che viene elaborato durante un meeting fisico. |
| **SkillSviluppatore** | Esprime le abilità degli sviluppatori. | **Sviluppatore[\*]** ruolo **ha**: indica gli sviluppatori che possiedono una determinata skill.  **Skill[\*]** ruolo **è posseduta**: indica le skill sono possedute da uno sviluppatore. |
| **SkillProjectManager** | Esprime le abilità dei project manager. | **ProjectManager[\*]** ruolo **ha**: indica i project manager che possiedono una determinata skill.  **Skill:** [\***]** ruolo **è posseduta**: indica le skill che sono possedute da un project manager. |
| **Appartenenza** | Esprime l’appartenenza di un progetto a uno o più ambiti. | **Ambito[\*]** ruolo **appartiene**: indica l’ambito a cui appartiene un progetto.  **Progetto[\*]** ruolo **ha appartenenza**: indica il progetto che ha appartenenza a un determinato ambito. |