**UNIVERSITÀ POLITECNICA DELLE MARCHE**
**FACOLTÀ DI INGEGNERIA**
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica e dell'Automazione


**TESI DI LAUREA**

# Progettazione e realizzazione di un servizio di navigazione tramite Graph Neural Network

# Design and implementation of a navigation service through a Graph Neural Network

Relatore

Dott. Enrico Corradini

Candidato

Paolo Roselli

**ANNO ACCADEMICO 2023-2024**

*Se ho visto oltre, è perché sono salito sulle spalle dei Giganti*

Isaac Newton

**Sommario**

Questa tesi presenta il progetto "NavigaUnivpm", un'applicazione innovativa sviluppata per migliorare l'esperienza di navigazione all'interno del campus universitario, sfruttando la realtà aumentata e tecnologie avanzate come le Graph Neural Networks (GNN). L'obiettivo principale è fornire indicazioni precise e tempestive agli utenti, facilitando il raggiungimento dei punti di interesse in modo rapido ed efficiente. Le GNN giocano un ruolo fondamentale nell'ottimizzazione del calcolo del percorso, permettendo di analizzare in modo intelligente i dati provenienti dai nodi e dagli archi della rete di navigazione, modellando relazioni complesse tra i punti di riferimento del campus. Grazie alla capacità della rete neurale di apprendere da strutture di grafi, il sistema è in grado di adattarsi dinamicamente ai cambiamenti nell'ambiente circostante, garantendo percorsi ottimizzati e personalizzati. La tesi affronta le sfide legate all'analisi dei requisiti, alla progettazione del sistema e all'integrazione del feedback degli utenti, con particolare attenzione all'impiego delle Graph Neural Network per migliorare l'efficienza della navigazione. Inoltre, viene discussa l'importanza dell'ottimizzazione delle prestazioni e della raccolta di dati per assicurare un servizio reattivo ed efficace, migliorando così l'esperienza accademica di studenti e visitatori. I risultati, ottenuti attraverso una serie di test e valutazioni, dimostrano come l'utilizzo delle GNN abbia contribuito a migliorare sensibilmente la precisione e l'usabilità del sistema, aprendo prospettive promettenti per ulteriori sviluppi dell'intelligenza artificiale applicata alla navigazione intelligente.

**Keyword**: Graph Neural Networks, Intelligenza artificiale, Ottimizzazione del percorso, Realtà aumentata, Applicazione mobile

# Indice

# Elenco delle figure

# Introduzione

Nel contesto storico attuale, si osserva una crescente integrazione tra sistemi basati sull'intelligenza artificiale e le attività umane, sia quotidiane che straordinarie. Esaminando la vita quotidiana, è evidente che le interazioni con sistemi intelligenti, sia dirette che indirette, avvengono innumerevoli volte. Il concetto di intelligenza artificiale ha le sue origini nel XVII secolo, con il matematico e filosofo René Descartes (1596-1650), che propose la teoria del dualismo mente-corpo. Tuttavia, il principio moderno dell'IA si sviluppa negli anni '50 del XX secolo, quando il matematico inglese Alan Turing sollevò la domanda: "Può una macchina pensare?" Turing *et al.* [1950]. Da qui prese avvio lo sviluppo del cosiddetto "Test di Turing", concepito per determinare se una macchina fosse in grado di esibire un comportamento intelligente.

Nonostante ciò, l'uso massiccio di modelli di IA nella vita quotidiana si è diffuso solo a partire dalla Conferenza di Dartmouth del 1956, durante la quale John McCarthy, Marvin Minsky, Nathaniel Rochester e Claude Shannon diedero inizio alla discussione accademica sull'IA, affermando che: "[...] ogni aspetto dell'apprendimento o della caratteristica dell'intelligenza doveva essere descritto con precisione, in modo da poter essere simulato con una macchina" Moor [2006].

L'introduzione dell'intelligenza artificiale ha reso la vita più organizzata, facilitando la gestione fluida delle attività. Un esempio di utilizzo diretto quotidiano è rappresentato dagli assistenti vocali sugli smartphone: oltre a eseguire le azioni richieste, analizzano le abitudini degli utenti per suggerire e velocizzare operazioni ricorrenti. Per quanto riguarda l'utilizzo indiretto di questi sistemi, esso è persino più diffuso di quanto si possa pensare, basti pensare all'uso delle e-mail, strumento ormai onnipresente. Un'altra innovazione significativa è rappresentata dalle tecnologie che prevengono i principali attacchi informatici, come whaling, smishing e vishing, tramite il filtraggio delle mail pericolose. Secondo un report di Microsoft, l'IA di Microsoft Defender ha bloccato oltre 37 miliardi di e-mail dannose nel 2022, di cui 9 miliardi erano tentativi di phishing, con un incremento del 25% rispetto all'anno precedente[1].

Attualmente, la maggior parte dei sistemi di intelligenza artificiale basati su reti neurali utilizza le Convolutional Neural Networks (CNN), particolarmente efficaci nel riconoscimento e nell'elaborazione delle immagini, o le Recurrent Neural Networks (RNN), che eccellono nell'analisi di dati sequenziali. Le Graph Neural Networks (GNN), invece, costituiscono un campo emergente, ancora oggetto di ricerca, con il potenziale di rivoluzionare l'analisi dei dati strutturati, come reti e grafi. Uno degli obiettivi principali delle GNN è la creazione di modelli di IA in grado di supportare i sistemi di navigazione, grazie alla loro naturale adattabilità a tali compiti.

[1] https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RWZiQ B?culture=it-it&country=it

Sulla base di queste premesse, l'obiettivo di questo lavoro è lo sviluppo di un sistema di navigazione basato sulle GNN, con il fine di contribuire alla creazione di un nuovo modello per la navigazione interna e allo sviluppo di backend e frontend.

L'elaborato si compone di quattro capitoli. Il Capitolo 1 fornisce una panoramica generale sull'intelligenza artificiale (IA), esplorando la sua evoluzione storica e i concetti fondamentali. Vengono discusse le varie definizioni di IA, con un'attenzione particolare alle differenze tra intelligenza artificiale debole e forte. Successivamente, nel Capitolo 2 viene introdotto il progetto "NavigaUnivpm", descrivendone l'architettura e le funzionalità e analizzando il concetto di realtà aumentata applicata alla navigazione all'interno del campus universitario. Le fasi di progettazione, dall'analisi dei requisiti alla scelta delle tecnologie, vengono illustrate per offrire una visione complessiva del sistema. Nel Capitolo 3 l'attenzione si sposta poi sul problema del calcolo del percorso, affrontando le sfide legate alla navigazione su reti complesse e descrivendo l'implementazione delle Graph Neural Networks (GNN), ancora in fase di ricerca, e la loro applicazione nel progetto. Viene inoltre trattato l'addestramento del modello e la valutazione delle sue prestazioni, esaminando i grafi utilizzati e i risultati ottenuti in termini di "loss" e "accuracy", insieme all'analisi del feedback degli utenti per valutare l'usabilità del sistema. Infine, nel Capitolo 4, vengono discussi i risultati raggiunti e le prospettive di sviluppo future, con proposte di ottimizzazione e miglioramento del sistema, al fine di rendere l'applicazione ancora più efficace e personalizzata.

CAPITOLO 1

# Introduzione all'intelligenza artificiale

*In questo capitolo viene fornita una panoramica generale sull'intelligenza artificiale (IA), esplorando la sua evoluzione storica e i concetti fondamentali. Vengono discusse le varie definizioni dell'IA, con un focus sulle differenze tra intelligenza artificiale debole e forte. Infine, vengono introdotti i concetti di machine learning e deep learning, con particolare attenzione alle reti neurali, che rappresentano il fulcro delle moderne applicazioni di IA.*

## 1.1 Definizione dell'intelligenza artificiale

Dopo aver brevemente introdotto il concetto di intelligenza artificiale nell'introduzione, proviamo a darne una definizione: secondo il parlamento Europeo "L'intelligenza Artificiale (IA) è l'abilità di una macchina di dimostrare capacità umane quali il ragionamento, l'apprendimento, la pianificazione e la creatività"[1]. Attualmente, due sistemi di IA sono diventati famosi in tutto il mondo: ChatGPT e Gemini. Due modelli di linguaggio di grandi dimensioni (LLM), basati sull'architettura GPT (Generative Pretrained Transformer), sviluppati rispettivamente da Open AI e Google AI. Le definizioni di intelligenza artificiale fornite da questi due modelli sono:

- per ChatGPT, l'intelligenza artificiale è un campo interdisciplinare che mira a creare sistemi informatici in grado di eseguire compiti che richiedono tipicamente l'intelligenza umana. Questi compiti includono il riconoscimento di modelli, il ragionamento, il problem solving, la comprensione del linguaggio naturale e il controllo di robot. Gli approcci all'IA possono includere l'uso di algoritmi di apprendimento automatico, reti neurali artificiali, algoritmi di ricerca e ottimizzazione, tra gli altri. L'obiettivo principale dell'IA è sviluppare sistemi che possano imitare o migliorare le capacità umane in modo efficiente ed efficace in una varietà di contesti e applicazioni.
- per Gemini, l'intelligenza artificiale è un campo di studio che mira a creare sistemi informatici in grado di svolgere compiti che richiedono intelligenza umana, come la capacità di ragionare, imparare, risolvere problemi, adattarsi a nuove situazioni e prendere decisioni autonome. L'IA non è una tecnologia unica e monolitica, ma piuttosto un insieme di diverse tecnologie e approcci. L'IA è ancora in fase di sviluppo e ci sono molte sfide da affrontare prima che possa raggiungere il suo pieno potenziale.

[1] https://www.europarl.europa.eu/topics/it/article/20200827STO85804/che-cos-e-l-intelligenza-artificiale-e-come-viene-usata

> Tuttavia, l'IA ha già avuto un impatto significativo su diversi settori e ha il potenziale per rivoluzionare molti aspetti della nostra vita.

Si può notare che le definizioni date sono molto simili, infatti il "pensiero" di un'intelligenza artificiale deriva dai dati utilizzati per il suo allenamento, i quali potrebbero essere condivisi tra i diversi modelli. Esistono due diverse classificazioni di intelligenza artificiale: debole e forte. "La macchina potrà eguagliare e addirittura superare il ragionamento umano oppure non sarà mai equivalente a esso?" (Flowers [2019]). Proprio da questa domanda si sviluppano queste due classificazioni.

### 1.1.1 Intelligenza artificiale debole

L'intelligenza artificiale debole si riferisce ad un tipo di sistema intelligente che si occupa di svolgere compiti complessi in modo efficiente e preciso. L'obiettivo dell'IA debole non è quello di realizzare macchine che abbiano un'intelligenza umana, ma che siano in grado di agire con successo in alcune funzioni complesse umane. La macchina non è capace di pensare in maniera autonoma, svolge egregiamente il suo compito ma ha bisogno della presenza dell'uomo. Il suo compito è semplicemente quello di realizzare un'intelligenza "simulata".

### 1.1.2 Intelligenza artificiale forte

La teoria sull'intelligenza artificiale forte, o super IA, al contrario di quella debole, afferma che la macchina non sia solamente uno strumento ma diventa una vera e propria mente con una capacità cognitiva non distinguibile da quella umana. La tecnologia alla base dell'Intelligenza Artificiale forte è quella dei sistemi esperti, cioè una serie di programmi che vogliono riprodurre, attraverso una macchina, le prestazioni e le conoscenze delle persone esperte in un determinato campo. Il sistema esperto opera in 3 step distinti. Il primo riguarda la base di partenza, che consiste in regole e procedure di cui il sistema ha bisogno nel corso del suo operato. Il motore inferenziale è il secondo elemento e consiste nell'applicazione, in una determinata situazione, dei dati e delle nozioni. Infine, l'interfaccia utente ove si profila l'interazione tra la macchina e l'essere umano, il nucleo dell'AI forte.

### 1.1.3 The imitation game

Il matematico inglese Alan Turing nel 1950 stabilì che un algoritmo possa essere definito intelligente quando in grado di imitare in maniera indistinguibile il comportamento umano. Per verificare che la procedura sia intellegente, sviluppò un test di verifica che prende il nome di "The Imitation Game " (Turing *et al.* [1950]). Il gioco prevede 3 partecipanti: un giudice (umano), un uomo ed una macchina posti in ambienti separati. Il giudice dialoga attraverso messagi testuali con i 2 partecipanti, come ad esempio in un servizio di assistenza clienti, il cui obiettivo è scoprire chi tra i 2 partecipanti è la macchina. Le regole del gioco prevedono che il sistema artificiale possa ritardare la risposta alle domande o addirittura sbagliarle di proposito per non smascherare le sue reali capacità computazionali. Turing profetizzò che nell'arco di 50 anni la programmazione dei computer avrebbe raggiunto un livello tale che il 70% dei giudici non sarebbe stato in grado di riconoscere la macchina.

Questa metodologia è passata alla storia come "Test di Turing" e rappresenta ancora oggi il benchmark di riferimento per stabilire quando si parla di intelligenza artificiale o meno. La pubblicazione Turing *et al.* [1950] è considerata una pietra miliare di conoscenza e visione, sopratutto nel contesto in cui è stata sviluppata, solamente 2 anni dopo la nascita del primo computer a Manchester. Turing prospettò anche la possibilità di una macchina in grado di imparare, concetto che oggi ritroviamo nel Machine Learning e Deep Learning,

oramai fondamentali nel campo dell'intelligenza artificiale. Questo tipo di sistemi sono in grado di imparare dai dati, riconoscendo gli schemi (pattern recognition) ed estraendo conoscenza da utilizzare per modificare di conseguenza il proprio programma e quindi il proprio comportamento.

Secondo la teoria del machine learning, per addestrare un sistema intelligente abbiamo bisogno di informazioni. Più dati si hanno a disposizione, più l'algoritmo impara velocemente e il sistema diventa intelligente. Per questo motivo si parla di "Big Data" [2], set di dati più grandi e complessi provenienti da diverse origini. Questi grandi insiemi di dati sono caratterizzati da 3 dimensioni: volume, velocità e varietà. Il volume si riferisce alla quantità di dati generati e raccolti. In passato, i dati erano generati in quantità relativamente piccole da fonti come database aziendali e fogli di calcolo. Oggi, con l'avvento di Internet of Things (IoT), social media e sensori, la quantità di dati generati è cresciuta esponenzialmente. La velocità si riferisce alla rapidità con cui i dati vengono generati e acquisiti. In passato, i dati venivano generati e aggiornati a intervalli regolari, come ad esempio una volta al giorno o alla settimana. Oggi, con i social media e i sensori in tempo reale, i dati vengono generati e acquisiti in tempo reale. La varietà si riferisce alla diversità dei tipi di dati generati. In passato, i dati erano principalmente strutturati, come ad esempio dati di database o fogli di calcolo. Oggi, i dati possono essere strutturati, semi-strutturati e non strutturati, come ad esempio testo, immagini, video e audio.

I primi algoritmi di Machine Learning sono stati sviluppati nel mondo della finanza, uno dei pochi settori che dispone di molti dati in tempo reale, di informazioni sui prezzi, di un numero sostanzialmente infinito di strumenti finanziari: gli algoritmi possono quindi crescere e imparare velocemente.

### 1.1.4 Test di Turing ed IA

Nel passato sono stati sviluppati diversi sistemi per verificare se un utente sia un umano o una macchina. Un esempio è il CAPTCHA[3] (Completely Automated Public Turing test to tell Computers and Humans Apart), un tipo di misura di sicurezza nota come autenticazione Challenge/Response. Il test è concepito per verificare l'identità dell'utente come umana al fine di prevenire intrusioni e attività indesiderate da parte di bot[4] o tentativi di spam.

I principali fornitori di servizi online usano il captcha per mantenere sicure e protette le informazioni degli utenti. La prima versione di questo tipo di verifica consiste in due semplici parti: una sequenza alfanumerica, generata casualmente, visualizzata in un'immagine distorta e una casella di testo. Per superare la prova, dimostrando di essere umano, l'utente deve digitare nella casella di testo i caratteri raffigurati nell'immagine. Nella sua evoluzione la prova ha integrato elementi sempre più complessi quali: puzzle visivi, domande basate su conoscenze generali o addirittura interazioni dinamiche che richiedono una comprensione. Contemporaneamente, si è passati a metodi innovativi, quali le verifiche fondate su comportamenti umani, l'analisi cinetica del cursore del mouse o la reazione temporale a input visivi e acustici.

Oggi è diventato sempre più indispensabile essere in grado di riconoscere un umano da una macchina. In una ricerca condotta da un team di accademici sotto la guida del Prof. Matthew Jackson (Mei *et al.* [2024]) dell'Università di Stanford e successivamente pubblicata su Proceedings of the National Academy of Sciences (PNAS), si sostiene che ChatGPT-4, la più recente iterazione del modello di linguaggio sviluppato da OpenAI, abbia manifestato comportamenti che sono praticamente indistinguibili da quelli umani. Questo rappresenta un

[2] https://www.oracle.com/it/big-data/what-is-big-data

[3] https://support.google.com/a/answer/1217728?hl=it

[4] Un bot, abbreviazione di "robot", è un programma per computer progettato per imitare o sostituire le azioni di un essere umano eseguendo attività automatizzate e ripetitive.

avanzamento significativo, ovvero la prima volta in cui un chatbot di OpenAI supera un test di Turing personalizzato. In questo specifico studio, l'obiettivo primario del gruppo di ricerca non era tanto valutare l'accuratezza delle risposte fornite da ChatGPT-4, quanto piuttosto esaminare sistematicamente i tratti comportamentali ed etici del modello e confrontarli con quelli di un campione umano diversificato, composto da oltre 100.000 individui provenienti da 52 diverse nazioni. Per la valutazione di tali tratti comportamentali hanno adottato il modello OCEAN Big-5, un framework [5] psicologico che valuta cinque tratti fondamentali della personalità umana: apertura mentale, coscienziosità, estroversione, gradevolezza e nevroticismo. Mediante una serie di giochi comportamentali e test etici, i modelli GPT-3 e GPT-4 sono stati sottoposti a un'analisi dettagliata per determinare la loro sovrapponibilità con i tratti umani misurati dal modello Big-5.

I risultati emersi indicano che ChatGPT-4 presenta tratti comportamentali e di personalità che risultano statisticamente indistinguibili da quelli umani. In particolare, il modello ha dimostrato una notevole capacità di adattarsi a vari ruoli in contesti simulati e ha mostrato una predisposizione "innaturale" verso la cooperazione e l'altruismo, evitando comportamenti di natura conflittuale.

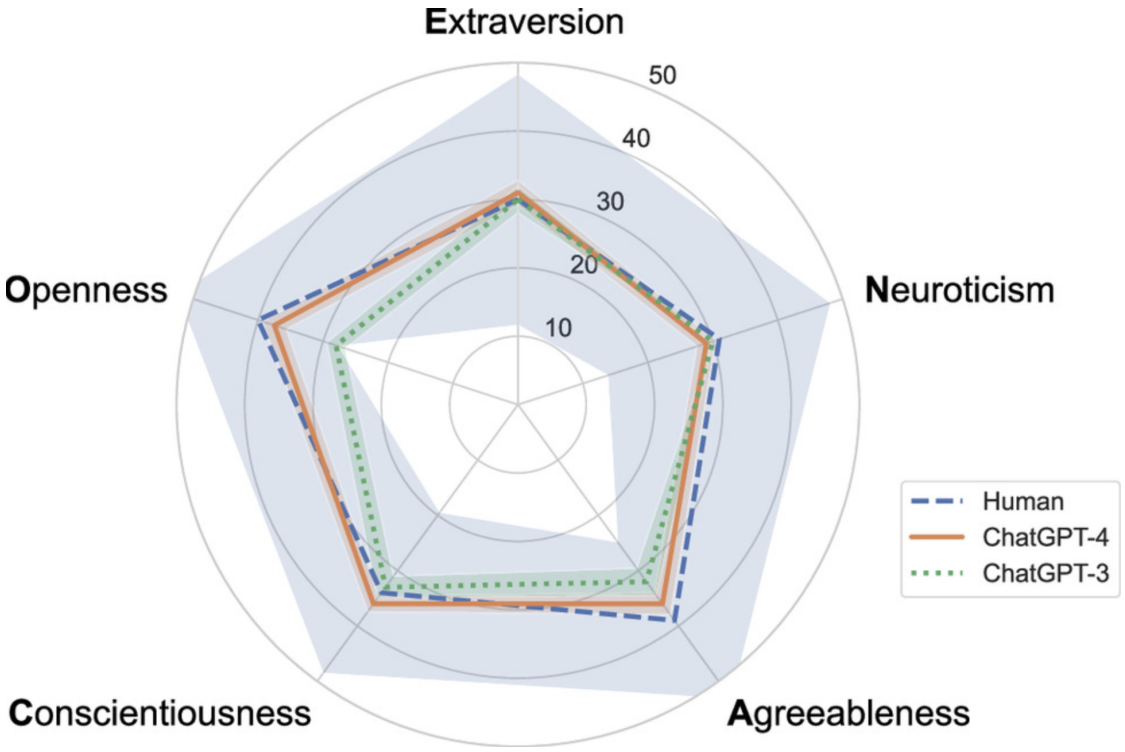


**Figura 1.1:** I tratti della personalità misurati da OCEAN Big–5 per GPT–3 e GPT–4, confrontati con la media tra gli intervistati umani. Credits: Mei *et al.* [2024].

## 1.2 Reti Neurali

### 1.2.1 Machine Learning e Deep Learning

Il Machine Learning è una disciplina dell'intelligenza artificiale che si concentra sullo sviluppo di algoritmi e modelli in grado di apprendere da dati e di effettuare previsioni o

[5] Un framework è un'architettura logica di supporto sulla quale un software può essere progettato e realizzato, spesso facilitandone lo sviluppo da parte del programmatore.

decisioni basate su di essi, senza la necessità di una programmazione esplicita. Nei paradigmi di programmazione tradizionali, i computer eseguono istruzioni predefinite per compiere determinate attività. Al contrario, negli algoritmi di machine learning, si utilizzano tecniche statistiche per addestrare il sistema a riconoscere schemi, relazioni e tendenze nei dati, permettendo così al sistema di prendere decisioni, effettuare previsioni o fornire raccomandazioni. Gli algoritmi di machine learning possono essere ampiamente classificati in quattro categorie principali:

- Apprendimento supervisionato, in cui gli algoritmi vengono addestrati su dati etichettati, dove l'input è associato all'output corretto. Questi algoritmi imparano a mappare gli input sugli output e a effettuare previsioni su nuovi dati.
- Apprendimento non supervisionato, in cui gli algoritmi apprendono da dati non etichettati, identificando schemi e relazioni intrinseche nei dati senza etichette di output specifiche. L'aggregazione e la riduzione della dimensionalità sono compiti tipici dell'apprendimento non supervisionato.
- Apprendimento semi-supervisionato, che combina elementi dell'apprendimento supervisionato e non supervisionato, utilizzando un piccolo set di dati etichettati insieme a un set più ampio di dati non etichettati per l'addestramento.
- Apprendimento per rinforzo, in cui gli algoritmi apprendono attraverso un meccanismo di tentativi ed errori, ricevendo feedback sotto forma di ricompense o penalità in base alle proprie azioni. L'obiettivo è massimizzare le ricompense nel tempo interagendo con un ambiente.

Al contratio del Machine Learning, il Deep Learning, si basa sul concetto di rete neurale. Il Machine Learning è una vasta disciplina dell'intelligenza artificiale che comprende vari algoritmi e tecniche progettate per permettere ai computer di apprendere da dati e di effettuare previsioni o decisioni senza la necessità di una programmazione esplicita. Le reti neurali modellano l'architettura e la funzionalità del cervello umano. Le reti neurali sono composte da nodi (neuroni) interconnessi organizzati in strati, dove ogni neurone elabora i dati di input e trasmette l'output allo strato successivo. Regolando i pesi e i bias delle connessioni tra i neuroni durante l'addestramento, le reti neurali possono apprendere schemi e relazioni complesse nei dati, rendendole strumenti potenti per compiti come il riconoscimento delle immagini, l'elaborazione del linguaggio naturale e il riconoscimento del parlato.

Le reti neurali hanno guadagnato notevole attenzione e popolarità negli ultimi anni, con lo sviluppo del Deep Learning, che prevede l'addestramento di reti neurali profonde con più strati nascosti per apprendere rappresentazioni gerarchiche dei dati.

### 1.2.2 Definizione di rete neurale

Le reti neurali rappresentano una classe di modelli computazionali ispirati al funzionamento del sistema nervoso biologico, in particolare al modo in cui i neuroni interagiscono tra loro attraverso sinapsi. Queste sono ampiamente utilizzati nel campo dell'intelligenza artificiale, in particolare del Deep Learning, offrendo un potente strumento per l'apprendimento automatico da dati grezzi.

A livello concettuale, una rete neurale è composta da un insieme di nodi, o neuroni artificiali, organizzati in strati o strutturati in una topologia specifica. Ogni neurone elabora le informazioni ricevute dai neuroni del layer precedente attraverso una funzione di attivazione, che può essere lineare o non lineare. Queste interazioni sono modellate attraverso un insieme di pesi sinaptici, che vengono aggiornati durante la fase di addestramento della rete.

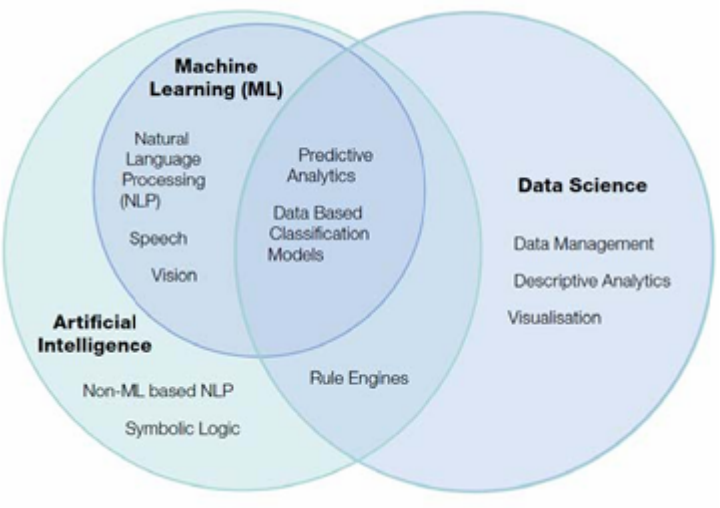


**Figura 1.2:** La relazione tra IA, Machine Learning and Data science. Credits: Szabadföldi [2021].

All'interno del panorama delle reti neurali, il Multilayer Perceptron (MLP) rappresenta una delle architetture più utilizzate. Il MLP è una rete neurale feedforward composta da almeno tre strati: uno strato di input, uno o più strati nascosti e uno strato di output. Ogni strato è costituito da un insieme di neuroni artificiali, ognuno dei quali è collegato a tutti i neuroni dello strato precedente e successivo attraverso pesi sinaptici. Il MLP può essere considerato come una estensione delle reti neurali feedforward standard, arricchite dalla presenza di strati nascosti che aumentano la capacità di apprendimento della rete. In questo contesto, il MLP rappresenta un potente strumento per modellare relazioni complesse e non lineari presenti nei dati, sfruttando l'abilità delle reti neurali di apprendere rappresentazioni gerarchiche e distribuite delle informazioni.

Negli anni '60 e '70, con l'avvento della teoria della retropropagazione (backpropagation), le reti neurali hanno iniziato a guadagnare popolarità come strumenti di apprendimento automatico. La retropropagazione, un algoritmo di apprendimento che permette di aggiornare i pesi delle connessioni in modo efficiente, ha rappresentato un passo significativo verso la creazione di reti neurali più profonde e complesse. Gli anni '80 e '90 hanno visto un interesse crescente per le reti neurali e l'emergere di tecniche più avanzate, come le reti neurali convoluzionali (CNN) e le reti neurali ricorrenti (RNN). Il nuovo millennio ha segnato una vera e propria rinascita delle reti neurali, alimentata dai rapidi progressi tecnologici, dall'incremento della potenza computazionale e dalla disponibilità di grandi quantità di dati. L'introduzione di nuove architetture come le reti neurali generative (GAN), le trasformer-based architectures e i modelli pre-addestrati ha ampliato notevolmente il campo di applicazione delle reti neurali, rendendole strumenti fondamentali in settori come il riconoscimento di immagini, l'elaborazione del linguaggio naturale e l'apprendimento rinforzato.

### 1.2.3 Reti neurali convoluzionali

Le reti neurali convoluzionali (Albawi *et al.* [2017]), o Convolutional Neural Networks (CNN) in inglese, sono una classe di reti neurali particolarmente efficaci per l'elaborazione di immagini e dati con una struttura spaziale. Le CNN sono state ispirate dal modo in cui il sistema visivo biologico elabora le informazioni visive attraverso la corteccia visiva e sono diventate lo standard de facto per molte applicazioni di visione artificiale. Analizzando le caratteristiche principali di una CNN:

- Questa tipologia di rete neurale utilizza strati di convoluzione per filtrare l'input attraverso una serie di maschere di convoluzione. Questi filtri possono catturare caratteristiche spaziali come bordi, texture e forme presenti nell'immagine.
- Gli strati di pooling vengono utilizzati per ridurre la dimensione spaziale dell'input, riducendo la complessità computazionale e rendendo la rete in grado di riconoscere pattern invarianti rispetto a piccole traslazioni o deformazioni dell'input.
- Le CNN sono organizzate in maniera gerarchica, con i primi strati che catturano caratteristiche basi come bordi e gradienti, mentre strati più profondi possono riconoscere caratteristiche complesse e astratte come parti di oggetti o interi oggetti.
- A differenza delle reti neurali fully connected, le CNN presentano una connettività locale e una pesatura condivisa attraverso i filtri di convoluzione, riducendo il numero di parametri e garantendo una maggiore capacità di generalizzazione.

Le reti neurali convoluzionali hanno rivoluzionato il campo della visione artificiale e sono state adottate in numerose applicazioni riguardanti:

- Riconoscimento di immagini, per classificare e identificare oggetti, persone, animali e scene all'interno di immagini.
- Segmentazione semantica, per assegnare etichette semantiche a ciascun pixel in un'immagine, distinguendo tra diverse categorie (es. strada, cielo, veicoli).
- Rilevazione di oggetti, per individuare e localizzare oggetti di interesse all'interno di immagini, spesso utilizzando bounding box o maschere di segmentazione.
- Generazione di immagini e trasferimento di stile, per generare nuove immagini realistiche o per applicare stili artistici a immagini esistenti.

Le CNN presentano ancora alcune limitazioni, come la necessità di grandi quantità di dati etichettati per l'addestramento e la difficoltà di gestire variazioni di scala, orientazione e illuminazione. Infatti la ricerca attuale è focalizzata sullo sviluppo di architetture più efficienti e leggere e sulla loro combinazione con altre tecniche di apprendimento profondo, come le reti neurali ricorrenti (RNN) e i trasformers.

### 1.2.4 Reti neurali ricorrenti

Le Reti Neurali Ricorrenti, o Recurrent Neural Networks (RNN) in inglese, costituiscono una classe di reti neurali profonde progettate per l'analisi e l'elaborazione di dati sequenziali, come sequenze temporali, linguaggio naturale e segnali audio. La peculiarità delle RNN risiede nella loro capacità intrinseca di mantenere una memoria delle informazioni passate durante l'elaborazione delle sequenze, consentendo di catturare dipendenze a lungo termine e strutture complesse nelle dati sequenziali. Le caratteristiche principali di una RNN sono:

- Stati nascosti e memoria temporale. Le RNN sono dotate di uno stato nascosto o di una cella di memoria che permette di memorizzare e aggiornare le informazioni sulle osservazioni precedenti durante l'elaborazione sequenziale, creando una memoria temporale che favorisce la cattura di dipendenze a lungo termine.
- Connessioni cicliche e struttura ricorsiva. A differenza delle reti neurali feedforward, le RNN presentano connessioni cicliche che permettono di elaborare sequenze di dati con una struttura temporale o spaziale intrinseca, eseguendo operazioni ricorsive sugli elementi della sequenza.

- Elaborazione sequenziale e dinamica temporale. Le RNN operano sequenzialmente, utilizzando l'informazione corrente e lo stato nascosto precedente per generare l'output corrente, rendendo possibile l'analisi e l'interpretazione di dati sequenziali complessi e dinamici.

Le reti neurali ricorrenti hanno trovato la loro applicazione in settori riguardanti:

- Elaborazione del linguaggio naturale, per la generazione di testo, la traduzione automatica, l'analisi semantica e l'elaborazione di sequenze di comandi e istruzioni.
- Riconoscimento vocale e trascrizione automatica, per il riconoscimento di parole e frasi, la trascrizione di dialoghi e la generazione di sintesi vocale.
- Apprendimento rinforzato e agenti intelligenti, per la creazione di sistemi di apprendimento adattivo e agenti intelligenti capaci di interagire e apprendere da ambienti complessi e dinamici.

### 1.2.5 Reti neurali per grafi

Le Reti Neurali per Grafi (Fan *et al.* [2019]), o Graph Neural Network (GNN) in inglese, rappresentano una classe emergente di modelli di apprendimento progettati specificamente per l'elaborazione di dati strutturati in forma di grafi. Le GNN estendono le capacità delle reti neurali tradizionali adattandosi alla natura complessa e dinamica dei grafi, consentendo l'analisi e l'interpretazione di reti complesse e interconnesse presenti in molteplici domini applicativi. Principalmente le reti neurali per grafi sono caratterizzate da:

- Elaborazione locale e connettività strutturale. Le GNN operano eseguendo un'elaborazione locale e iterativa dei nodi e delle loro vicinanze nel grafo, aggregando e combinando le informazioni attraverso le connessioni del grafo per generare rappresentazioni informative e contestualizzate dei nodi e delle strutture del grafo.
- Invarianza ed equivarianza strutturale. Le GNN sono progettate per essere invarianti o equivarianti rispetto alle permutazioni dei nodi, garantendo che la loro rappresentazione sia indipendente dall'ordinamento o dall'etichettatura dei nodi nel grafo.
- Architetture scalabili e adattive. Esistono diverse architetture di GNN, tra cui Graph Convolutional Networks (GCN), Graph Attention Networks (GAT) e GraphSAGE, che offrono una flessibilità e una scalabilità notevoli per modellare grafi di diversa dimensione, complessità e topologia.

Le GNN stanno rivoluzionando l'applicazione dell'apprendimento su grafi nei seguenti settori:

- Nel settore della Computer Vision. Le CNN tradizionali hanno permesso alle macchine di distinguere e identificare oggetti in immagini e video. Le GNN offrono potenziali miglioramenti in questo campo, specialmente per la generazione di grafi di scena. Questi modelli analizzano un'immagine per creare un grafo semantico che rappresenta gli oggetti e le loro relazioni semantiche. L'uso delle GNN è in continua crescita, con applicazioni che includono l'analisi delle interazioni tra umano e oggetto e la classificazione di immagini.
- Nel settore dell'elaborazione del linguaggio naturale (NLP). Sebbene i dati testuali siano tradizionalmente gestiti da modelli sequenziali come RNN o LSTM, l'uso di grafi sta

guadagnando popolarità grazie alla loro capacità di rappresentare relazioni complesse. Le GNN possono essere utilizzate per vari compiti di NLP, dalla classificazione del testo all'estrazione delle relazioni, passando per la traduzione automatica e la risposta alle domande. Tecniche avanzate come GraphSage permettono di adattare questi modelli anche a nodi non visti precedentemente.

- Nel settore della gestione del traffico del traffico e della mobilità urbana. Le GNN sono utilizzate per prevedere la velocità del traffico, il volume e la densità delle strade. Attraverso l'uso di grafi spazio-temporali, le GNN possono modellare la rete stradale come un sistema dinamico in cui i sensori sulle strade rappresentano i nodi e le distanze tra di loro definiscono gli archi. Questa rappresentazione consente di elaborare previsioni accurate sulla base delle dinamiche del traffico osservate.

- Nel campo della chimica. Le GNN sono strumenti preziosi per la modellazione e lo studio della struttura dei composti molecolari. Questi modelli rappresentano gli atomi come nodi e i legami chimici come archi, permettendo ai chimici di analizzare e comprendere le proprietà e le caratteristiche dei composti in modo più efficiente e accurato.

- Nel settore delle Social Network. Le GNN giocano un ruolo cruciale, in quanto possono sfruttare le informazioni ricche e complesse presenti nelle reti sociali, per potenziare i sistemi di raccomandazione. Questi modelli possono migliorare l'engagement degli utenti e fornire raccomandazioni più personalizzate basate su interazioni sociali e preferenze, sfruttando la capacità delle GNN di integrare informazioni da diversi nodi e archi all'interno di un grafo.

Le applicazioni delle GNN non si limitano ai settori sopra menzionati, poiché la versatilità di queste reti consente di applicarle a una vasta gamma di problemi in diversi domini, come la verifica e il ragionamento sui programmi. La capacità delle GNN di modellare relazioni complesse e strutture dinamiche le rende adatte per affrontare sfide in vari campi, sfruttando la loro potenza computazionale e la capacità di apprendere da dati strutturati.

## 1.3 Applicazioni dell'intelligenza artificiale

Nel contesto dell'innovazione tecnologica, l'intelligenza artificiale (IA) è ormai una disciplina chiave, con il potenziale di trasformare radicalmente una miriade di settori e discipline. I progressi compiuti nell'apprendimento automatico, nelle reti neurali e nelle tecniche di elaborazione dati delineano nuovi orizzonti applicativi, che spaziano dalla diagnosi medica alla robotica, dalla finanza alla progettazione di nuovi farmaci.

### 1.3.1 Ambito finanziario

L'intelligenza artificiale sta rivoluzionando il panorama finanziario attraverso una serie di applicazioni trasversali. In primo luogo, l'IA viene impiegata con crescente frequenza per potenziare la rilevazione delle frodi, l'ottimizzazione dei portafogli, la gestione del rischio e il servizio clienti. Gli avanzati algoritmi di IA possono esaminare vasti dataset per identificare schemi sospetti, affinare le strategie di investimento e offrire consigli finanziari altamente personalizzati (Buchanan [2019]).

Parallelamente, l'automazione delle funzioni finanziarie sta guadagnando terreno, con soluzioni alimentate da sistemi intelligenti che automatizzano operazioni come il processing dei pagamenti, l'interazione con i clienti attraverso chatbot e la gestione degli investimenti

tramite robo-consulenti. Queste innovazioni non solo potenziano l'efficienza operativa e riducono i costi, ma arricchiscono anche l'esperienza del cliente attraverso servizi più focalizzati e proattivi. Infine, l'IA si rivela un potente alleato nella gestione del rischio, permettendo alle istituzioni finanziarie di analizzare grandi volumi di dati per identificare e mitigare potenziali frodi, rischi di credito e volatilità di mercato, contribuendo così a rafforzare la stabilità finanziaria e a informare decisioni più prudenti e basate su dati.

### 1.3.2 Ambito militare

L'intelligenza artificiale sta assumendo un ruolo preminente nel contesto militare, rivoluzionando e ridefinendo le tecnologie militari avanzate e promuovendo un approccio innovativo alla difesa. Le soluzioni basate sull'IA si distinguono innanzitutto per la loro capacità di integrazione e analisi, sfruttando metodologie analitiche avanzate per l'interpretazione e l'elaborazione dei dati. Queste soluzioni sono progettate per operare in un ecosistema interconnesso, che comprende reti sia virtuali che fisiche, coinvolgendo una varietà di attori come sensori, organizzazioni, individui e agenti autonomi.

Il Dipartimento della Difesa degli Stati Uniti (DoD) ha identificato l'IA come una priorità strategica (Szabadföldi [2021]), allocando significativi investimenti finanziari. Nel budget del 2020, quasi un miliardo di dollari è stato destinato a iniziative relative all'IA, con una particolare enfasi sul potenziamento delle capacità operative attraverso l'integrazione di sistemi robotici guidati dall'IA e sull'ottimizzazione della precisione delle operazioni militari.

Le implicazioni dell'IA nel dominio militare vanno ben oltre l'ambito operativo diretto, influenzando profondamente settori critici come armamenti nucleari, guerra cibernetica, tecnologie dei materiali avanzati e biotecnologie. Tali sviluppi tecnologici sono suscettibili di avere un impatto trasformativo sull'ordine mondiale, analogamente all'introduzione delle armi nucleari, alimentando una potenziale corsa agli armamenti basata sull'IA.

Nel contesto delle operazioni C4ISR (Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance), i sistemi intelligenti stanno giocando un ruolo centrale nell'ottimizzazione delle capacità operative. L'implementazione di sistemi autonomi intelligenti consentono l'automazione di compiti complessi e rischiosi, migliorando la qualità e l'efficienza delle analisi dell'intelligence e dei processi decisionali.

Parallelamente, la NATO ha lanciato il progetto "Military Uses of Artificial Intelligence, Automation, and Robotics" (MUAAR) nel 2020 [6], nell'ambito della sua Campagna Multinazionale di Sviluppo delle Capacità (MCDC). Questo progetto mira a sviluppare nuovi concetti e capacità per affrontare le sfide emergenti legate all'impiego congiunto di sistemi autonomi e intelligenti nelle operazioni militari.

In conclusione, la sinergia tra l'IA e le tecnologie avanzate di analisi dei Big Data (Big Data Advanced Analytics, BDAA) sta ridefinendo le strategie e le capacità militari, con l'IA che si avvale dei dati forniti dalle piattaforme BDAA per sviluppare soluzioni sempre più sofisticate e adattive alle esigenze operative contemporanee.

### 1.3.3 Ambito sanitario

L'introduzione dell'intelligenza artificiale in ambito medico richiede molta attenzione, poiché si vanno a trattare dati sensibili dei pazienti. Per questo è fondamentale che i sistemi di IA (Rajpurkar *et al.* [2022]) siano affidabili ed espliciti ed, in seconda istanza, anche facili da utilizzare ed integrare nei flussi di lavoro clinici. L'esplicabilità e la trasparenza sono elementi cruciali per instaurare la fiducia nei sistemi di intelligenza artificiale applicati alla medicina. La possibilità di comprendere come il sistema ha elaborato le sue conclusioni favorisce

[6] $https://www.act.nato.int/wp-content/uploads/2023/05/2020_mcdc-muaar.pdf$

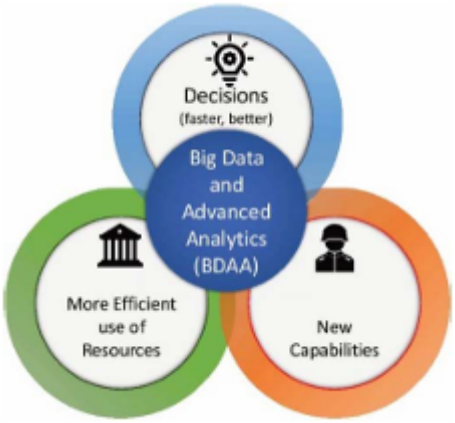


**Figura 1.3:** Obiettivi del BDAA. (Fonte: Nato science & technology organization, 2020)

l'accettazione e la validazione delle predizioni formulate. Al momento tale funzionalità non è supportata dagli attuali modelli, infatti operano come "scatole nere", rendendo difficile interpretare i suoi processi decisionali. Oltre all'accuratezza, per valutare i modelli, vengono considerate anche la robustezza e la generalizzabilità fra i vari contesti clinici e popolazioni di pazienti, assicurandosi che i sistemi proteggano la privacy di essi. Poiché la corretta elaborazione dell'input da parte del sistema può dipendere pesantemente da come esso viene fornito, è richiesta una formazione adeguata agli utenti umani del modello che dovranno anche interpretare l'output fornito dall'elaboratore.

Sorgono specifiche sfide regolatorie dall'apprendimento continuo, dove i modelli apprendono da nuovi dati nel tempo e si adattano ai cambiamenti nelle popolazioni dei pazienti, nella raccolta dei dati e nella gestione della cura. Tradizionalmente, i regolatori dei sistemi di intelligenza artificiale adottano un approccio che prevede l'approvazione di un set di parametri fissi per ciascun modello. Tuttavia, tale approccio non è adeguato a gestire la dinamicità dei dati medici, che possono evolversi a causa di cambiamenti nelle popolazioni dei pazienti, nelle metodologie di raccolta dei dati e nelle strategie di gestione della cura. Di conseguenza, si sta riconoscendo la necessità di sviluppare processi di certificazione più flessibili e adattabili che consentano l'aggiornamento e l'ottimizzazione dei modelli di IA in risposta ai cambiamenti dei dati e alle nuove informazioni disponibili. Di recente, la FDA ha proposto un framework per i sistemi AI adattivi in cui approverebbero non solo un modello iniziale ma anche un processo per aggiornarlo nel tempo.

Esistono preoccupazioni che gli attori malintenzionati interessati al furto di identità e ad altri comportamenti illeciti possano approfittare dei dataset medici, che spesso contengono grandi quantità di informazioni sensibili sui pazienti reali. La decentralizzazione della memorizzazione dei dati è un modo per ridurre il danno potenziale di qualsiasi hack o fuga di dati. Il processo di apprendimento federato facilita tale decentralizzazione, rendendo anche più facile la collaborazione tra istituzioni senza complicati accordi di condivisione dei dati. I dati dei pazienti possono essere meglio protetti da tali attacchi se gli input sono crittografati prima dell'addestramento, ma questo approccio comporta il costo della interpretabilità del modello.

Recenti ricerche hanno sfruttato la disponibilità di ampi dataset di testi medici per compiti di elaborazione del linguaggio naturale correlati alla salute, approfittando di avanzamenti tecnici come i trasformatori e le word embeddings contestuali. Un esempio è BioBERT, un modello addestrato su un ampio corpus di testi medici che ha superato le prestazioni precedenti nello stato dell'arte su compiti di linguaggio naturale, come rispondere a domande biomediche. Tali modelli sono stati utilizzati per migliorare le prestazioni su compiti come

l'apprendimento dalla letteratura biomedica su quali farmaci sono noti per interagire tra loro o l'etichettatura automatica dei referti radiologici. Grandi dataset di testo sono stati anche estratti dai social media e utilizzati per tracciare tendenze di salute mentale su larga scala. I metodi di machine learning sono stati utilizzati per prevedere gli esiti dai dati dei segnali medici, come EEG, elettrocardiogramma e dati audio. Ad esempio, il machine learning applicato ai segnali EEG di pazienti clinicamente non responsivi con lesioni cerebrali ha permesso la rilevazione dell'attività cerebrale, un predittore della ripresa futura. Inoltre, la capacità dell'IA di trasformare direttamente le onde cerebrali in parole o testo ha un notevole valore potenziale per i pazienti con afasia o sindrome di locked-in che hanno avuto ictus. I dati dei segnali medici possono anche essere raccolti passivamente al di fuori di un ambiente clinico nel mondo reale utilizzando sensori indossabili come smartwatch che consentono il monitoraggio remoto della salute.

Alcuni modelli di Deep Learning integrano diverse fonti di dati medici per un approccio multimodale. Ad esempio, un modello per la diagnosi di disturbi respiratori ha preso come input registrazioni audio della tosse dei pazienti e i rapporti dei loro sintomi. I modelli multimodali hanno anche sfruttato input molto più complessi, come i fascicoli sanitari elettronici, che comprendono una vasta gamma di dati come diagnosi mediche, segni vitali, prescrizioni e risultati di laboratorio. Il campo dell'AI medica ha fatto notevoli progressi verso la distribuzione su larga scala, specialmente attraverso studi prospettici come gli RCT[7] e attraverso l'analisi delle immagini mediche. Tuttavia, l'IA medica rimane in una fase iniziale di convalida e implementazione. Nonostante il potenziale del campo, restano importanti questioni tecniche ed etiche.

### 1.3.4 Ambito ambientale

L'ambiente naturale sta subendo una crescente pressione a causa dei cambiamenti climatici, dell'inquinamento e della perdita di biodiversità. In questo contesto, l'Intelligenza Artificiale sta emergendo come una potente tecnologia per affrontare queste sfide. Tuttavia, l'applicazione dell'IA nell'ambiente presenta una serie di sfide etiche, tecniche e regolatorie che devono essere affrontate per garantire un uso efficace e responsabile. Recenti avanzamenti tecnologici hanno permesso lo sviluppo di sistemi IA avanzati che possono analizzare grandi quantità di dati ambientali, prevedere cambiamenti climatici, ottimizzare l'uso delle risorse naturali e contribuire alla conservazione della biodiversità.

L'IA può essere utilizzata per analizzare e interpretare i dati ambientali provenienti da sensori remoti, satelliti e stazioni di monitoraggio. Questi sistemi possono rilevare cambiamenti nelle condizioni ambientali, come la deforestazione, l'inquinamento dell'acqua e dell'aria e i cambiamenti climatici. Gli algoritmi intelligenti possono essere utilizzati per ottimizzare l'uso delle risorse naturali, come l'acqua e l'energia, riducendo gli sprechi e promuovendo pratiche sostenibili. L'intelligenza artificiale supporta gli sforzi di conservazione della biodiversità attraverso il monitoraggio e l'analisi delle popolazioni di specie in pericolo, l'identificazione delle aree prioritarie per la conservazione e lo sviluppo di strategie di gestione sostenibile. L'intelligenza artificiale (IA) e l'internet of things (IoT) rivestono un ruolo cruciale nell'evoluzione dell'edilizia sostenibile e nel promuovere il risparmio energetico nelle abitazioni. Mediante l'integrazione di sistemi intelligenti con dispositivi IoT, è possibile monitorare e regolare l'uso dell'energia in tempo reale, ottimizzando parametri come illuminazione, riscaldamento e raffreddamento. Ciò consente di ridurre significativamente il consumo energetico e le emissioni di gas serra, sottolineando l'importanza di una gestione energetica proattiva e personalizzata. Dal punto di vista progettuale, l'IA contribuisce all'elaborazione di solu-

[7] Gli studi clinici controllati randomizzati (randomized controlled trial, RCT) sono studi sperimentali che permettono di valutare l'efficacia di uno specifico trattamento in una determinata popolazione.

zioni edilizie altamente efficienti dal punto di vista energetico. Attraverso l'applicazione di algoritmi di ottimizzazione avanzati, gli algoritmi intelligenti guidano la configurazione degli spazi, l'isolamento termico e la selezione di materiali eco-compatibili. Questa sinergia tra intelligenza artificiale e IoT non solo promuove la realizzazione di edifici in linea con gli standard di edilizia sostenibile, ma facilita anche una gestione dinamica e adattiva delle esigenze energetiche degli occupanti.

In conclusione, l'integrazione dell'IA e dell'IoT rappresenta un avvenimento significativo nell'ambito dell'edilizia intelligente e sostenibile, favorendo un utilizzo ottimale delle risorse e un sensibile ridimensionamento dell'impatto ambientale associato alle abitazioni, massimizzando i benefici e minimizzando i rischi e gli impatti negativi.

### 1.3.5 Ambito automotive

L'Intelligenza Artificiale (IA) si sta imponendo come elemento catalizzatore della trasformazione del settore automobilistico, permeando ogni aspetto del processo di ideazione, progettazione e fruizione del veicolo. L'integrazione dell'IA apre le porte a un futuro di mobilità caratterizzato da una concomitanza di incrementi in termini di sicurezza, efficienza e personalizzazione.

Al cuore della rivoluzione automobilistica guidata dall'IA si collocano i sistemi di guida assistita e autonoma. Attraverso l'impiego di sensori, telecamere e algoritmi di apprendimento automatico, i veicoli moderni sono dotati di funzionalità avanzate che supportano il conducente in svariate situazioni di guida, riducendo il suo carico di lavoro e lo stress. Esempi tangibili di questa evoluzione includono il controllo automatico della velocità e il mantenimento della corsia, elementi che garantiscono una guida più sicura e confortevole in autostrada o in condizioni di traffico intenso. Inoltre, l'IA permette ai veicoli di assumere il controllo completo della guida in determinate circostanze, come incolonnamenti o su strade ben definite, aprendo la strada verso la guida autonoma di Livello 3, 4 e 5, con la prospettiva di un futuro in cui la guida umana potrebbe non essere più necessaria.

L'IA non si limita a migliorare l'esperienza di guida, ma ottimizza anche il viaggio nel suo complesso. I sistemi di navigazione intelligenti analizzano in tempo reale i dati sul traffico, le condizioni stradali e le previsioni meteo per fornire percorsi ottimizzati e previsioni di arrivo precise, evitando code e strade congestionate. Questo si traduce in viaggi più efficienti, veloci e meno stressanti per i conducenti. Al fine di garantire la massima sicurezza, l'IA dota i veicoli di capacità di percezione avanzate. Utilizzando tecniche di visione artificiale e sensori ad alta risoluzione, i sistemi intelligenti riconoscono e interpretano ostacoli sulla strada, come veicoli, pedoni, ciclisti e segnaletica, permettendo al veicolo di reagire prontamente per evitare collisioni e incidenti.

L'IA non solo migliora l'esperienza di guida, ma contribuisce anche a mantenere il veicolo in perfetta efficienza. L'analisi dei dati provenienti dai sensori e dalle prestazioni del veicolo permette di diagnosticare precocemente potenziali guasti o problemi meccanici, consentendo interventi tempestivi e ottimizzando i programmi di manutenzione. Ciò si traduce in una riduzione dei tempi di fermo del veicolo e un prolungamento della sua vita utile. L'IA trasforma l'auto in un ambiente personalizzato, adattandosi alle esigenze e alle preferenze del conducente. Sistemi di infotainment intelligenti offrono esperienze di intrattenimento su misura, navigazione basata sull'apprendimento automatico e comandi vocali intuitivi per un controllo interattivo e senza distrazioni. Inoltre, sensori e telecamere monitorano lo stato di allerta del conducente, rilevando segni di affaticamento o distrazione. In caso di pericolo, il sistema può fornire avvisi tempestivi o addirittura assumere il controllo del veicolo per evitare incidenti.

L'Intelligenza Artificiale sta rivoluzionando l'industria automobilistica a passi da gigante. I veicoli di oggi e di domani saranno sempre più sicuri, efficienti, connessi e personalizzati,

offrendo un'esperienza di guida inedita e anticipando il futuro della mobilità su strada. L'IA rappresenta un cambio di paradigma nel settore, ponendo le basi per un ecosistema automobilistico più intelligente, sicuro e sostenibile, contribuendo a ridurre l'impatto ambientale del trasporto automobilistico.

### 1.3.6 Ambito agricoltura

L'intelligenza artificiale gioca un ruolo cruciale nel migliorare le pratiche di gestione del suolo in agricoltura, sfruttando tecnologie avanzate per ottimizzare la salute e la produttività del suolo. Eli-Chukwu [2019], hanno condotto un'analisi complessiva di diverse ricerche effettuate e hanno categorizzato i miglioramenti apportati alla gestione del suolo:

- Insight Predittivi, i sistemi di IA analizzano varie fonti di dati sul suolo come temperatura, condizioni meteorologiche, livelli di umidità e performance storiche delle colture. Elaborando questi dati, il modello può fornire insight predittivi sui migliori sementi da piantare in una specifica area, tempi ottimali per seminare e raccogliere, e requisiti di nutrienti del suolo. Ciò aiuta gli agricoltori a prendere decisioni per migliorare i rendimenti delle colture.

- Monitoraggio della Salute del Suolo, dove sensori e telecamere integrati con l'IA possono essere dispiegati nei campi per monitorare in tempo reale le condizioni del terreno. Questi dispositivi raccolgono dati su umidità del suolo, livelli di nutrienti, pH e altri parametri critici. Monitorando continuamente la salute di esso, i sistemi intelligenti possono rilevare tempestivamente problemi come carenze di nutrienti o degrado, consentendo interventi tempestivi.

- Agricoltura di Precisione, l'intelligenza artificiale apre a nuove tecniche di agricoltura di precisione creando dettagliate mappe del suolo e analisi della variabilità. Identificando le variazioni all'interno dei campi, il modello aiuta gli agricoltori ad applicare fertilizzanti, irrigazione e altri input precisamente dove sono necessari. Questo approccio mirato minimizza gli sprechi, riduce l'impatto ambientale e ottimizza l'utilizzo delle risorse.

- Sistemi di Supporto Decisionale, assistono gli agricoltori nella presa di decisioni basate sui dati relative alla gestione del terreno. Questi sistemi analizzano complessi set di dati per raccomandare strategie ottimali di trattamento, rotazioni colturali e pratiche di conservazione.

- Miglioramento della qualità del suolo, le tecnologie intelligenti possono suggerire materiali organici adatti, compost e applicazioni di letame per migliorarne la qualità. Mediante la raccomandazione di interventi adeguati derivati dall'analisi e dai dati storici, l'IA contribuisce a potenziare la struttura, la fertilità e la salubrità del terreno.

I sistemi intelligenti giocano un ruolo vitale nella gestione delle malattie delle piante in agricoltura, sfruttando conoscenze specializzate e algoritmi per diagnosticare, prevenire e trattare efficacemente le malattie delle colture. Ecco come i modelli di intelligenza artificiale possono essere impiegati per potenziare l'efficienza nella gestione delle malattie:

- Rilevamento Precoce delle Malattie, utilizzando l'IA si possono analizzare sintomi, condizioni ambientali e dati storici per identificare potenziali malattie in una fase precoce. Riconoscendo schemi di malattia e fattori di rischio, questi sistemi aiutano gli agricoltori ad adottare misure proattive per prevenire epidemie e minimizzare le perdite delle colture.

- Assistenza Diagnostica, gli algoritmi intelligenti forniscono assistenza diagnostica abbinando i sintomi osservati con un database di malattie e patogeni conosciuti. Utilizzando motori di inferenza basati su regole e logica fuzzy[8], questi sistemi possono diagnosticare accuratamente le malattie e raccomandare appropriate strategie di trattamento.

- Raccomandazioni di Trattamento, offrono raccomandazioni di trattamento personalizzate basate sul tipo di malattia, varietà di coltura e fattori ambientali. Considerando molteplici variabili e vincoli, questi sistemi suggeriscono le misure di controllo più efficaci, come interventi chimici, biologici o fisici.

- Gestione Integrata dei Parassiti, supportano le strategie di gestione integrata dei parassiti considerando molteplici fattori che influenzano la loro dinamica. Integrando dati sulle popolazioni di parassiti, condizioni meteorologiche, fenologia delle colture e metodi di controllo, questi sistemi aiutano gli agricoltori a implementare approcci olistici alla prevenzione e al controllo delle malattie.

- Apprendimento Continuo e Miglioramento, i sistemi possono essere progettati per apprendere dai nuovi input di dati e dai feedback degli utenti, migliorando continuamente la loro accuratezza diagnostica e le raccomandazioni di trattamento.

### 1.3.7 Previsioni sul mercato

L'Intelligenza Artificiale rappresenta una delle tecnologie emergenti più promettenti, destinata a rivoluzionare diversi settori industriali nel prossimo futuro. Secondo le previsioni di Gartner, illustrate nella figura 1.4, il mercato globale del software per l'IA è previsto crescere del 17% nel 2023 fino ad arrivare al 20% nel 2026, sottolineando l'importanza e l'adozione sempre più diffusa di questa tecnologia (Gartner, 2022). Le innovazioni tecnologiche, in particolare nell'apprendimento profondo, stanno alimentando questa crescita, aprendo nuove prospettive e possibilità in settori come l'agricoltura, la sanità, l'automotive e molti altri (Goodfellow *et al.* [2016]).

Tuttavia, l'adozione su larga scala dell'IA porta con sé una serie di sfide, inclusi i potenziali impatti sul mercato del lavoro e la necessità di regolamentazioni etiche e normative. Secondo uno studio condotto da Bessen (Bessen [2018]), l'integrazione dell'IA nel mercato del lavoro richiede una revisione delle competenze e delle qualifiche necessarie, ma offre anche opportunità significative di innovazione e crescita economica. A tale riguardo, la Commissione Europea ha delineato linee guida etiche per un'IA affidabile, stabilendo principi fondamentali per un uso responsabile e etico della tecnologia (European Commission, 2020[9]).

In conclusione, l'IA sta rivoluzionando l'approccio alle tecnologie in vari settori, offrendo soluzioni innovative per migliorare la produttività, l'efficienza e la sostenibilità. Nonostante le sfide e le preoccupazioni, l'IA continua a mostrare un potenziale significativo per guidare la trasformazione digitale e la crescita economica nei prossimi anni, richiedendo un impegno collettivo per sfruttare appieno le sue potenzialità.

[8] La logica fuzzy è una logica in cui si può attribuire a ciascuna proposizione un grado di verità diverso da 0 e 1 e compreso tra di loro. È una logica polivalente, ossia un'estensione della logica booleana.

[9] *https://commission.europa.eu/document/download/d2ec4039-c5be-423a-81ef-b9e44e79825b_en?filename=commission-white-paper-artificial-intelligence-feb2020_en.pdf*

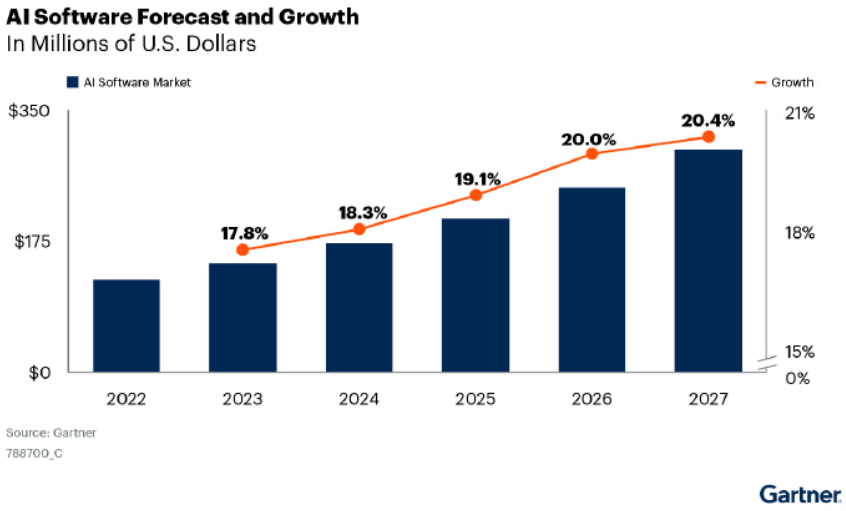


**Figura 1.4:** Previsioni e crescita dei software di Intelligenza Artificiale. (Fonte: Gartner)

# CAPITOLO 2

# NavigaUnivpm

*In questo capitolo viene analizzato il progetto NavigaUnivpm, descrivendone l'idea, i requisiti e l'architettura del sistema. Viene approfondito il concetto di realtà aumentata applicata alla navigazione all'interno del campus universitario e descritte le fasi di progettazione dell'applicazione, dall'analisi dei requisiti alla scelta delle tecnologie impiegate.*

## 2.1 Introduzione ed idea dell'applicazione

Nell'ambito della crescente digitalizzazione dei servizi universitari, l'implementazione di soluzioni tecnologiche mirate a migliorare l'esperienza degli studenti e del personale accademico diventa sempre più rilevante. Tra le sfide più comuni affrontate dagli utenti all'interno del campus universitario, la navigazione efficace tra i vari edifici, aule e servizi può risultare particolarmente complessa. Per risolvere questo problema e fornire un supporto pratico, è stata sviluppata un'applicazione di navigazione dedicata, progettata per guidare gli utenti attraverso il campus in modo intuitivo e efficiente. Questa soluzione, basata su tecnologie digitali avanzate come la realtà aumentata e l'interfaccia utente intuitiva, si propone di semplificare il percorso degli utenti all'interno dell'università, migliorando così la loro esperienza complessiva e promuovendo una maggiore fruibilità dei servizi offerti.

## 2.2 Realtà Aumentata

La realtà aumentata, in inglese Augmented Reality (AR), rappresenta una frontiera tecnologica rivoluzionaria che fonde elementi digitali con il mondo reale, creando un'esperienza interattiva ed immersiva per gli utenti. Attraverso dispositivi come smartphone, tablet, visori AR e occhiali intelligenti, la realtà aumentata permette di visualizzare e interagire con oggetti virtuali direttamente nel proprio ambiente reale.

Questa sovrapposizione di contenuti digitali arricchisce la percezione sensoriale dell'utente, aprendo nuove frontiere di interazione e apprendimento. Non si tratta di una semplice sovrapposizione di immagini, ma di un'integrazione intelligente che contestualizza gli elementi virtuali all'interno dell'ambiente reale, rendendoli parte integrante della scena. L'utente non è più un semplice spettatore, ma diventa protagonista attivo di un'esperienza ibrida che integra il reale con il virtuale.

Le tecnologie alla base della realtà aumentata variano a seconda dell'applicazione e del dispositivo utilizzato. Tuttavia, alcuni elementi chiave sono sempre presenti:

- Riconoscimento di immagini e oggetti. Il dispositivo AR identifica e riconosce gli oggetti nel mondo reale, permettendo agli elementi virtuali di essere posizionati e ancorati in modo preciso. Questa tecnologia sfrutta algoritmi di intelligenza artificiale e computer vision per analizzare le immagini acquisite dalla fotocamera del dispositivo e identificare i pattern distintivi di oggetti, loghi, opere d'arte o persino punti di riferimento naturali.

- Tracciamento del movimento. Il dispositivo monitora i movimenti dell'utente, adattando la visualizzazione degli oggetti virtuali di conseguenza, creando un'esperienza fluida e realistica. Questa tecnologia si basa su sensori come giroscopi, accelerometri e magnetometri per determinare la posizione e l'orientamento del dispositivo nello spazio. In questo modo, gli oggetti virtuali si muovono e si adattano in base ai movimenti dell'utente, mantenendo una coerenza spaziale e una prospettiva realistica.

- Visualizzazione in tempo reale e grafica computerizzata. Queste tecnologie rendono possibile la visualizzazione degli elementi virtuali in modo realistico e la loro integrazione convincente nell'ambiente reale. La grafica computerizzata 3D crea modelli virtuali dettagliati e realistici, mentre la visualizzazione in tempo reale permette di sovrapporre questi modelli al mondo reale in modo fluido e senza ritardi. L'illuminazione e le ombre virtuali vengono calcolate in base all'ambiente reale, creando un'integrazione impeccabile e un'esperienza immersiva.

L'avvento degli smartphone ha reso l'AR accessibile a un pubblico più ampio che mai, aprendo le porte a un panorama di applicazioni innovative e coinvolgenti. Le app mobili che integrano la realtà aumentata stanno rivoluzionando diversi settori, offrendo esperienze utente uniche e interattive. Ecco alcuni esempi:

- Gaming e intrattenimento:
  - Giochi di ruolo in realtà aumentata. Si può immaginare di esplorare la propria città come protagonisti di un'avventura fantasy, combattendo creature virtuali che appaiono nelle strade o risolvendo enigmi nascosti nei luoghi reali.
  - Giochi di strategia in AR. È possibile comandare eserciti virtuali che si scontrano su un tavolo da pranzo o pianificare tattiche di guerra utilizzando il giardino come campo di battaglia.
  - Filtri AR divertenti. Si possono trasformare in personaggi fantastici, applicare maschere divertenti o aggiungere effetti speciali ai selfie e video realizzati.

- Shopping e vendite al dettaglio:
  - Visualizzazione virtuale di prodotti. È possibile provare virtualmente vestiti, scarpe o accessori prima di acquistarli, visualizzandoli sovrapposti all'immagine in tempo reale.
  - Arredamento virtuale. Si ha la possibilità di posizionare mobili virtuali nel proprio spazio abitativo per valutare come si abbinano all'arredamento e considerare l'impatto estetico prima dell'acquisto.
  - Cataloghi AR interattivi. È possibile sfogliare cataloghi di prodotti che prendono vita grazie a modelli 3D, video e informazioni aggiuntive accessibili con un semplice tocco sullo schermo.

- Educazione e apprendimento:

  - Apprendimento interattivo di lingue. Si ha la possibilità di apprendere nuove lingue interagendo con oggetti virtuali che "parlano" la lingua straniera e completando attività immersive.
  - Esplorazione della storia e della scienza. È possibile viaggiare nel tempo visitando virtualmente antichi siti archeologici o esplorando l'anatomia umana in 3D con modelli interattivi.
  - Apprendimento pratico di materie STEM. Si possono condurre esperimenti virtuali di chimica, fisica o biologia, visualizzando fenomeni scientifici in modo coinvolgente e sicuro.

- Navigazione e turismo:
  - Mappe AR interattive. Si possono esplorare mappe che si arricchiscono di informazioni contestuali, punti di interesse in 3D e indicazioni di navigazione in tempo reale attraverso sovrapposizioni AR.
  - Guide turistiche immersive. È possibile scoprire i segreti di una città o di un monumento storico con guide virtuali che accompagnano attraverso percorsi interattivi e ricostruzioni AR di epoche passate.
  - Esperienze turistiche personalizzate. Si ha l'opportunità di creare un itinerario di viaggio ideale, visualizzando recensioni AR, suggerimenti locali e informazioni su eventi e attrazioni in tempo reale.

- Social media e comunicazione:
  - Condivisione di esperienze AR. Catturare e condividere momenti unici con amici e follower tramite foto e video che integrano elementi AR personalizzati.
  - Filtri AR interattivi per social media. Creare contenuti divertenti e coinvolgenti utilizzando filtri AR che trasformano il volto, aggiungono effetti speciali o inseriscono elementi virtuali nei vostri video.
  - Interazione in tempo reale con oggetti virtuali. Gli utenti possono partecipare a esperienze AR condivise, interagendo con oggetti virtuali in uno spazio digitale comune.

Questi sono solo alcuni esempi di come l'AR stia trasformando le app mobili, offrendo esperienze utente uniche e rivoluzionarie. Con il continuo sviluppo della tecnologia e la crescente creatività degli sviluppatori, le possibilità di applicazione della realtà aumentata nelle app mobili sono infinite, pronte a sorprenderci e a cambiare il modo in cui interagiamo con il mondo che ci circonda. In questo contesto si inserisce il progetto dell'app presentata, che introduce il concetto di AR per la navigazione all'interno di ambienti complessi. L'idea alla base dell'app è di utilizzare la realtà aumentata per guidare l'utente in modo intuitivo e interattivo, sovrapponendo indicazioni visive in tempo reale all'interno dell'ambiente reale, facilitando l'orientamento e la scoperta di punti di interesse. Questo approccio non solo rende la navigazione più efficiente, ma arricchisce l'esperienza utente, sfruttando appieno le potenzialità dell'AR per un'interazione immersiva e coinvolgente.

## 2.3 Analisi dei requisiti

L'analisi dei requisiti rappresenta il primo passo fondamentale per il successo di qualsiasi progetto. In questa fase cruciale, si gettano le basi per comprendere a fondo le necessità, le aspettative e gli obiettivi degli stakeholder coinvolti. Come un architetto meticoloso

che delinea le fondamenta di un edificio, l'analista dei requisiti pone le pietre miliari per la realizzazione di un progetto solido e rispondente alle reali esigenze. Per iniziare efficacemente l'analisi dei requisiti, è necessario adottare un approccio metodico e strutturato. Inizialmente vanno definiti gli obiettivi ed il campo di applicazione del progetto: l'obiettivo principale del sistema è quello di permetterci di navigare all'interno degli edifici del campus universitario, permettendoci di raggiungere il punto di interesse nel modo più veloce ed efficiente possibile. Per poter definire al meglio le funzionalità da fornire, abbiamo dovuto prima definire gli stakeholder[1] del sistema. A seguito di un'analisi avente in considerazione i principali eventi che si svolgono all'interno dell'Ateneo, lezioni ed esami, iniziative di orientamente a futuri studenti, esami di abilitazione, fino ad arrivare a conferenze pubbliche, gli attori principali identificati sono:

- Studenti, i quali molte volte si trovano in difficoltà a raggiungere aree meno note o frequentate del campus
- Futuri studenti, che raggiungono la struttura per svolgere test di ingresso o in ambito di iniziative volte a far conoscere l'Ateneo
- Personale esterno, il quale raggiunge il campus in ambito di iniziative straordinarie come conferenze, open day e job service day
- Manutentori, molto spesso personale esterno con poca conoscenza della struttura
- Staff universitario, che comprende il personale dell'Ateneo che si occuperà della gestione dei dati dei punti di interesse e di parte dei feedback ricevuti
- Staff di sviluppo, personale di sviluppo che rimane in carica per la manutenzione e gestione dei feedback di tipo tecnico

Dopo un attenta indagine con gli stakeholder appena definiti, per quanto riguarda la navigazione sono state richieste le seguenti funzionalità:

- Selezionare il punto di partenza. Impostare il punto di interesse da dove l'utente vuole iniziare la navigazione, ciò deve poter essere fatto in due modalità:
  - scegliendo da una lista di opzioni filtrata in base all'edfificio ed al piano
  - attraverso la scansione di un codice QR[2] che deve essere posto in concomitanza di ogni punto di interesse
- Selezionare la destinazione. La possibilità di scegliere da una lista di opzioni oppurtanamente filtrate per piano all'interno dell'edificio di partenza.
- Terminare la navigazione. Oltre all'automatica terminazione quando si è raggiunta la destinazione, dove esser lasciata la possibilità di terminare anticipatamente la navigazione attraverso un'apposiza opzione.
- Visualizzare il percorso. Duranta la navigazione è necessaria la possibilità di visualizzare una panoramica completa del percorso proposto per la navigazione.

[1] gli stakeholder rappresentano attori chiave del sistema software, quali: utenti, membri del team di sviluppo,...

[2] Un codice QR è un codice a barre bidimensionale, ossia a matrice, composto da moduli neri disposti all'interno di uno schema bianco di forma quadrata, impiegato in genere per memorizzare informazioni destinate a essere lette tramite un apposito lettore ottico o anche smartphone.

Mentre dallo staff universitario e di sviluppo sono emersi dei requisiti di carattere più gestionale, che avranno il compito di semplificare le attività di manutenzione del sistema software. La prima area di interesse proposta dallo staff, ovvero la gestione dei feedback degli utenti, riguarda gli altri stakeholder coinvolti, i quali sono interessati dalla prima delle seguenti funzionalità:

- Raccolta feedback utente. Il sistema software deve permettere agli utenti di poter lasciare un feedback sull'esperienza ed eventuali dati di ricontatto se dovessero essere necessari.
- Classificazione dei feedback. Il sistema software deve classificare i feedback ricevuti differenziando quelli di carattere tecnico da quelli generici. In seguito quelli di carattere generico devono essere poi divisi in 3 categorie: positivi, negativi, urgenti. L'ultima categoria indica che serve un intervento immediato per garantire il corretto funzionamento del sistema.
- Gestione dei feedback ricevuti. Il personale dello staff universitario dovrà essere immediatamente notificato qualora venisse ricevuto un feedback categorizzato come urgente. Deve essere comunque garantita la possibilità di visualizzare tutti i feedback ricevuti divisi in categorie.
- Visualizzazione delle statistiche. Deve essere permesso allo staff di poter visualizzare le statische relative ai dati di navigazione, quali tratte più richieste, e dei feedback, in modo da analizzare la stabilità del sistema.

Sono presenti anche delle richieste che riguardano la gestione dei dati ambientali, ovvero dei dati di alcuni tipi interesse che possono semplificare il processo di ricerca dell'utente, che dovranno poter essere modificati in autonomia dal personale universitario:

- Fornire l'orario delle aule. Se il punto di interesse di destinazione è classificato come aula, l'applicativo dovrà fornire a schermo i suoi orari.
- Fornire l'orario degli uffici. Qualora il punto di interesse di destinazione è classificato come ufficio, il programma dovrà fornire a schermo i suoi orari. Nel caso si trattasse di un ufficio docente fornirà gli orari in cui il docente sta svolgendo attività di insegnamento.
- Fornire gli orari della segreteria. Il sistema software dovrà rendere accessibili in qualsiasi momento gli orari delle varie segreterie.
- Modificare la mappa degli edifici dell'Ateneo. Deve essere possibile fare variazioni alla mappa da remoto qualora alcune zone dovranno essere interdette.

Analizzando i requisiti raccolti fino a questo momento, possiamo notare la necessità di una piattaforma da mettere a disposizione dello staff per poter soddisfare le loro richieste con più semplicità, che da ora chiameremo piattaforma di amministrazione per distinguerla dal sistema a disposizione degli utenti. Da qui nasce la necessità di elencare alcuni requisiti base riguardanti questa piattaforma:

- Autenticazione. E' necessario autenticare l'utente che prova ad accedere in modo tale da assicurarsi che abbia i privilegi per potervi accedere.
- Recupero credenziali. Il sistema deve garantire all'amministratore la possibilità di reimpostare le credenziali di accesso qualora non fossero più a sua disposizione.

- Gestione account amministratori. Solamente gli amministratori autenticati possono fornire nuove credenziali di accesso a nuovi utenti che potrebbero avere necessità di accedere alla piattaforma.

Studiando le necessità emerse fino a questo momento segue la necessità di esplicare alcuni vincoli non funzionali che potrebbero rimanere poco chiari:

- Implementazione del sistema backend in python3.
- Implementazione dell'applicazione mobile in Unity per poter integrare facilmente la realtà aumentata.
- Le credenziali di accesso degli amministratori saranno gestite internamente al sistema.
- Il database viene implementato usando Firebase.

## 2.4 Le fasi della progettazione

Basandoci sui requisiti appena definiti possiamo proseguire con la progettazione, andando a rappresentare i diagrammi di progettazione che vengono divisi in due settori principali:

- Diagrammi di analisi: prima fase nel processo di sviluppo software, questo tipo di diagrammi si concentrano sulla rappresentazione dei requisiti del sistema e sulle interazioni tra gli utenti e il sistema stesso. I principali tipi di diagrammi di analisi includono:
  - Diagramma dei casi d'uso, rappresentano le interazioni tra gli attori esterni (gli utenti o altri sistemi) e il sistema, identificando i vari casi d'uso o scenari di utilizzo del sistema.
  - Diagramma delle classi, vengono utilizzati per rappresentare in modo conciso le entità e le relazioni chiave all'interno del sistema, concentrandosi principalmente sulla struttura concettuale e sulle relazioni statiche tra gli oggetti.
  - Diagramma di attività, descrivono il flusso di lavoro o il processo all'interno del sistema, evidenziando le azioni e le decisioni che si verificano in sequenza.
- Diagrammi di progettazione: si concentrano sulla definizione dell'architettura del sistema, dei suoi componenti e delle relazioni tra di essi. I principali tipi di diagrammi di progettazione includono:
  - Diagramma delle classi, rappresentano le classi all'interno del sistema e le relazioni tra di esse, inclusi gli attributi e i metodi di ciascuna classe.
  - Diagramma di sequenza, mostrano l'interazione tra gli oggetti o le classi all'interno del sistema, evidenziando la sequenza temporale delle operazioni durante l'esecuzione di un caso d'uso specifico.
  - Diagramma delle macchine a stati, utilizzati per modellare il comportamento dinamico di un sistema attraverso una serie di stati e transizioni.
  - Diagramma dei componenti, utilizzati per rappresentare la struttura ad alto livello del sistema e le relazioni tra i componenti software.

Nelle seguenti sezioni andiamo ad apprfondire tutte le attività legate alla navigazione.

### 2.4.1 Diagrammi di analisi

#### Diagramma dei casi d'uso

I casi d'uso principali rappresentano la gestione della navigazione. Questo scenario rappresenta nel complesso tutte le azioni che si possono, e vengono, svolte durante la navigazione. L'attività di "Avvia Navigazione" richiede che venga caricato il percorso, questo viene evidenziatao dall'include. Il caricamento del percorso include al suo interno due attività fondamentali che contribuiscono al suo completamento: "SelezionaPartenza" e "SelezionaDestinazione". La partenza, come specificato nei requisti, può essere fatta scegliendo il punto di interesse dalla lista oppure scansionando il codice QR presente fuori dal punto di interesse. Completata la scelta le informazioni sulla partenza e sulla destinazione vengono trasmesse al sistema che elabora il percorso, a questo punto possiamo avviare la navigazione. L'applicativo da anche la possibilità di visualizzare la panoramica del percorso attraverso "VisualizzaPercorso", viene indicato come estensione di "AvviaNavigazione" perché è uno scenario facoltativo. "TerminaNavigazione" è uno scenario che può essere attivato sia dall'utilizzatore che dal sistema. Nel primo caso si tratta di una terminazione anticipata della navigazione, mentre nel secondo caso viene attivata automaticamente al raggiungimento della destinazione. In questa fase viene proposto all'utente di lasciare un feedback sull'esperienza di navigazione ed in seguito la possibilità di tornare al punto di partenza e configurare una nuova navigazione.

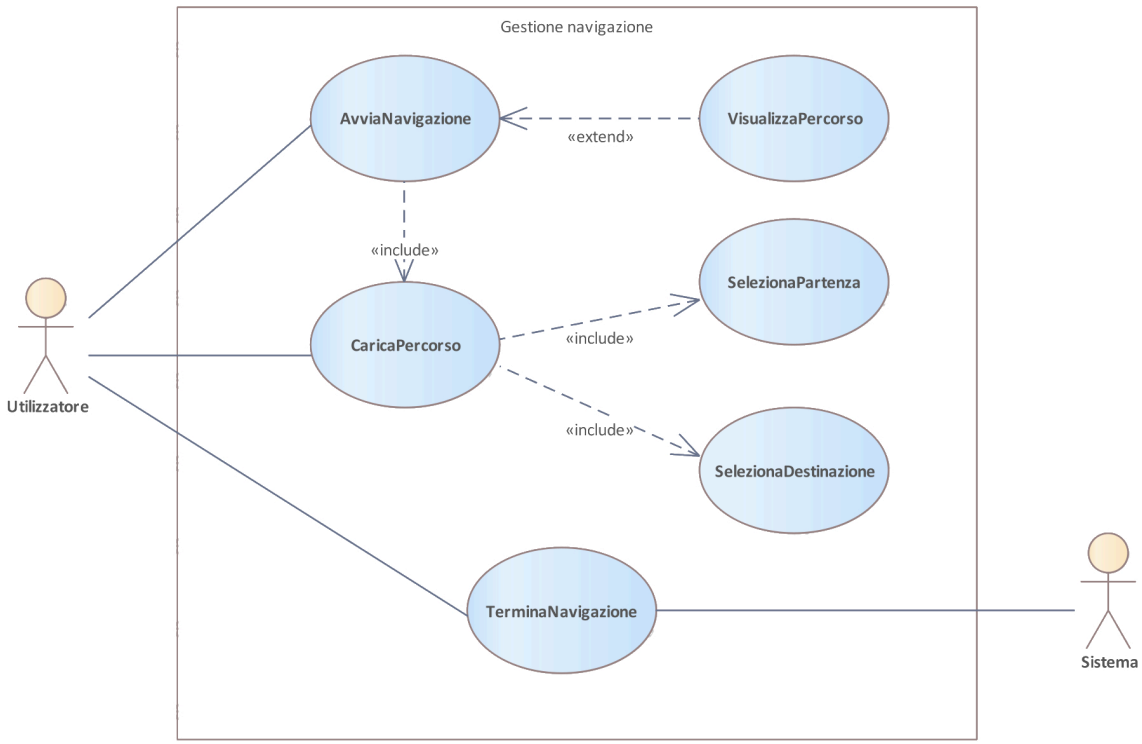


**Figura 2.1:** Diagramma del caso d'uso gestione navigazione

| **Caso d'uso:** CaricaPercorso |
|---|
| **Breve descrizione:**<br>Il sistema permette all'utente di caricare il percorso selezionato. |
| **Attori primari:**<br>Utente |
| **Precondizioni:**<br>Nessuna |
| 1. Il caso d'uso inizia quando l'utente vuole iniziare una nuova navigazione;<br>2. *include* (SelezionaPartenza);<br>3. *include* (SelezionaDestinazione);<br>4. Il sistema elabora il percorso;<br>5. Il sistema salva il percorso; |
| **Postcondizioni:**<br>Percorso caricato |
| **Sequenza degli eventi alternativa:**<br>Nessuna |

| **Caso d'uso:** AvviaNavigazione |
|---|
| **Breve descrizione:**<br>Il sistema permette all'utente di avviare la navigazione. |
| **Attori primari:**<br>Utente |
| **Precondizioni:**<br>Nessuna |
| 1. Il caso d'uso inizia quando l'utente vuole avviare la navigazione;<br>2. *include* (CaricaPercorso);<br>3. *if* L'utente seleziona la voce per visualizzare il percorso;<br>(a) *extend* (VisualizzaPercorso);<br>4. La navigazione inizia e l'utente viene guidato fino a destinazione con le opportune indicazioni da parte del sistema; |
| **Postcondizioni:**<br>Navigazione avviata |
| **Sequenza degli eventi alternativa:**<br>Nessuna |

Le attività collaterali alla navigazione vengono svolte dagli amministratori di ogni Università. Il primo amministratore viene aggiunto in fase di configurazione da parte degli sviluppatori, in seguito i nuovi amministratori dovranno seguire la procedura prevista in "RichiediCredenziali" e successivamente gli amministratori precedentemente abilitati possono accettare le richieste ricevute attraverso i passi specificati nell'attività "AccettaAmministratore". L'attività principale che possono svolgere riguarda la consultazione delle statistiche riguardanti i dati di navigazione degli utenti all'interno della specifica Università. Queste generalmente vengono calcolato in situazioni di basso carico dal sistema, così che siano sempre pronte alla consultazione. In casi particolari sono autorizzati a modificare gli orari della segreteria o ad apportare modifiche temporanee alle mappe degli edifici nel caso in cui alcune aree siano chiuse per lavori o eventi.

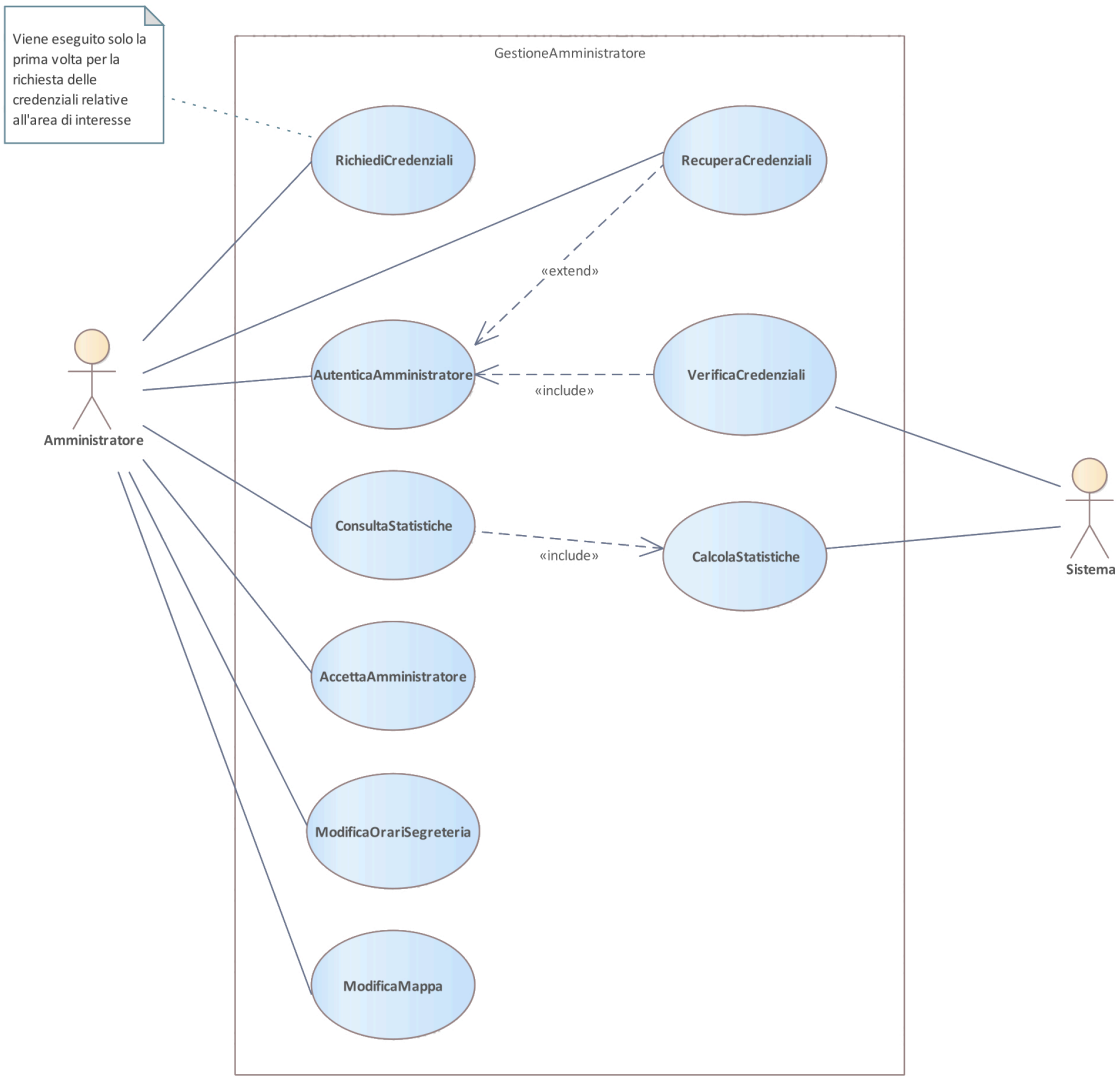


**Figura 2.2:** Diagramma del caso d'uso gestione amministrazione

| **Caso d'uso:** ConsultaStatistiche |
|---|
| **Breve descrizione:**<br>L'amministratore consulta le statistiche. |
| **Attori primari:**<br>Amministratore |
| **Precondizioni:**<br>L'amministratore deve essere autenticato; |
| 1. Il caso d'uso inizia quando l'amministratore desidera consultare le statistiche;<br>2. *include* (CalcolaStatistiche);<br>3. Il sistema visualizza le statistiche; |
| **Postcondizioni:**<br>L'amministratore visualizza le statistiche; |
| **Sequenza degli eventi alternativa:**<br>Nessuna |

**Diagramma delle classi**

I diagrammi delle classi di analisi rappresentano una componente fondamentale nel processo di sviluppo, in quanto offrono una prima rappresentazione strutturata delle entità che compongono il sistema e delle loro interrelazioni. Attraverso tali diagrammi, è possibile individuare le principali classi del sistema, definirne gli attributi e i metodi, nonché stabilire le relazioni e le associazioni che intercorrono tra di esse. Questo approccio permette di tradurre i requisiti funzionali in un modello visivo e concettuale, il quale costituisce una base solida per la progettazione e l'implementazione successiva. Inoltre, i diagrammi delle classi di analisi consentono di mantenere un elevato grado di coerenza e correttezza all'interno del modello concettuale, assicurando che il sistema software sviluppato risponda in maniera adeguata alle esigenze specificate.

Per ottimizzare la gestione delle classi si è deciso di creare tre package principali:

- Models. Questo package include tutte le classi comuni sia all'applicazione frontend che al servizio API. Le classi presenti in questa sezione definiscono i modelli di dati condivisi e utilizzati da entrambi i livelli dell'applicazione, garantendo coerenza e riusabilità.

- Frontend. Il package frontend contiene esclusivamente le classi necessarie per il funzionamento dell'applicazione mobile. Qui sono definite le componenti specifiche per l'interfaccia utente e la logica che si occupa dell'interazione con l'utente finale.

- Api. Il package frontend contiene esclusivamente le classi necessarie per il funzionamento dell'applicazione mobile. Qui sono definite le componenti specifiche per l'interfaccia utente e la logica che si occupa dell'interazione con l'utente finale.

In prima analisi all'interno del package "Models" sono stati inserite le classi fondamentali per gestire la posizione degli interest point e lo stato della navigazione. Infatti le classi: University, Building, Floor e InterestPoint si occupano della gestione del punto di interesse

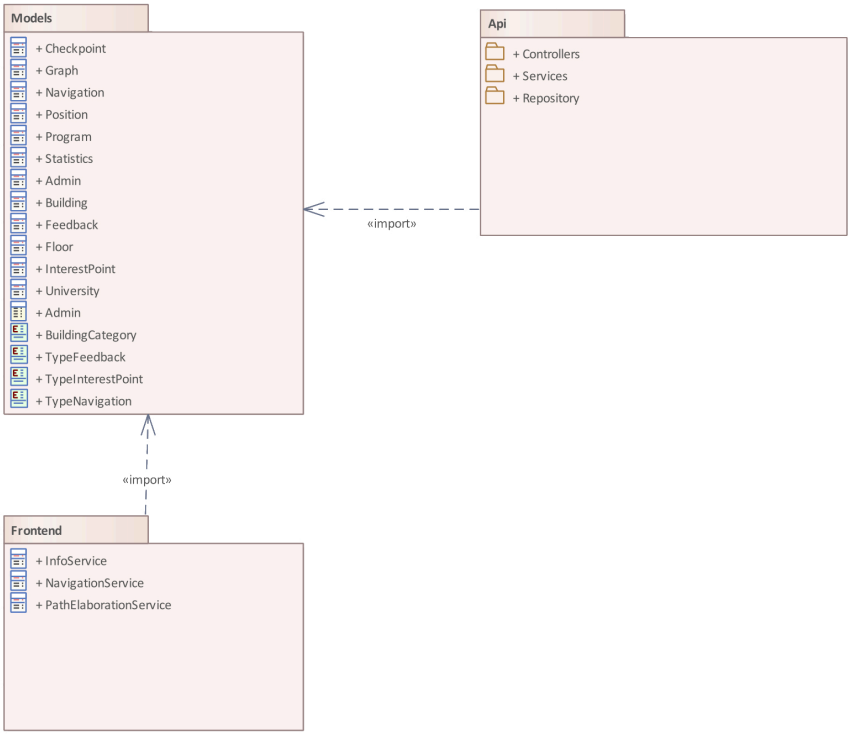


**Figura 2.3:** Organizzazione dei package

nello spazio. Nello specifico InterestPoint contiene al suo interno anche un oggetto Position che attraverso le coordinate ci permette di identificarlo nella mappa. Mentre Navigation si occupa di mantenere i dati della navigazione, quali posizione di partenza, posizione attuale, checkpoint superati, checkpoint da superare e interest point di arrivo. A supporto di Navigation abbiamo TypeNavigation che ci permette di gestire lo stato della navigazione, quale avviata, in corso o terminata, e Checkpoint che ci aiuta a gestire l'ordine dei punti di interesse da attraverso ed aggiorna lo stato in caso di superamento di esso.

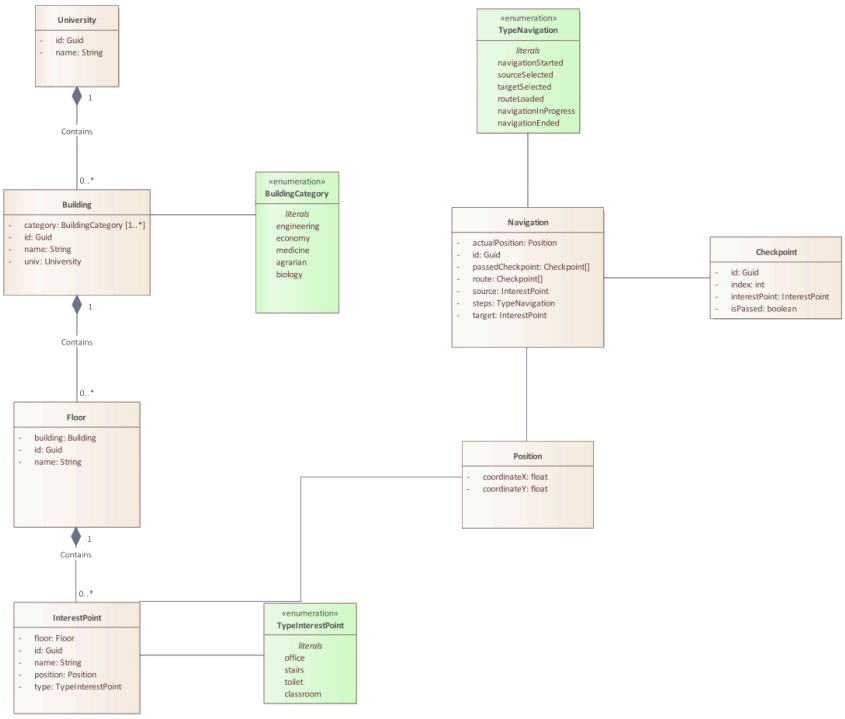


**Figura 2.4:** Una parte delle classi contenute in "Models"

In questa prima fase all'interno del package "Frontend" sono state inserite solamente tre classi a cui poi in fase di progettazione andremo ad aggiungere dei servizi di utility che ci permettono di garantire un corretto funzionamento. Attraverso la classe NavigationService andiamo a caricare tutti gli interest point presenti all'interno dell'Univrsità da cui poter scegliere il punto di partenza e la destinazione. Una volta completata la scelta attraverso il metodo "getRoute" andiamo a comunicare con il servizio API che dati come parametri i punti di interesse appena scelti ci fornisce la sequenza di punti di interesse da attraversare. Utilizzando il PathElaborationService andiamo a disegnare nell'interfaccia utente il tragitto che dovrà compiere per arrivare a destinazione. La classe InfoService viene utilizzata per caricare l'orario del punto di interesse al momento dell'arrivo ed anche per inviare al nostro backend, nel caso venga fornita, la recensione sull'esperienza di navigazione per raggiungere la destinazione.

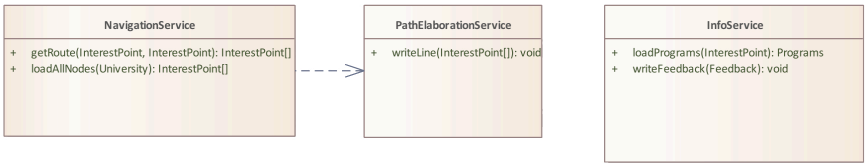


**Figura 2.5:** Le classi di analisi contenute in "Frontend"

Per quanto riguarda il servizio API, le classi sono state divise in quattro namespace[3]:

- Controllers. Rappresentano le classi responsabili della gestione delle chiamate HTTP in ingresso. Ogni controller espone uno o più endpoint, ciascuno associato a una specifica richiesta, indirizzando la chiamata verso il servizio appropriato. I controller fungono

[3]Namespace: spazio dei nomi che raggruppa un insieme di classi

da punto di ingresso per le operazioni dell'API, orchestrando il flusso dei dati tra l'interfaccia utente e la logica di business. Sono stati creati tre controller principali per dividere le azioni eseguite dall'amministrazione, da quelle riguardanti la navigazione e il recupero degli orari dei punti di interesse. Per quanto riguarda l'amministrazione è stato predisposto un endpoint dedicato alla sola autenticazione.

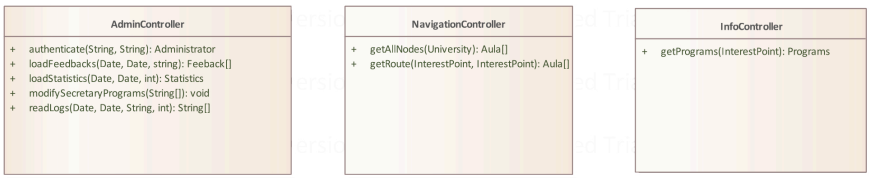


**Figura 2.6:** Le classi di analisi contenute in "Controllers"

- Services. Le classi di servizio gestiscono l'elaborazione delle richieste ricevute dai controller. Esse contengono la logica di business applicativa, agendo come intermediari tra i controller e i livelli sottostanti dell'applicazione, quali il repository o altri servizi esterni. I servizi sono progettati per mantenere una separazione chiara tra la logica di business e la gestione delle chiamate HTTP. Seguendo lo schema applicato ai controllers anche per i servizi sono state create tre classi che dividono l'area di business in cui devono operare. Tutti i servizi estendo il LoggerService che viene utilizzato per mantenere traccia delle richieste eseguite e relativo carico di lavoro per predisporre futuri miglioramenti e manutenzione del sistema.

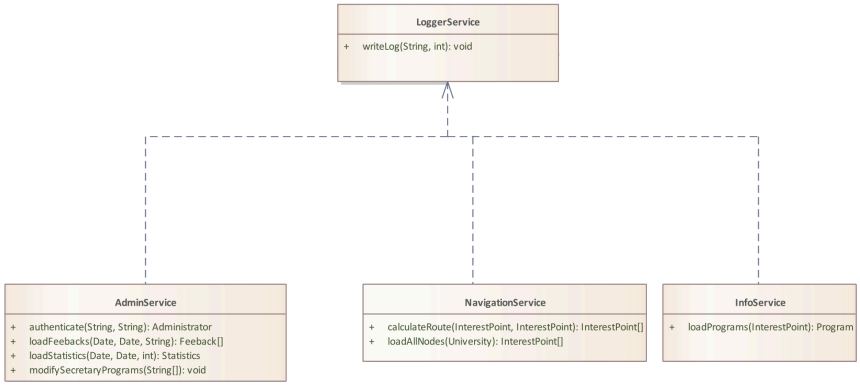


**Figura 2.7:** Le classi di analisi contenute in "Services"

- Helpers. Raggruppa l'insieme delle classi che forniscono delle funzioni a supporto dei controllers e dei services. Gli helpers possono includere utility per la conversione di dati, validazione, formattazione, o altre operazioni ripetitive e generiche che non rientrano direttamente nella logica di business. Essi contribuiscono a mantenere il codice modulare e riutilizzabile.
- Repository. Le classi di repository gestiscono l'accesso ai dati, fungendo da interfaccia tra il sistema e la fonte dei dati, come un database o un servizio esterno. I repository si occupano delle operazioni di persistenza, recupero, aggiornamento e cancellazione dei dati, astraggono la logica di accesso ai dati dalla logica di business, permettendo ai servizi di interagire con i dati senza dipendere dalla specifica implementazione della persistenza. Questo approccio facilita la manutenzione e la scalabilità del sistema. Naturalmente la divisione in tre classi è stata condivisa anche con le repository per

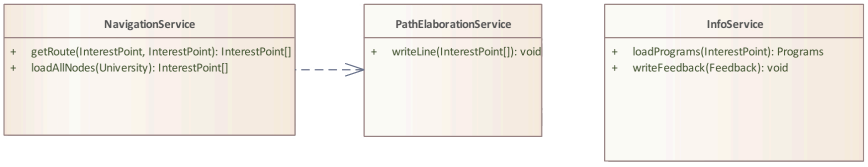


**Figura 2.8:** Le classi di analisi contenute in "Frontend"

mantenere lo stesso principio di divisione all'interno dell'elaborazione delle richieste alla nostra base di dati.


**Figura 2.9:** Le classi di analisi contenute in "Repository"

### 2.4.2 Diagrammi di progettazione

I diagrammi di progettazione, sono utilizzati nelle fasi successive avanzate di progettazione del sistema software. Questi diagrammi sono più dettagliati e tecnici, poiché descrivono come il sistema sarà effettivamente implementato. Il focus è quindi sull'organizzazione delle componenti del sistema e sulle interazioni precise che devono avvenire per soddisfare i requisiti.

#### Diagramma delle classi

Nel processo di progettazione, le classi di progettazione sono state sviluppate estendendo e raffinando le classi individuate nella fase di analisi. Queste classi di progettazione non solo mantengono la struttura concettuale definita in analisi, ma ne approfondiscono i dettagli, introducendo attributi specifici, metodi concreti e relazioni tecniche necessarie per l'implementazione. Nel package dei "Models" è stata introdotta una classe "Base" da cui tutte le classi ereditano gli attributi id e name con i rispettivi getter e setter. Rispetto al diagramma delle classi di analisi abbiamo aggiunto varie entità collaterali per raggiungere gli obiettivi prefissati:

- Admin. Per gestire le autorizzazioni concesse agli amministratori e gli edifici delle varie Università a cui possono applicare le varie modifiche.
- Program e OpeningTime. Program rappresenta la classe che ci permettere di leggere e salvare gli orari della segreteria, a cui viene in aiuto OpeningTime che divide gli orari di apertura della mattina da quelli del pomeriggio per ogni giorno settimanale. La classe Program ha un array di 7 OpeningTime.
- Statistics. Ci permette di avere la sequenza di valori da visualizzare in un ipotetico grafico e, in caso sia disponibile, anche la media dei valori raccolti. Si è scelgo di mantenere l'attributo values generico per renderlo compatibile con la maggior parte delle situazioni.
- Feedback. Per quanto concerne la raccolta delle informazioni relative all'esperienza di navigazione dell'utente, è stata adottata una struttura dati che include l'Università di appartenenza, il messaggio, e l'indirizzo email, quest'ultimo necessario per eventuali contatti futuri. Inoltre, è stato previsto un campo specifico per il tipo di feedback, al fine

di consentirne la classificazione in base alla sua rilevanza, distinguendo tra feedback urgente, negativo o positivo.

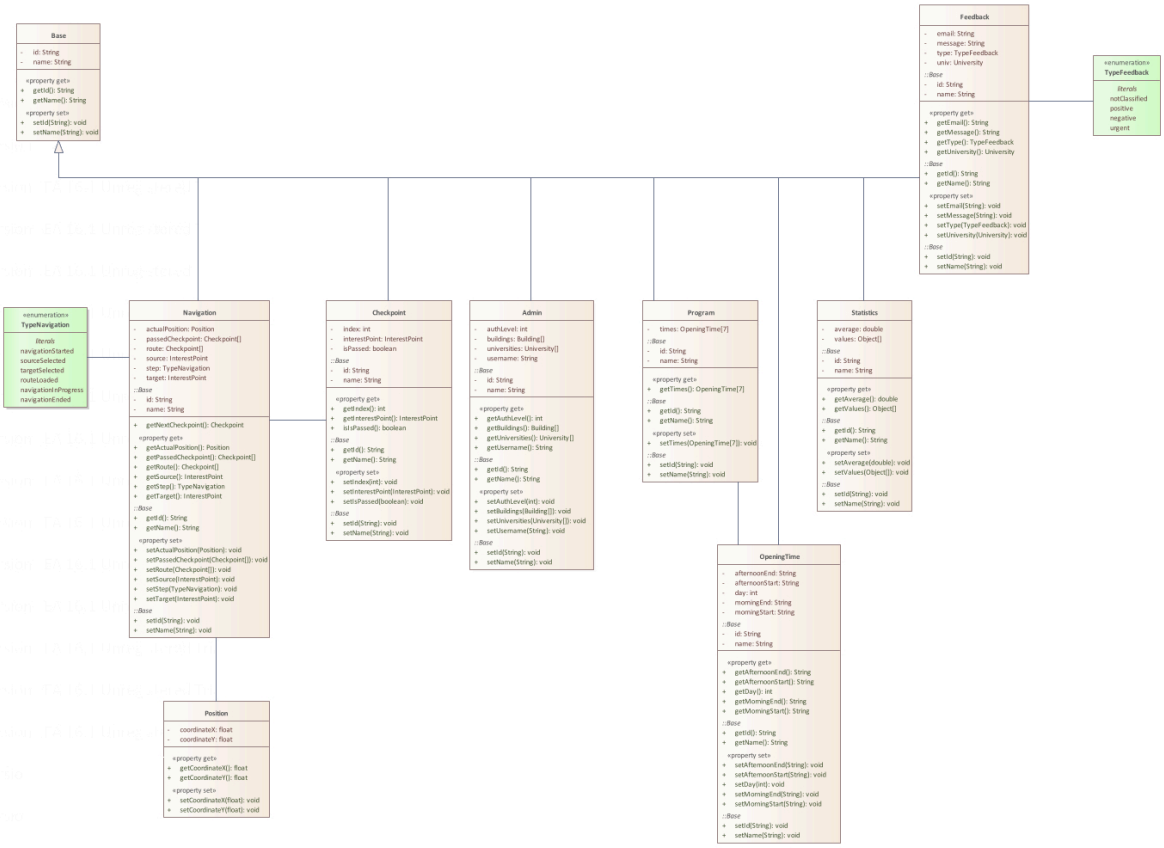


**Figura 2.10:** Le classi di progettazione contenute in "Models"

Dal lato API, i controller sono stati ampliati, con un significativo aggiornamento del NavigationController. In particolare, gli endpoint forniti da questo controller sono stati riorganizzati, distinguendo in modo netto il recupero dei nodi per ciascun edificio e piano, piuttosto che utilizzare un metodo generale. Questa modifica è stata implementata per ottimizzare i tempi di risposta, delegando il calcolo al database, che dispone di una maggiore capacità di eseguire tali query in modo efficiente. Inoltre, l'InfoController è stato esteso con l'aggiunta di un endpoint dedicato al salvataggio del feedback dell'utente. In seguito alle

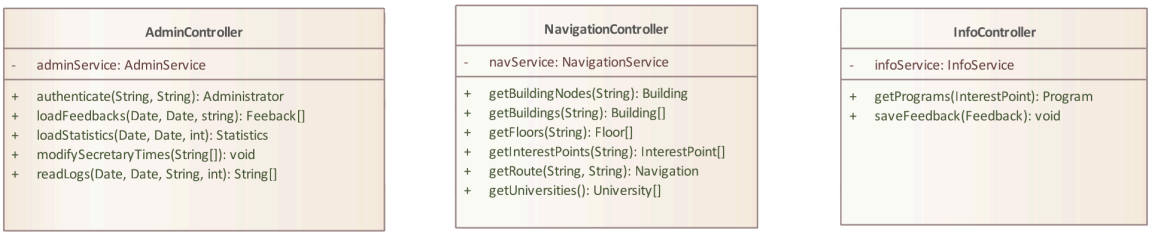


**Figura 2.11:** Le classi di progettazione contenute in "Controllers"

modifiche apportate ai controller, anche i servizi (Service) sono stati adeguati, con particolare riferimento al NavigationService, che ha implementato i metodi necessari per supportare i nuovi endpoint creati. Sono state inoltre introdotte interfacce per creare un livello di astrazione che separa le richieste, con i relativi parametri e valori di ritorno, dall'implemen-

tazione specifica nel servizio. Questo approccio rende il servizio indipendente e facilmente modificabile. Inoltre, è stato adottato il pattern singleton per garantire che ogni servizio venga istanziato una sola volta, ottimizzando l'utilizzo delle risorse e garantendo un accesso coerente ai servizi stessi.

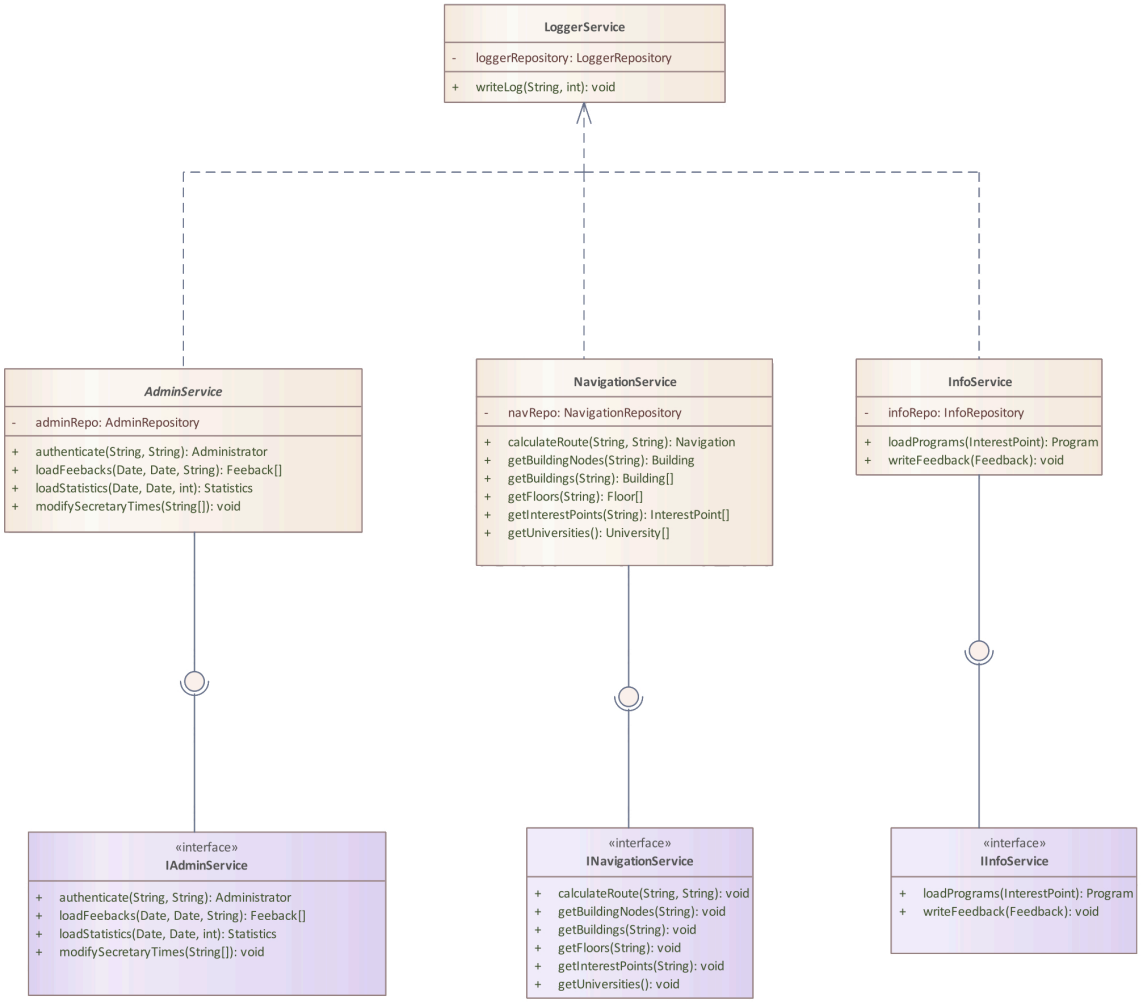


**Figura 2.12:** Le classi di progettazione contenute in "Services"

Anche il frontend è stato oggetto di una significativa revisione dei metodi necessari, con il NavigationController che ora implementa separatamente la selezione dell'Università, dell'edificio, del piano e del punto di interesse. Contestualmente, è stata introdotta la funzionalità che consente di scegliere il punto di partenza scansionando il codice QR situato all'ingresso di ogni punto di interesse, una soluzione pensata per accelerare il processo di selezione del punto di partenza. Di conseguenza, il NavigationService è stato adeguato per gestire queste nuove esigenze e, accanto ad esso, è stato integrato il NavigationStorage, il quale si occupa di recuperare i dati richiesti e può applicare una logica di caching per ottimizzare le performance. Inoltre, è stato introdotto l'ApiHelper, che facilita la gestione delle chiamate HTTP verso l'API, garantendo una gestione uniforme degli errori nelle richieste e la formattazione corretta dei dati nella struttura designata, attraverso l'uso di metodi generici (Generics).

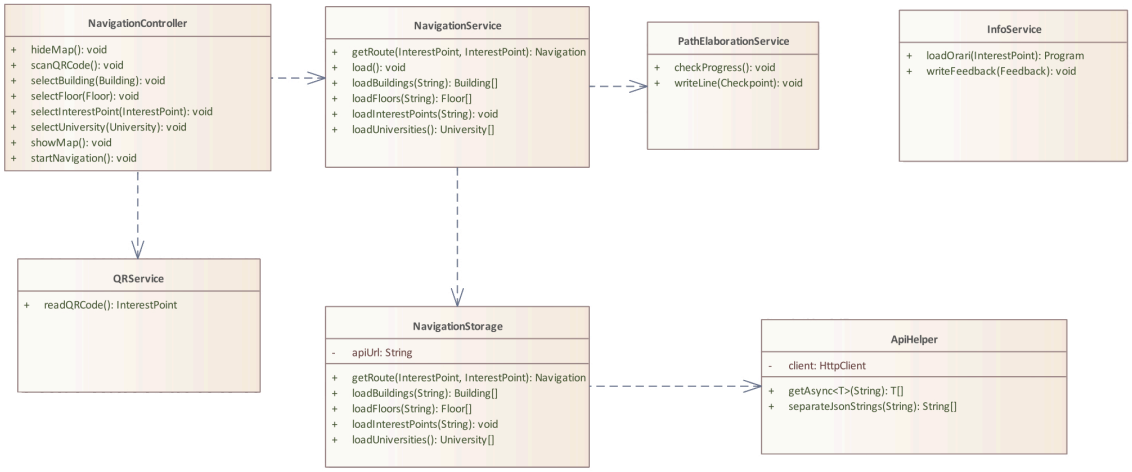


**Figura 2.13:** Le classi di progettazione contenute in "Frontend"

### Diagramma di sequenza

I diagrammi di sequenza rappresentano un elemento cruciale nella modellazione dei sistemi orientati agli oggetti, in quanto permettono di visualizzare l'interazione tra gli oggetti nel tempo. Questi diagrammi, parte integrante del linguaggio UML (Unified Modeling Language), descrivono come gli oggetti di un sistema collaborano per svolgere una funzione o un processo specifico, illustrando la sequenza di messaggi scambiati tra di essi. Questi diagrammi sono particolarmente utili per analizzare il comportamento dinamico di un sistema, permettendo di identificare chiaramente il flusso di controllo e la sequenza delle operazioni. Inoltre, forniscono una visione dettagliata del processo di comunicazione tra gli oggetti, facilitando così la comprensione delle interazioni complesse e la verifica della correttezza del modello rispetto ai requisiti definiti.

Analizziamo i principali diagrammi di sequenza che riguardano la navigazione quali "Carica percorso", "Avvia navigazione" e "Visualizza percorso". Partendo dall'attività del caricamento del percorso possiamo notare che parte solamente dopo aver selezionato la partenza e la destinazione, attraverso il NavigationService del Frontend andiamo a richiedere il percorso da seguire al nostro servizio API. Come descritto in precedenza la richiesta viene gestita inizialmente dal NavigationController che si occupa poi di indirizzarla alla corretta funzione del NavigationService. Se non già precedentemente calcolato si procede con la creazione del grafo dell'intero edificio su cui stiamo preparando la navigazione, quindi a partire dal punto di partenza riusciamo a risalire al piano e successivamente all'edificio da cui andiamo a recuperare tutti i punti di interesse. Dato il grafo a disposizione possiamo procedere con il calcolo del percorso, che in un primo momento è stato implementato utilizzando l'algoritmo di Dijkstra e successivamente sostituito dalla Graph Neural Network, che ci restituisce la lista dei Checkpoint da attraversare per arrivare a destinazione. Tutte le informazioni vengono inserite all'interno di un oggetto di tipo Navigation e restituite al client[4] che ha inviato la richiesta.

[4]Client: componente o dispositivo remoto che invia la richiesta al server.

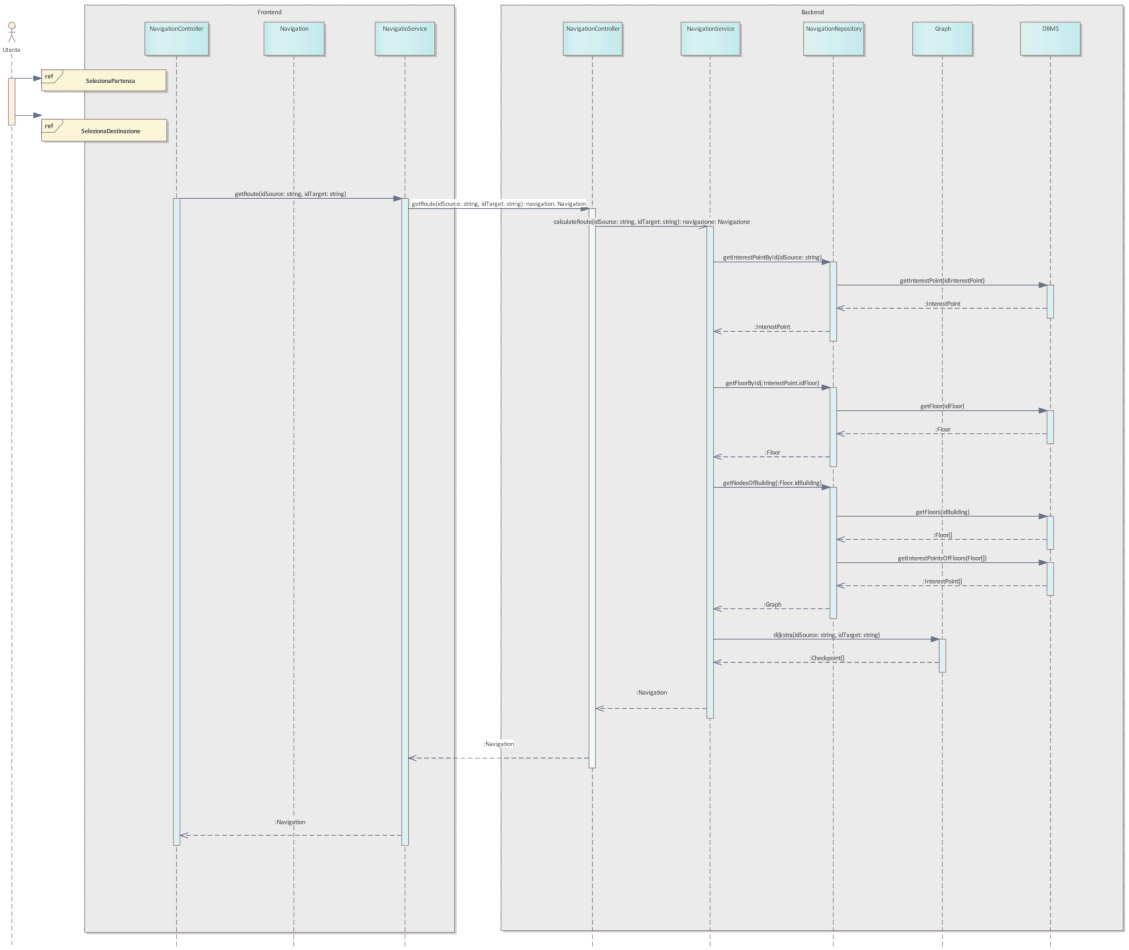


**Figura 2.14:** Diagramma di sequenza Carica percorso

Una volta caricato il percorso è possibile avvia la navigazione. Attraverso l'interfaccia grafica l'utente può dare inizio alla procedura di navigazione cliccando il pulsante predisposto, successivamente il NavigationController gestisce l'interazione ed attraverso il PathElaborationService aggiorna lo stato della Navigation a NavigationStarted. Successivamente inizia a leggere uno ad uno i Checkpoint e disegna la linea da seguire per raggiungerlo. Una volta raggiunto recupera il Checkpoint successivo e disegna la nuova linea, questo fino al momento in cui la navigazione viene terminata dall'utente oppure si giunge a destinazione. Lateralmente all'esperienza di navigazione diretta è possibile avviare l'attività Visualizza Percorso che ci permette di avere una visuale panoramica del tragitto da seguire per arrivare alla destinazione finale.

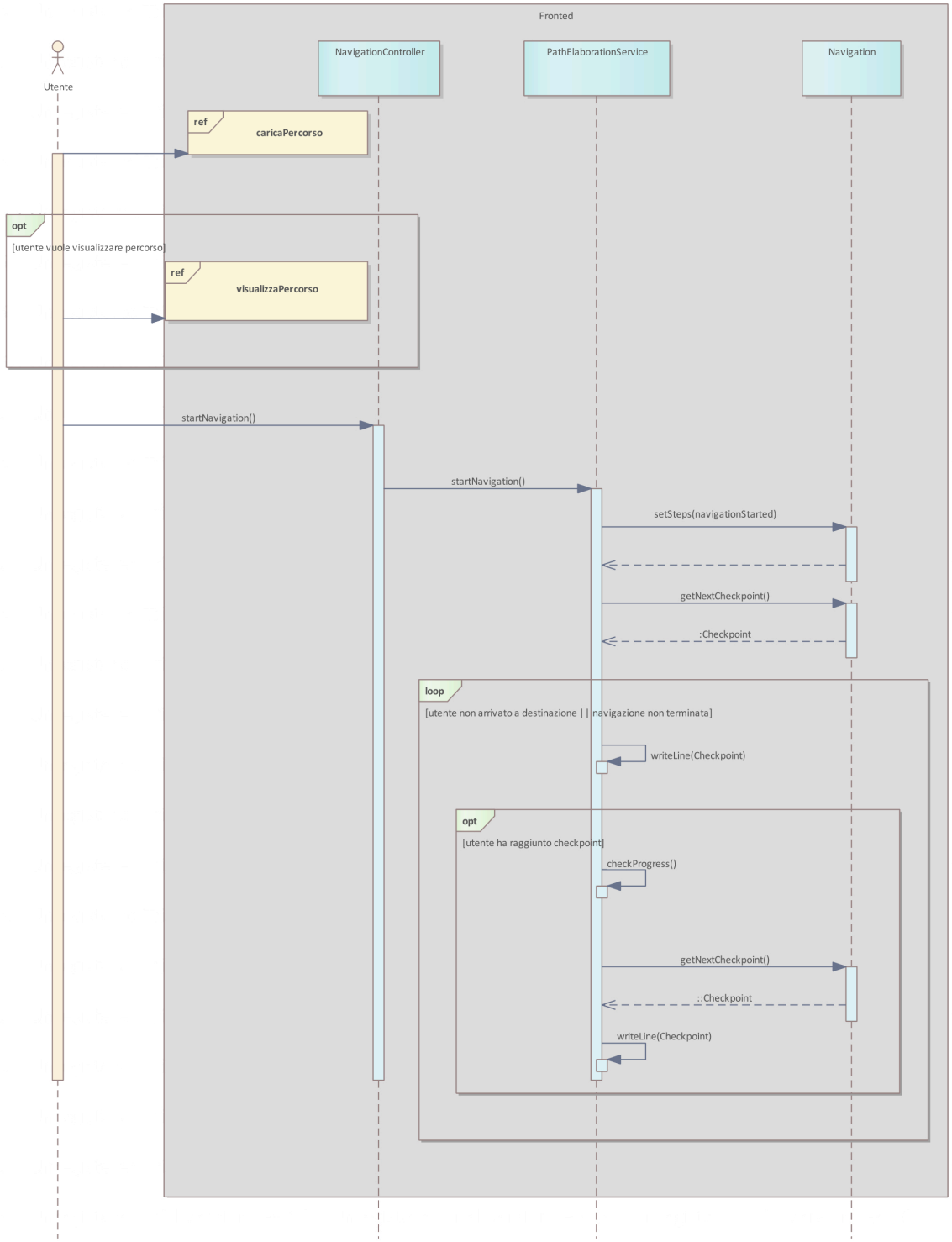


**Figura 2.15:** Diagramma di sequenza Avvia navigazione

Come accennato in precedenza è stata predisposta l'opzione di poter visualizzare una panoramica del percorso da seguire fino alla destinazione. L'avvio di tale attività è completamente facoltativo e non influisce con il normale svolgimento delle funzioni principali. Interagendo con la minimappa, che avrà un comportamento da pulsante, il NavigationController gestisce la richiesta andando ad impostare la visibilità della mappa rendendola visibile.

Nel momento in cui l'utente voglia ridurre lo spazio dedicato alla mappa, e quindi tornare alla precedente schermata di visualizzazione, puo' farlo andando a premere in un'area al di fuori del popup che il NavigationController riconosce e va a reimpostare la visibilità della mappa.

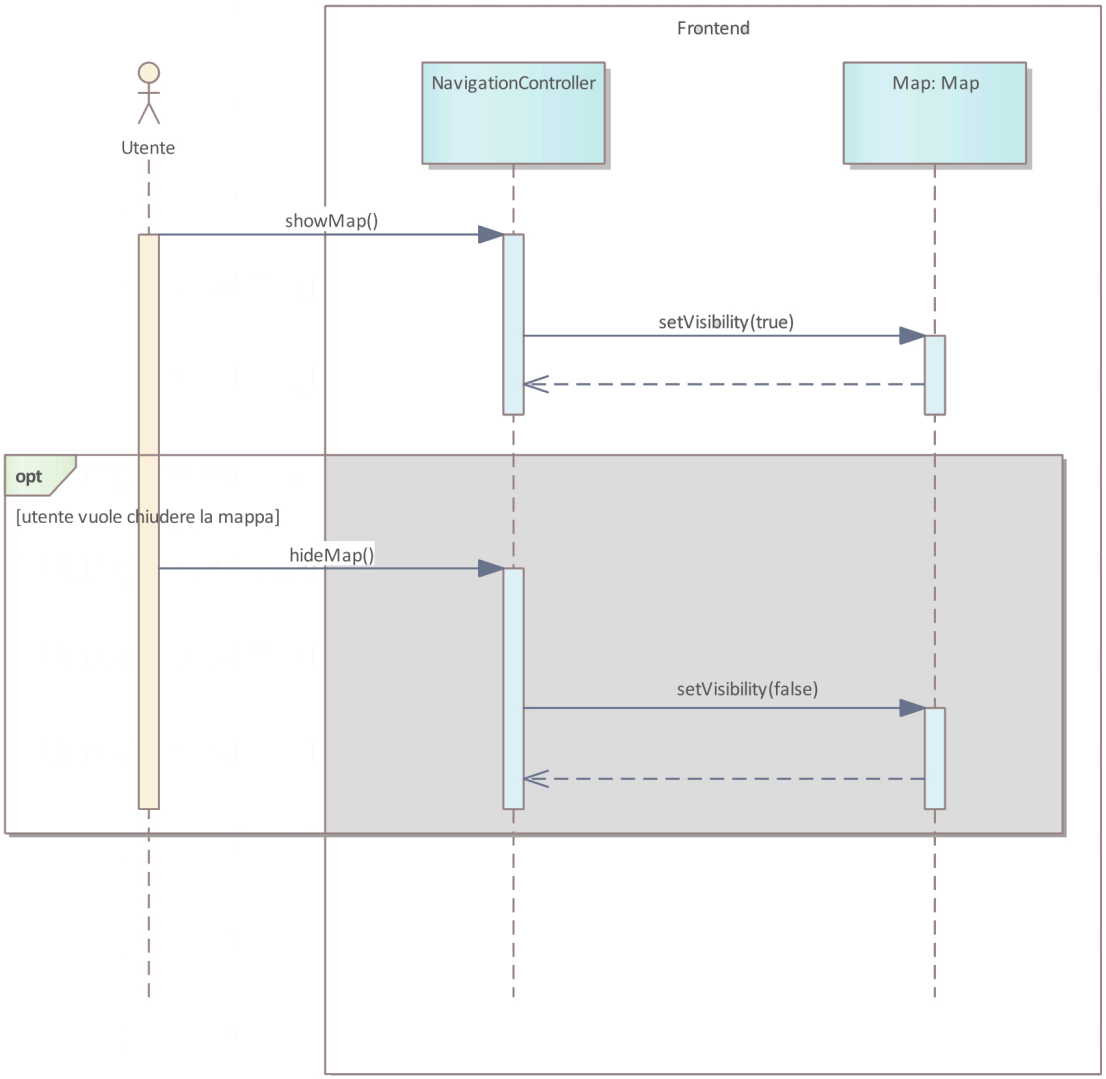


**Figura 2.16:** Diagramma di sequenza Visualizza percorso

### Diagramma delle macchine a stati

I diagrammi delle macchine a stati, noti anche come diagrammi di stato, sono uno strumento fondamentale nella modellazione del comportamento dinamico di un sistema in ingegneria del software. Questi diagrammi rappresentano gli stati possibili che un oggetto o un sistema può assumere durante il suo ciclo di vita e le transizioni tra questi stati in risposta a determinati eventi o condizioni. Ogni stato in un diagramma delle macchine a stati rappresenta una particolare condizione o situazione in cui l'oggetto si trova in un dato momento. Le transizioni, rappresentate da frecce, indicano il passaggio da uno stato all'altro, generalmente in risposta a un evento esterno, come un input da parte dell'utente o un cambiamento di condizione interna. Le transizioni possono anche essere condizionate, richiedendo che certe condizioni siano soddisfatte prima che il passaggio possa avvenire. I diagrammi delle macchine a stati sono particolarmente utili per modellare sistemi complessi in cui il comportamento dipende fortemente dalla sequenza di eventi. Sono spesso utilizzati per descrivere il comportamento di interfacce utente, sistemi di controllo, protocolli di comunicazione e qualsiasi altro sistema in cui il comportamento è reattivo e orientato agli eventi. Analizziamo nello specifico i cambiamenti di stato delle classi: Admin, Feedback, Navigation e Checkpoint. Le istanze della classe Admin possono trovarsi in due soli stati: NotAuthenticated e Authenticated. L'oggetto inizia nello stato NotAuthenticated e, dopo il controllo delle credenziali (CredentialCheck), passa allo stato Authenticated, a condizione che la condizione "isAuthenticated == true" sia soddisfatta.

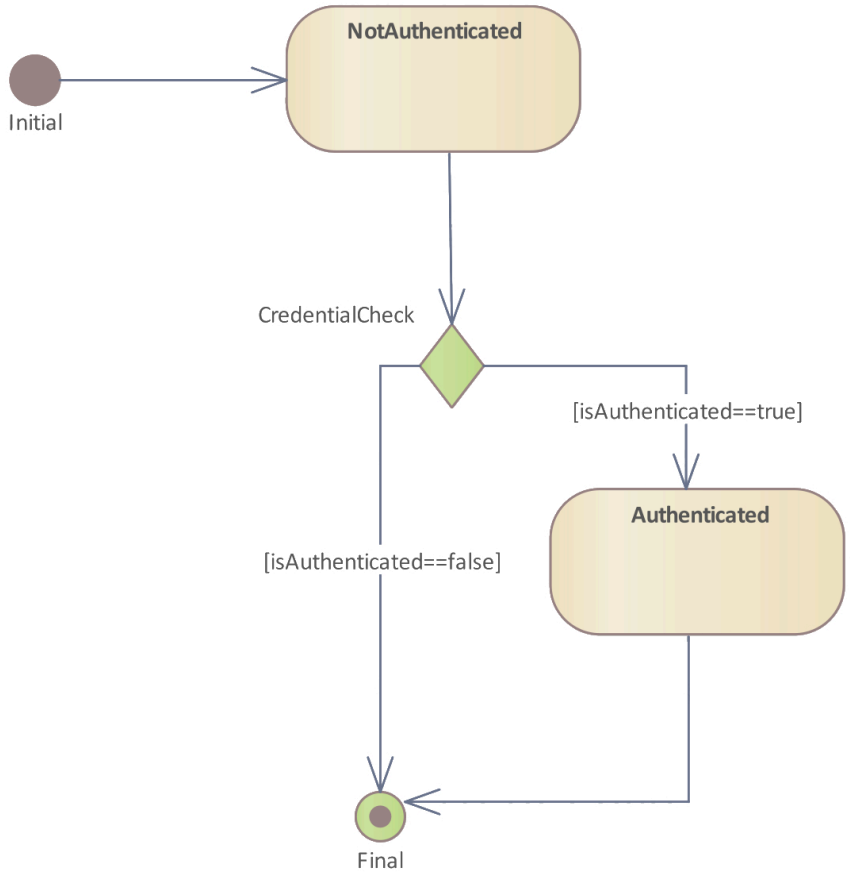


**Figura 2.17:** Diagramma di Stato Admin

Per assicurare una gestione efficiente delle segnalazioni degli utenti, la classe Feedback è stata progettata per rappresentare i diversi stati che un feedback può assumere nel corso del suo ciclo di vita. Inizialmente, l'istanza è contrassegnata come Inserted. Successivamente, dopo la classificazione effettuata dal sistema, il feedback può transitare in uno dei seguenti stati: Positive, Negative, o Urgent. Nel caso di stato Positive, il messaggio contiene osservazioni favorevoli riguardo al sistema. Se lo stato è Negative, il feedback include critiche o informazioni negative, richiedendo l'intervento di un operatore per verificare e risolvere la situazione. Quando il sistema identifica il feedback come Urgent, ciò indica la presenza probabile di un problema critico che necessita di un intervento immediato per garantire il corretto funzionamento del sistema.

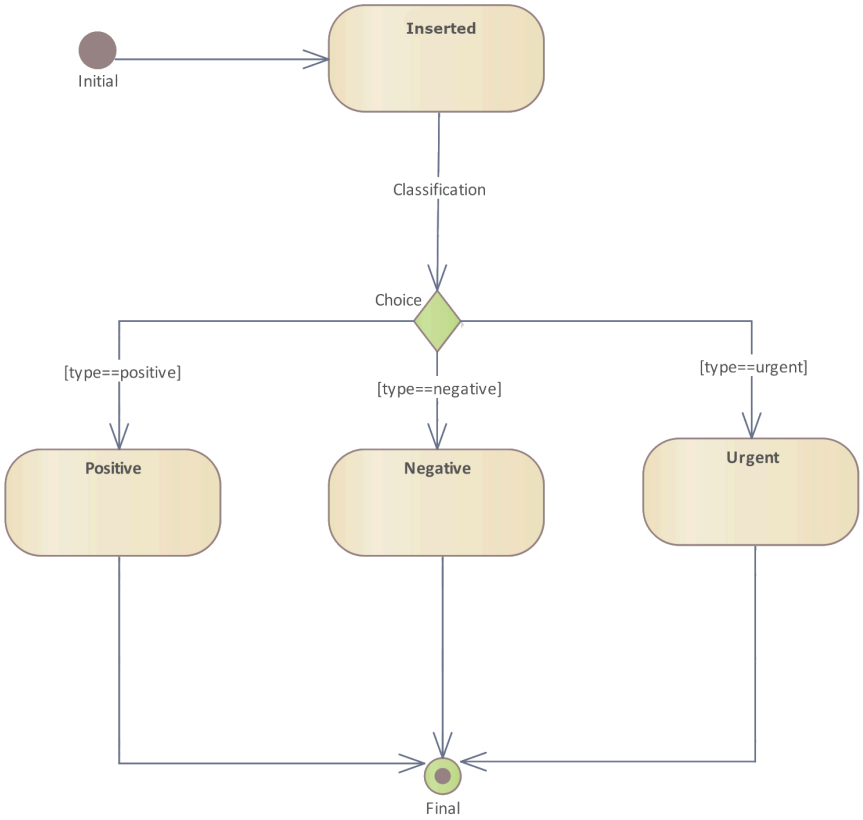


**Figura 2.18:** Diagramma di Stato Feedback

La classe Navigation è la componente più centrale del sistema, poiché sovrintende l'intero processo di navigazione e gestisce le relative transizioni di stato. Progettata per riflettere le diverse fasi del processo di navigazione, questa classe consente al sistema di adattarsi dinamicamente alle azioni dell'utente e agli eventi che si verificano durante l'uso. Attraverso i suoi stati, la classe Navigation coordina la selezione dei percorsi, l'interazione con i punti di interesse, e l'aggiornamento in tempo reale delle informazioni, assicurando un'esperienza utente fluida ed efficiente. La gestione degli stati all'interno di Navigation è cruciale per il funzionamento ottimale del sistema, in quanto permette di monitorare e rispondere tempestivamente ai cambiamenti nelle condizioni operative. L'istanza di Navigation inizia nello stato Idle. Quando vengono selezionate sia la Source che il Target, la classe passa allo stato Preparation, in attesa che venga caricata la rotta. Una volta caricato il percorso, la navigazione viene avviata, portando la classe allo stato Started. Da questo punto, può transitare direttamente allo stato Terminated se la navigazione viene interrotta forzatamente, oppure allo stato Progress quando la posizione dell'utente cambia. Resterà in Progress fino a raggiungere la destinazione o fino a quando l'utente decide di terminare anticipatamente la navigazione.

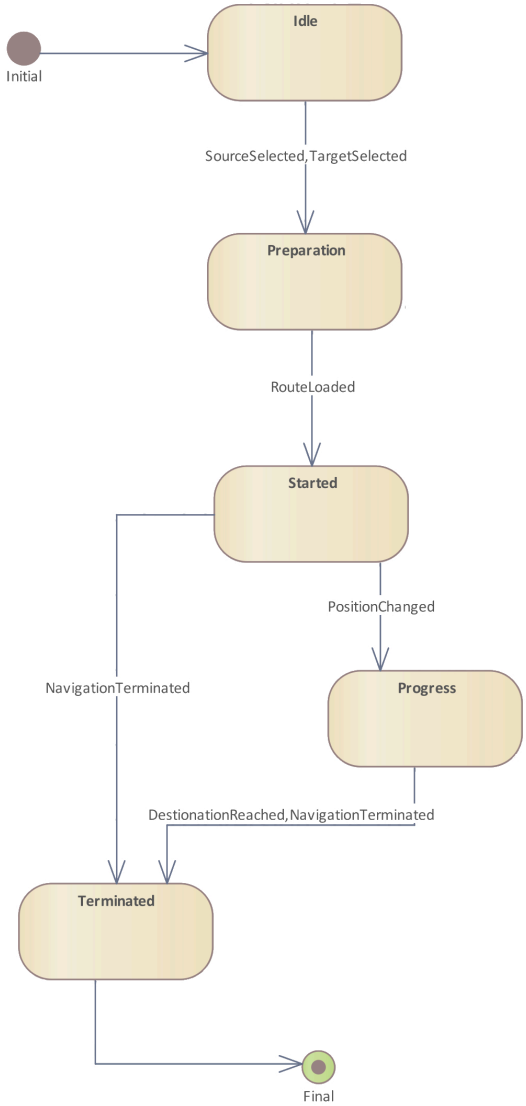


**Figura 2.19:** Diagramma di Stato Navigation

Lo stato della classe Checkpoint svolge un ruolo cruciale nel supportare la classe Navigation, garantendo il monitoraggio accurato dei progressi lungo il percorso definito. Inizialmente, tutti i checkpoint sono nello stato NotPassed, indicando che non sono ancora stati raggiunti. Quando l'utente si avvicina a un checkpoint e lo supera con successo, lo stato della relativa istanza di Checkpoint cambia a Passed, segnalando che il punto è stato correttamente attraversato. Nel caso in cui la navigazione venga terminata anticipatamente il checkpoint rimane nello stato di NotPassed.

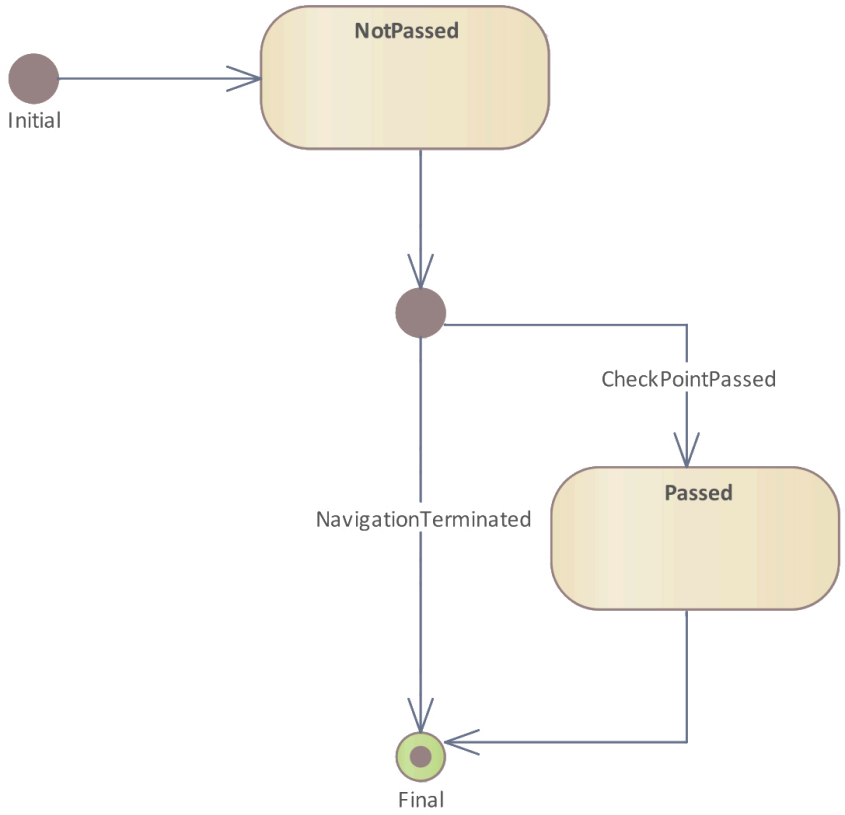


**Figura 2.20:** Diagramma di Stato Checkpoint

**Diagramma dei componenti**

I diagrammi dei componenti sono strumenti essenziali nella modellazione architetturale dei sistemi software. Questi diagrammi forniscono una rappresentazione visiva dell'architettura fisica del sistema, evidenziando le principali componenti software, le loro interazioni e le dipendenze tra di esse. Ogni componente in un diagramma delle componenti rappresenta una parte modulare e riutilizzabile del sistema, come un modulo, una libreria o un servizio, che può essere implementato e distribuito indipendentemente dagli altri. I componenti sono collegati tra loro attraverso interfacce e connessioni, che descrivono come le diverse parti del sistema comunicano e collaborano per fornire funzionalità complete. Questi diagrammi sono particolarmente utili per comprendere la struttura del sistema, evidenziando come le varie componenti si integrano per formare l'intera applicazione. Essi sono utilizzati per pianificare l'implementazione e la distribuzione del software, garantendo che ogni componente abbia un ruolo ben definito e che le interazioni tra le componenti siano chiare e coerenti.

Il componente Controller rappresenta il nucleo centrale dell'architettura, gestendo l'interazione tra l'utente e il sistema. Al suo interno, sono presenti tre controller specifici: NavigationController, InfoController e AdminController, ciascuno incaricato di delegare compiti a moduli specializzati, assicurando una gestione efficiente delle operazioni.

- Il NavigationController interagisce direttamente con il modulo Navigazione, responsabile della logica di navigazione. Questo modulo si occupa di operazioni cruciali come il calcolo dei percorsi e il recupero di informazioni relative a edifici, piani e punti di interesse. La comunicazione tra i due garantisce che il NavigationController possa richiedere i dati necessari per offrire all'utente un'esperienza di navigazione fluida e precisa. Il modulo Navigazione, a sua volta, coordina le attività con sottocomponenti come NavigationService e NavigationRepository, nonché altri moduli come Building, Floor e InterestPoint, assicurando una gestione completa ed efficiente dei processi di navigazione interna.

- Il InfoController è strettamente collegato al modulo Information, che gestisce componenti fondamentali come Programs, InfoRepository e Feedback. Questa interazione permette al sistema di gestire la raccolta e l'archiviazione di informazioni critiche, come i programmi o i feedback degli utenti. Grazie a questa connessione, i dati vengono continuamente aggiornati e resi disponibili, assicurando una corretta sincronizzazione delle informazioni all'interno del sistema.

- Il InfoController è strettamente collegato al modulo Information, che gestisce componenti fondamentali come Programs, InfoRepository e Feedback. Questa interazione permette al sistema di gestire la raccolta e l'archiviazione di informazioni critiche, come i programmi o i feedback degli utenti. Grazie a questa connessione, i dati vengono continuamente aggiornati e resi disponibili, assicurando una corretta sincronizzazione delle informazioni all'interno del sistema.

In sintesi, il componente Controller funge da ponte tra l'utente e i vari moduli del sistema, coordinando l'interazione e garantendo che le operazioni siano eseguite in maniera fluida e responsiva, grazie a una precisa delega delle responsabilità a ciascun modulo specializzato.

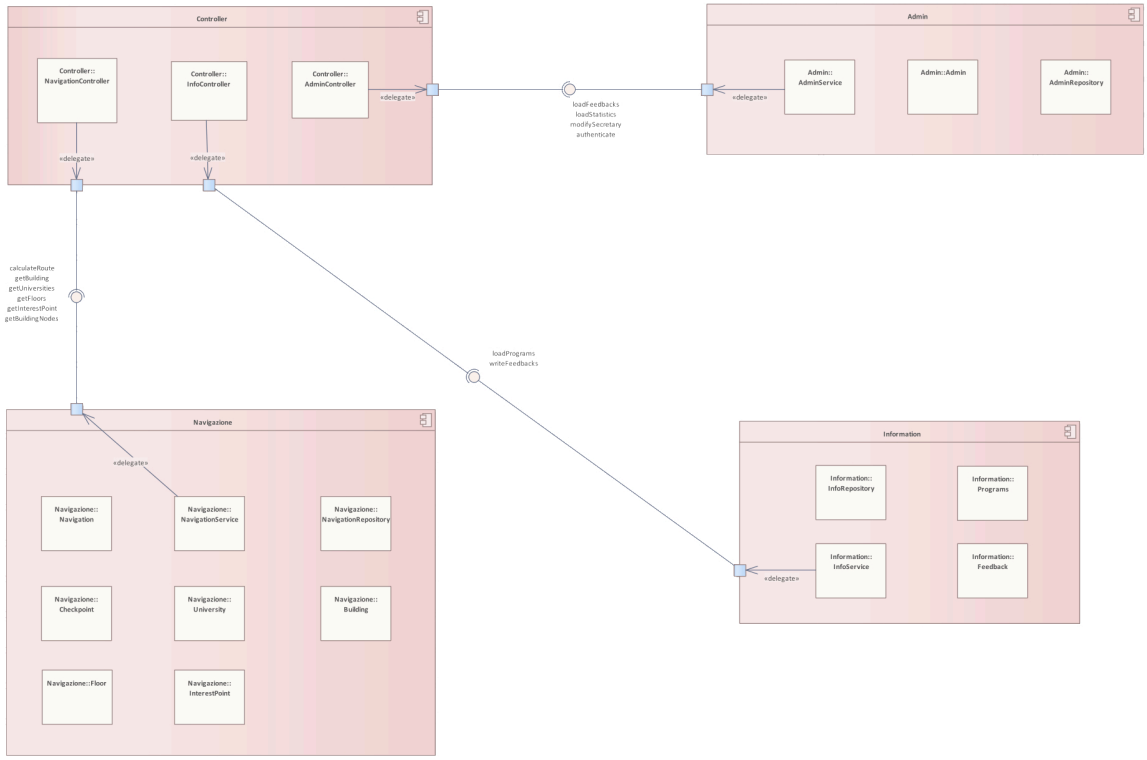


**Figura 2.21:** Diagramma dei Componenti

## 2.5 Architettura del sistema software

L'architettura software illustrata nella figura rappresenta un sistema moderno e flessibile, progettato per supportare sia applicazioni web che mobile. Al centro di questa architettura c'è un insieme di componenti che lavorano insieme per offrire un'esperienza utente fluida e sicura.

Il sistema si basa su un'interfaccia web destinata agli amministratori e su un'app mobile per gli utenti finali. Gli amministratori utilizzano un'applicazione web, costruita con tecnologie moderne quali Angular, per gestire e monitorare il sistema. Gli utenti finali, invece, accedono al servizio tramite un'app mobile, sviluppata in Unity, per supportare le funzionalità di realtà aumentata in modo efficiente, sia per Android che per iOS, offrendo un accesso facile e intuitivo alle funzionalità del sistema. Entrambe le applicazioni comunicano con il back-end attraverso un'API REST. Questa API, ospitata su un ambiente Linux, funge da ponte tra il front-end e il database, gestendo le richieste degli utenti e assicurando che le informazioni siano scambiate in modo efficiente e sicuro. Il fatto che l'API sia basata su REST la rende versatile e facilmente integrabile con altri sistemi, un aspetto cruciale in architetture moderne.

Il sistema utilizza un database NoSQL per la memorizzazione dei dati. Questo tipo di database è scelto per la sua capacità di gestire grandi quantità di dati in modo scalabile e flessibile. È particolarmente utile in contesti in cui i dati possono essere non strutturati o semi-strutturati e dove è necessaria una scalabilità orizzontale per gestire carichi di lavoro variabili.

Un altro componente essenziale dell'architettura è il server di identità, che gestisce l'autenticazione e l'autorizzazione degli amministratori. Questo server assicura che solo gli utenti autorizzati possano accedere al sistema, utilizzando protocolli standard come OAuth2 o OpenID Connect per garantire la sicurezza delle transazioni.

Infine, l'integrazione con Azure DevOps evidenzia l'importanza di un flusso di lavoro DevOps automatizzato e continuo. Utilizzando Azure DevOps, il team di sviluppo può gestire il codice sorgente, implementare pipeline di integrazione e distribuzione continua (CI/CD), e mantenere un ciclo di sviluppo agile e reattivo. Questo approccio consente di rilasciare nuove funzionalità e aggiornamenti in modo rapido ed efficiente, mantenendo al contempo elevati standard di qualità e sicurezza.

In sintesi, questa architettura è progettata per essere scalabile, sicura e altamente integrata, rispondendo alle esigenze di un ambiente digitale moderno e in continua evoluzione.

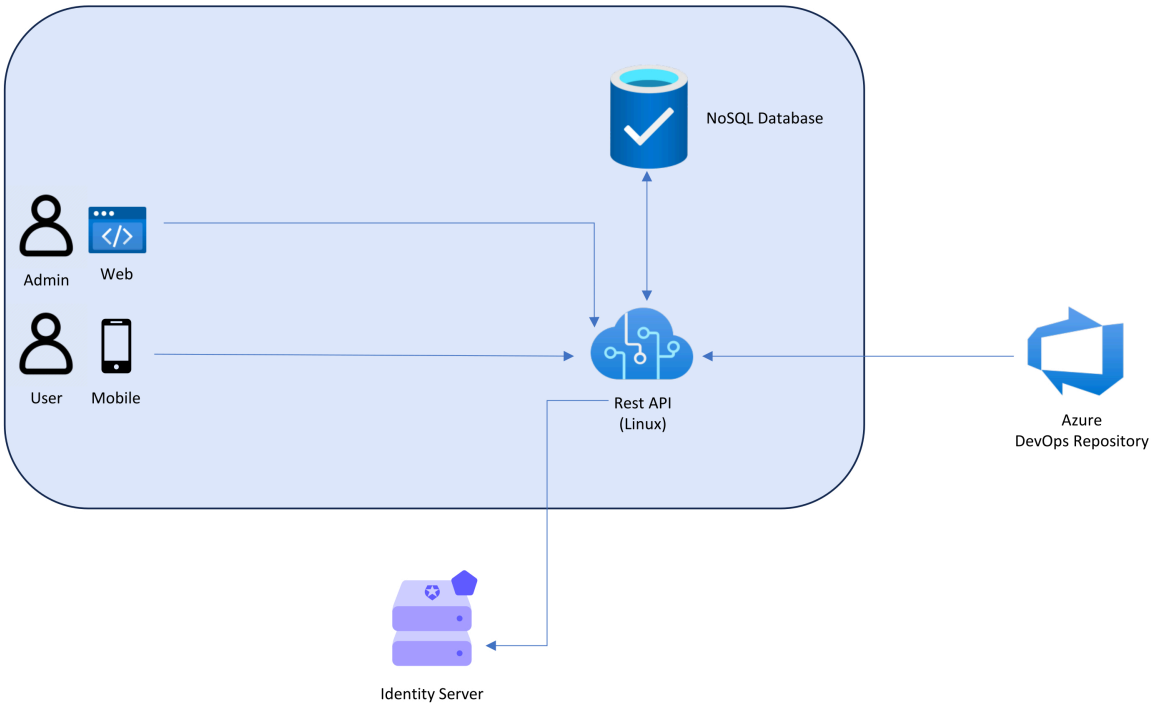


**Figura 2.22:** Architettura del sistema

CAPITOLO 3

# Calcolo del percorso

*In questo capitolo l'attenzione si concentra sul problema del calcolo del percorso, analizzando le sfide legate alla navigazione all'interno di reti complesse. Vengono descritte le tecnologie utilizzate per affrontare tale problematica, con particolare riferimento all'implementazione delle Graph Neural Networks (GNN) e alla loro applicazione specifica nel progetto NavigaUnivpm.*

## 3.1 Il calcolo del percorso

Il calcolo del percorso nell'applicazione NavigaUnivpm rappresenta uno degli elementi cardine, fondamentale per garantire agli utenti una navigazione fluida e senza errori all'interno del campus universitario. L'obiettivo principale di questo processo è guidare lo studente o il visitatore verso la destinazione desiderata nel minor tempo possibile, fornendo indicazioni dettagliate e precise attraverso un'interfaccia intuitiva. Alla base di questo calcolo vi è la costruzione di un grafo, in cui ogni edificio del campus viene modellato come un insieme di nodi e archi. I nodi corrispondono ai punti di interesse[1], mentre gli archi rappresentano i corridoi e le connessioni tra questi punti. Ogni nodo è associato a una serie di metadati che possono includere informazioni come la sua esatta posizione geografica, la funzione del punto e la sua accessibilità. Il calcolo del percorso nell'applicazione NavigaUnivpm si fonda su algoritmi di ricerca del cammino minimo, come l'algoritmo di Dijkstra, che garantisce la determinazione del percorso ottimale tra un punto di partenza e una destinazione finale. L'algoritmo di Dijkstra è particolarmente adatto per reti con pesi non negativi, come quelle rappresentate dai percorsi all'interno degli edifici universitari. Il funzionamento si basa su una tecnica chiamata "greedy", ovvero una strategia in cui, ad ogni passo, si seleziona il nodo con la distanza cumulativa minore rispetto alla sorgente. L'algoritmo inizia assegnando al nodo di partenza una distanza di zero, mentre tutti gli altri nodi hanno una distanza infinita. Man mano che si esplorano i nodi, l'algoritmo aggiorna le distanze per ogni nodo adiacente, assicurando che, una volta visitato un nodo, il percorso più breve verso di esso sia stato trovato. Il processo si ripete fino a quando non vengono visitati tutti i nodi o fino a quando non si è raggiunta la destinazione desiderata. Questo lo rende particolarmente adatto in contesti statici, come la navigazione all'interno di edifici o strutture dove le distanze e i collegamenti tra i nodi (ad esempio stanze, aule o altri punti di interesse) non cambiano frequentemente. Un altro aspetto rilevante è l'integrazione delle informazioni in tempo reale. Grazie a un continuo scambio di dati tra il dispositivo mobile dell'utente e il server centrale, il

[1] Punto di interesse: indichiamo così le aule, gli uffici e tutti i locali di interesse nel contesto della navigazione.

sistema è in grado di aggiornare il percorso in base a variazioni come chiusure temporanee di aree, eventi, o sovraffollamenti, migliorando l'esperienza complessiva di navigazione. Tutto ciò rende il calcolo del percorso non solo un semplice strumento di navigazione, ma una soluzione integrata per facilitare la mobilità all'interno del campus universitario, offrendo un supporto concreto a tutti i tipi di utenti.

## 3.2 Descrizione del problema

L'algoritmo di Dijkstra, pur essendo uno degli approcci più noti e utilizzati per il calcolo del percorso in grafi pesati, presenta alcuni limiti significativi, soprattutto quando viene applicato a sistemi complessi e ad alta densità di traffico, come quello universitario.

Un primo problema è legato alla sua complessità computazionale. L'algoritmo richiede di analizzare tutti i nodi del grafo per garantire di trovare il percorso più breve, il che significa che la sua efficienza peggiora significativamente all'aumentare della dimensione del grafo stesso. In ambienti come un campus universitario, che può comprendere molti edifici, piani e punti di interesse, questo approccio diventa inefficiente, in particolare se gli utenti interagiscono con il sistema in modo continuo. Di conseguenza, quando le richieste di calcolo del percorso aumentano, la latenza nell'elaborazione delle risposte diventa un ostacolo alla fruibilità del servizio. L'algoritmo di Dijkstra, inoltre, tende a sovraccaricare l'API del sistema quando vi sono richieste multiple e contemporanee. Ogni richiesta richiede un'interazione con il database per recuperare le informazioni necessarie, come la posizione degli edifici, i punti di interesse e la topografia degli spazi interni. Se un gran numero di utenti richiede simultaneamente percorsi diversi, si può verificare un collo di bottiglia nelle prestazioni del server, con una conseguente degradazione dell'esperienza utente.

## 3.3 Soluzione proposta

La soluzione proposta per superare i limiti dell'algoritmo di Dijkstra e migliorare l'efficienza del calcolo dei percorsi nell'applicazione NavigaUnivpm si basa sull'adozione di una Graph Neural Network (GNN). Questa tecnologia, adattata per lavorare su dati strutturati in forma di grafo, è particolarmente indicata per problemi complessi legati alla navigazione e all'analisi dei percorsi in ambienti articolati come il campus universitario.

In un'applicazione come NavigaUnivpm, dove gli utenti richiedono risposte in tempo reale, la rapidità nel calcolo dei percorsi è cruciale. Una volta addestrata la GNN, l'inferenza avviene in tempi estremamente rapidi. La rete è in grado di identificare quasi immediatamente il percorso più efficiente, considerando vari fattori come distanza, accessibilità e congestione pedonale. Questo permette al sistema di rispondere prontamente alle richieste, evitando lunghi tempi di attesa e garantendo che i percorsi siano sempre aggiornati rispetto alle condizioni reali.

Il grafo, una volta generato, viene ricalcolato solo in caso di modifiche da parte dell'amministratore, come l'aggiunta o la rimozione di aule, corridoi o chiusure temporanee. Questo approccio ottimizza l'efficienza del sistema, riducendo drasticamente il carico computazionale durante l'utilizzo quotidiano e consentendo al sistema di gestire un elevato numero di richieste senza sovraccaricare le risorse.

L'architettura della GNN adottata in NavigaUnivpm si basa su un notebook[2] di Ten-

[2]I notebook sono costituiti da una serie di celle, ognuna delle quali può contenere codice eseguibile, testo descrittivo, equazioni, visualizzazioni e altro ancora.

sorFlow[3] progettato per risolvere problemi di cammino minimo su grafi. Questo approccio utilizza un'architettura Encode-Process-Decode, che sfrutta il message passing tra nodi e archi per apprendere le relazioni interne al grafo e predire il percorso ottimale. Nel contesto dell'applicazione, i nodi rappresentano punti di interesse, come aule e uffici, mentre gli archi corrispondono ai percorsi fisici, come corridoi e scale.

Grazie all'adozione di questo framework, NavigaUnivpm è in grado di gestire percorsi dinamici in tempo reale con un'elevata velocità di inferenza e un'accuratezza predittiva migliorata, ottimizzando così l'efficienza del calcolo dei percorsi e migliorando l'esperienza utente complessiva.

## 3.4 Tecnologie

Prima di approfondire l'implementazione della soluzione, a partire dalla creazione del modello fino all'integrazione nel sistema, introduciamo le tecnologie utilizzate.

### 3.4.1 Tensorflow

TensorFlow è una delle tecnologie di machine learning più avanzate e largamente utilizzate per lo sviluppo di modelli di intelligenza artificiale, e si dimostra particolarmente adatta anche per l'implementazione di Graph Neural Networks (GNN). Creata da Google, TensorFlow è una piattaforma open-source che fornisce un'ampia gamma di strumenti per costruire, addestrare e distribuire modelli di deep learning su larga scala. TensorFlow è progettato per gestire grandi quantità di dati e modelli complessi, il che lo rende particolarmente adatto a un'applicazione come NavigaUnivpm, che deve servire numerosi utenti in tempo reale. La sua capacità di addestrare modelli di deep learning su infrastrutture distribuite consente di scalare facilmente il sistema per rispondere all'aumento delle richieste, mantenendo elevate prestazioni.

Grazie alla libreria TF-GNN, TensorFlow offre componenti predefiniti per costruire e addestrare modelli su grafi, semplificando lo sviluppo. Questa integrazione permette agli sviluppatori di concentrarsi sull'ottimizzazione e sull'applicazione pratica dei modelli, senza dover gestire aspetti tecnici complessi. Con TensorFlow è possibile addestrare modelli per risolvere il problema del cammino minimo su grafi, utilizzando pipeline di supervised learning. Ciò garantisce un continuo miglioramento della precisione predittiva, rendendo il sistema di navigazione capace di offrire percorsi ottimali in modo rapido e affidabile.

Infine, una volta completato l'addestramento, il modello può sfruttare l'inferenza in tempo reale, garantendo risposte immediate e precise agli utenti. Questa caratteristica è fondamentale per un'applicazione come NavigaUnivpm, dove l'efficienza nella risposta è essenziale per un'esperienza utente di qualità.

### 3.4.2 Firebase

Firebase è una piattaforma completa per lo sviluppo di applicazioni mobili e web, fornita da Google, che offre una serie di strumenti e servizi integrati per la gestione dei dati, l'autenticazione, le notifiche push e l'analisi delle prestazioni. La sua semplicità d'uso e la capacità di scalare automaticamente la rendono ideale per applicazioni che richiedono aggiornamenti in tempo reale e un'esperienza utente fluida. Un componente chiave di Firebase è il suo Realtime Database, una soluzione NoSQL progettata per sincronizzare i dati tra il server e i dispositivi degli utenti in modo immediato, garantendo che le informazioni siano sempre

[3] https://github.com/tensorflow/gnn/blob/main/examples/notebooks/graph_network_shortest_path.ipynb

aggiornate e accessibili. Essendo un database NoSQL, il Firebase Realtime Database offre una grande flessibilità nella gestione e nel salvataggio dei dati. A differenza dei database relazionali tradizionali, che richiedono una struttura rigida con tabelle e schemi predefiniti, Firebase consente di memorizzare i dati in una struttura gerarchica simile a un albero JSON. Questo approccio permette di adattare facilmente la struttura dei dati alle esigenze dell'applicazione, senza dover rispettare vincoli rigidi di schema. Questa flessibilità facilita lo sviluppo e l'aggiornamento del sistema, rendendo il salvataggio dei dati più semplice e adattabile a esigenze in continua evoluzione. Un altro vantaggio chiave di Firebase è la sua capacità di scalare automaticamente in base alle esigenze dell'applicazione. Man mano che cresce il numero di utenti o la quantità di dati gestiti, il sistema si adatta automaticamente senza richiedere interventi complessi. Inoltre, Firebase supporta l'accesso offline, consentendo agli utenti di continuare a utilizzare l'app anche in aree con connessione limitata. I dati vengono memorizzati localmente sui dispositivi e sincronizzati una volta che la connessione è ristabilita.

### 3.4.3 Python

Python è considerato uno dei linguaggi di programmazione più adatti per lo sviluppo di modelli di intelligenza artificiale (AI) e machine learning, grazie alla sua semplicità, flessibilità e alla ricchezza del suo ecosistema. La sintassi chiara e leggibile di Python consente agli sviluppatori di scrivere codice in modo più efficiente e di concentrarsi sulla logica del problema piuttosto che su complessità sintattiche. Questa caratteristica è particolarmente vantaggiosa nel contesto dell'AI, dove la prototipazione rapida e la sperimentazione sono essenziali.

Uno dei principali vantaggi di Python è la disponibilità di un'ampia gamma di librerie e framework specializzati. Librerie come TensorFlow, sviluppata da Google, e PyTorch, sviluppata da Facebook, offrono strumenti avanzati per la costruzione, l'addestramento e l'ottimizzazione di reti neurali complesse. Keras, un'interfaccia di alto livello per TensorFlow, semplifica ulteriormente la creazione di modelli deep learning, mentre scikit-learn fornisce algoritmi robusti per l'apprendimento automatico tradizionale.

Python non solo supporta il machine learning e il deep learning, ma è anche eccellente per la manipolazione e l'analisi dei dati grazie a librerie come Pandas e NumPy. Questi strumenti facilitano la preparazione dei dati, una fase cruciale nella costruzione di modelli AI, permettendo agli sviluppatori di gestire e trasformare grandi quantità di dati con facilità. Inoltre, Python integra bene con altre tecnologie e strumenti, come i database e le piattaforme di cloud computing, rendendolo una scelta versatile per progetti complessi e su larga scala.

La comunità di Python è un altro elemento chiave del suo successo nel campo dell'AI. Con un vasto numero di sviluppatori e ricercatori che contribuiscono costantemente a nuovi strumenti e librerie, Python beneficia di aggiornamenti frequenti e del supporto per le ultime innovazioni tecnologiche. Questo ambiente collaborativo e dinamico accelera il progresso nella ricerca e nello sviluppo di soluzioni di intelligenza artificiale, consolidando ulteriormente il ruolo di Python come linguaggio preferito per l'AI.

## 3.5 Sviluppo del modello

A partire dalla documentazione fornita da Tensorflow è stato sviluppato il modello di Graph Neural Network adattato alle esigenze del sistema software sviluppato. Il nucleo del nostro modello è rappresentato dal Layer "EncodeProcessDecode", il quale è una classe che implementa quanto Sanchez-Gonzalez *et al.* [2020] hanno sperimentato. Come suggerito dal nome, la classe è composta da tre componenti principali: un encoder, uno o più processori e un

decoder. Il modello accetta come input set di nodi, set di archi e un contesto globale, ciascuno dei quali contiene uno stato nascosto iniziale associato alla funzione "tfgnn.HIDDEN_STATE". Il modello esegue quindi una serie di passi di elaborazione del grafo utilizzando il message passing per aggiornare gli stati dei nodi, degli archi e del contesto, e infine decodifica le rappresentazioni ottenute per produrre le uscite desiderate. Prima di analizzare i componenti descriviamo i parametri necessari da fornire in input al layer:

```
class EncodeProcessDecode(tf.keras.layers.Layer):
    def __init__(
      self,
      edge_output_size: Optional[int],
      node_output_size: Optional[int],
      context_output_size: Optional[int],
      num_message_passing_steps: int,
      num_mlp_hidden_layers: int,
      mlp_hidden_size: int,
      latent_size: int,
      use_layer_norm: bool,
      shared_processors: bool,
      reduce_type_to_nodes: str = "sum",
      reduce_type_to_context: str = "sum",
      name: str = "encode_process_decode"):
        super().__init__(name=name)
```

- `edge_output_size`, `node_output_size` e `context_output_size` definiscono rispettivamente la dimensione dell'output per quanto riguardi gli archi, i nodi ed il contesto globale del grafo. Nel caso sia None non verrà generato un output per quell'elemento.

- `num_message_passing_steps`: Indica il numero di iterazioni di "message passing" all'interno del grafo. Durante ciascun passo, le informazioni vengono scambiate tra nodi e archi, consentendo agli stati dei nodi di essere aggiornati sulla base delle informazioni provenienti dai nodi e archi vicini. Un numero maggiore di passi permette di propagare le informazioni attraverso distanze più lunghe nel grafo.

- `num_mlp_hidden_layers` e `mlp_hidden_size` indicano rispettivamente il numero di strati nascosti nel Multi-Layer Perceptron (MLP) e la dimensione di ciascuno degli strati nascosti dell'MLP. Il numero di strati nascosti influisce sulla capacità espressiva dell'MLP: più strati possono catturare relazioni più complesse ed una dimensione maggiore permette di catturare più informazioni e rappresentazioni più ricche. Una dimensione non corretta potrebbe destabilizzare il modello, sia nel caso abbia poche informazioni sia nel caso ne abbia in eccesso.

- use_layer_norm definisce se applicare la normalizzazione a livello di strato (Layer Normalization) durante la fase di encoding e processing che aiuta a stabilizzare il processo di addestramento e può migliorare la convergenza del modello; mentre shared_processors determina se i processori utilizzati nei diversi passi di message passing condividono o meno i pesi. Se True, lo stesso processore viene riutilizzato per tutti i passi; se False, ogni passo ha un processore diverso con pesi distinti. L'uso di processori condivisi riduce il numero di parametri nel modello, ma può limitare la capacità di apprendere dinamiche più complesse durante il message passing.

- `reduce_type_to_nodes` specifica il tipo di operazione di riduzione da applicare quando si aggregano le informazioni dagli archi verso i nodi. Il valore predefinito è "sum", il che significa che le informazioni provenienti dagli archi in entrata vengono sommate. Altri possibili valori potrebbero essere "mean" o "max", che rispettivamente calcolano la media o il massimo delle informazioni provenienti dagli archi.; mentre `reduce_type_to_context` in modo analogo specifica il tipo di operazione per la riduzione delle informazioni dai nodi e dagli archi verso il contesto globale del grafo.

```
# Build graph encoder.
def encoder_fn(graph_piece, *, edge_set_name=None, node_set_name=None):
  piece_name = (f"edges_{edge_set_name}" if edge_set_name else
                f"nodes_{node_set_name}" if node_set_name else "context")
  mlp = build_mlp(num_hidden_layers=num_mlp_hidden_layers,
                  hidden_size=mlp_hidden_size,
                  output_size=latent_size,
                  use_layer_norm=use_layer_norm,
                  name=f"{self.name}/encoder/{piece_name}")
  return mlp(graph_piece[tfgnn.HIDDEN_STATE])

self._encoder = tfgnn.keras.layers.MapFeatures(
    edge_sets_fn=encoder_fn,
    node_sets_fn=encoder_fn,
    context_fn=encoder_fn)
```

L'encoder è responsabile di trasformare lo stato nascosto iniziale di ciascun nodo, arco e contesto in una rappresentazione latente, utilizzando un multi-layer perceptron (MLP). La funzione `encoder_fn` è utilizzata per costruire questi MLP in modo parametrico, dove ogni componente (nodi, archi o contesto) viene elaborato separatamente. La struttura dell'encoder è implementata con `tfgnn.keras.layers.MapFeatures`, che permette di applicare la funzione di encoding su nodi, archi e contesto in modo modulare.

```
def processor_fn(*, processor_name, edge_set_name=None,
                 node_set_name=None):
  if edge_set_name is not None:
    mlp_name = f"{processor_name}/edges_{edge_set_name}"
  elif node_set_name is not None:
    mlp_name = f"{processor_name}/nodes_{node_set_name}"
  else:
    mlp_name = f"{processor_name}/context"
  mlp = build_mlp(name=mlp_name,
                  num_hidden_layers=num_mlp_hidden_layers,
                  hidden_size=mlp_hidden_size,
                  output_size=latent_size,
                  use_layer_norm=use_layer_norm)
  return tfgnn.keras.layers.NextStateFromConcat(mlp)

num_processors = (1 if shared_processors else
                                        num_message_passing_steps)

processors = []
for processor_i in range(num_processors):
```

```
    processor_name = f"{self.name}/processor_{processor_i}"
    processor_fn_named = functools.partial(processor_fn,
                                           processor_name=processor_name)
    processors.append(GraphNetworkGraphUpdate(
        edges_next_state_factory=processor_fn_named,
        nodes_next_state_factory=processor_fn_named,
        context_next_state_factory=processor_fn_named,
        reduce_type_to_nodes=reduce_type_to_nodes,
        reduce_type_to_context=reduce_type_to_context,
        name=processor_name))

if shared_processors:
  self._processors = processors * num_message_passing_steps
else:
  self._processors = processors
```

Il cuore del modello risiede nei processori, che implementano la logica di message passing, ossia il meccanismo attraverso il quale i nodi e gli archi comunicano tra loro e aggiornano i loro stati. Per ciascun nodo o arco, gli input sono concatenati e passati attraverso un MLP che calcola il nuovo stato. I processori possono essere condivisi o meno tra i diversi passi di elaborazione del grafo, a seconda della configurazione fornita dall'utente tramite il parametro `shared_processors`. I processori vengono costruiti attraverso la funzione `processor_fn`, che utilizza il meccanismo NextStateFromConcat di TensorFlow GNN per concatenare gli input e aggiornare gli stati. La fase di message passing viene eseguita per un numero specificato di iterazioni (`num_message_passing_steps`), il che consente al modello di propagare informazioni a lungo raggio all'interno del grafo.

```
# Build graph decoder.
def decoder_fn(graph_piece, *, edge_set_name=None, node_set_name=None):
  piece_name = (f"edges_{edge_set_name}" if edge_set_name else
                f"nodes_{node_set_name}" if node_set_name else "context")
  if edge_set_name:
    output_size = edge_output_size
  elif node_set_name:
    output_size = node_output_size
  else:
    output_size = context_output_size
  mlp = build_mlp(num_hidden_layers=num_mlp_hidden_layers,
                  hidden_size=mlp_hidden_size,
                  output_size=output_size,
                  use_layer_norm=False,
                  name=f"{self.name}/decoder/{piece_name}")
  return mlp(graph_piece[tfgnn.HIDDEN_STATE])

self._decoder = tfgnn.keras.layers.MapFeatures(
    edge_sets_fn=decoder_fn if edge_output_size else None,
    node_sets_fn=decoder_fn if node_output_size else None,
    context_fn=decoder_fn if context_output_size else None)
```

Dopo il processamento, il decoder trasforma le rappresentazioni latenti dei nodi, archi e contesto in output finali utilizzando un MLP. Simile all'encoder, il decoder è specifico per

ciascuno dei componenti (nodi, archi o contesto), e applica un MLP per produrre gli output finali con dimensioni definite dai parametri `edge_output_size`, `node_output_size` e `context_output_size`. Il decoder è progettato per non utilizzare la normalizzazione a livello di layer (LayerNorm), poiché non è desiderata per gli output finali.

Per gestire al meglio il salvataggio del modello è stata introdotta la classa `ShortestPathModel` che estende `tf.keras.Model` e aggiunge al suo interno solamente il Layer definito in precedenza che viene eseguito attraverso la primitiva `call`.

```
class ShortestPathModel(tf.keras.Model):
  def __init__(self, edge_output_size, node_output_size,
                context_output_size, num_message_passing_steps,
                num_mlp_hidden_layers, mlp_hidden_size,
                latent_size, use_layer_norm, shared_processors):
      super(ShortestPathModel, self).__init__()

      self.encode_process_decode = EncodeProcessDecode(
          edge_output_size=edge_output_size,
          node_output_size=node_output_size,
          context_output_size=context_output_size,
          num_message_passing_steps=num_message_passing_steps,
          num_mlp_hidden_layers=num_mlp_hidden_layers,
          mlp_hidden_size=mlp_hidden_size,
          latent_size=latent_size,
          use_layer_norm=use_layer_norm,
          shared_processors=shared_processors
      )

  def call(self, inputs):
    return self.encode_process_decode(inputs)
```

Il salvataggio viene eseguito sfruttando le funzioni messi a disposizione da `tf.Keras.Model`, quali `load_weights` e `save_weights`, per andare a salvare e ricaricare solamente i pesi della rete neurale.

```
trainable_gnn = ShortestPathModel(
      edge_output_size=2,
      node_output_size=2,
      context_output_size=None,
      num_message_passing_steps=2,
      num_mlp_hidden_layers=2,
      mlp_hidden_size=64,
      latent_size=64,
      use_layer_norm=True,
      shared_processors=True,
      )
  try:
    trainable_gnn.load_weights("weightsAllConfig/")
  except:
    print("Weights not found")
  optimizer = tf.keras.optimizers.Adam(
      learning_rate=tf.keras.optimizers.schedules.ExponentialDecay(
          initial_learning_rate=1e-3,
```

```
            decay_steps=1500,
            decay_rate=0.1),
        )
...
trainable_gnn.save_weights("weightsAllConfig/")
```

La scelta dei parametri per il modello è stata fatta con attenzione per bilanciare la complessità del modello con la sua capacità di apprendere e generalizzare in modo efficace. Per la classificazione degli archi e dei nodi, è stato scelto di usare una dimensione di output pari a 2 sia per gli archi (`edge_output_size=2`) che per i nodi (`node_output_size=2`), poiché l'obiettivo è una classificazione binaria. Non è stato necessario produrre un output per il contesto globale (`context_output_size=None`) poiché in questo specifico caso il contesto non aggiunge valore all'output desiderato. Per quanto riguarda i passi di message passing, è stato scelto di usare due iterazioni (`num_message_passing_steps=2`). Questo numero consente di propagare le informazioni nel grafo fino ai vicini di secondo ordine, il che è sufficiente per catturare le relazioni locali necessarie per determinare il percorso più breve senza eccessiva complessità computazionale. Gli strati nascosti del Multi-Layer Perceptron (MLP) sono stati impostati a due (`num_mlp_hidden_layers=2`). Questa configurazione rappresenta un buon compromesso tra capacità espressiva e rischio di overfitting: più strati potrebbero rendere il modello troppo complesso e soggetto a sovradattamento, mentre un numero inferiore potrebbe non essere in grado di catturare sufficientemente le interazioni non lineari tra le caratteristiche dei nodi e degli archi. La dimensione dello strato nascosto del MLP è stata scelta pari a 64 unità (`mlp_hidden_size=64`). Questa scelta fornisce al modello una capacità sufficiente per apprendere rappresentazioni complesse, senza rischiare di esagerare nella complessità. Allo stesso modo, la dimensione latente è stata impostata a 64 (`latent_size=64`), consentendo una rappresentazione efficace delle informazioni di ciascun nodo o arco dopo l'encoding. È stata inoltre abilitata la normalizzazione a livello di strato (`use_layer_norm=True`) e deciso di utilizzare processori condivisi tra i vari passi di message passing (`shared_processors=True`). Questa scelta riduce il numero di parametri del modello e quindi il rischio di overfitting. Per quanto riguarda l'ottimizzatore, è stato scelto `Adam`, una delle tecniche di ottimizzazione più utilizzate nei modelli di deep learning. Adam combina i vantaggi della discesa del gradiente con momento e dell'adaptive learning rate, rendendolo particolarmente efficace per problemi complessi e reti neurali profonde. La sua capacità di adattare i tassi di apprendimento per ogni parametro consente di migliorare la velocità di convergenza e la stabilità del processo di addestramento.

### 3.5.1 Integrazione della progettazione

Il modello richiede in input un tipo particolare di grafo della classe `GraphTensor` che richiede una precedente elaborazione avendo già a disposizione i parametri riguardandi i nodi di partenza e arrivo. Questa modalità rende il processo poco efficiente in quanto richiederebbe un continuo ricalcolo del grafo. Per risolvere questa problematica viene generato il grafo di tipo `GraphTensor`, partendo dal grafo dell'edificio. Per rispettare i requisiti sui parametri, tutti i nodi hanno i valori relativi alla loro presenza come nodo di partenza, di arrivo o di passaggio impostati su falso. Viene aggiunta una nuova classe che estende `GraphTensor` chiamata `GraphTensorUtility` in cui vengono aggiunti i metodi necessari per manipolare istanze di questo tipo.

```
class GraphTensorUtility(tfgnn.GraphTensor):
    def __init__(self, graph_tensor: tfgnn.GraphTensor, graph_nodes):
      self.graph_tensor = graph_tensor
      self.graph_nodes = graph_nodes
```

```
5
6       def get_graph(self):
7         return self.graph_tensor
8
9       def get_graph_nodes(self):
10        return self.graph_nodes
11
12      def set_graph(self, graph_tensor):
13        self.graph_tensor = graph_tensor
14
15      def set_graph_nodes(self, graph_nodes):
16        self.graph_nodes = graph_nodes
```

I metodi sopra indicati si riferiscono a istruzioni di base, come i getter e setter del `graph_tensor` e dei `graph_nodes` attualmente salvati all'interno dell'utility. Mentre il seguente metodo viene utilizzato per impostare il nodo di partenza e di arrivo prima di passare il grafo al modello ed ottenere il percorso migliore.

```
1  def setStartEndCity(self, startNode, endNode):
2    # Aggiungi/aggiorna le feature nel set di nodi "cities"
3    nodes_data = self.graph_nodes
4    nodes_data[startNode]["is_start"] = True
5    nodes_data[endNode]["is_end"] = True
6    node_features = tf.nest.map_structure(
7      lambda *n: np.stack(n, axis=0), *nodes_data)
8
9    assert isinstance(node_features, Mapping)
10
11   new_city_feat = self.graph_tensor.node_sets["cities"]
12                          .replace_features(node_features)
13   # Sostituisce le feature del set di nodi "cities"
14   # con le nuove feature
15   test_graph = self.graph_tensor.replace_features(
16                      node_sets={"cities": new_city_feat.features})
17   return test_graph
```

A partire da una copia dei nodi con parametri generici, vengono assegnati i valori effettivi al nodo di partenza e al nodo di arrivo. Successivamente, i nodi vengono trasformati in un oggetto di tipo `Mapping` che raccoglie tutte le feature. In questo modo, è possibile sostituire le feature dei nodi all'interno del grafo e ottenere un nuovo grafo con le informazioni aggiornate. Il metodo restituisce il grafo modificato, pronto per essere utilizzato come input nel modello per il calcolo del percorso.

```
1  def printNodesInPath(self):
2    nodes = self.graph_tensor.node_sets["cities"]
3
4    is_start_attribute = nodes['is_start']
5    is_end_attribute = nodes['is_end']
6    is_path_attribute = nodes['is_in_path']
7
8    start_indices = tf.where(is_start_attribute)[:, 0]
9    path_indices = tf.where(is_path_attribute)[:, 0]
```

```
    end_indices = tf.where(is_end_attribute)[:, 0]
    start_indices = start_indices.numpy()
    path_indices = path_indices.numpy()
    end_indices = end_indices.numpy()
    all_indices = tf.unique(tf.concat([start_indices,
                                       path_indices,
                                       end_indices],
                                       axis=0)).y.numpy()

    return all_indices
```

Una volta calcolato il percorso utilizzando la funzione sopra descritta, si possono ottenere gli indici di tutti i nodi che lo compongono. Per ogni nodo, i valori degli attributi "is_start", "is_end" e "is_in_path" vengono assegnati rispettivamente alle variabili is_start_attribute, is_end_attribute e is_path_attribute, creando così un array per ciascuno di questi attributi. Successivamente, vengono identificati e salvati gli indici dei nodi per cui questi attributi sono impostati su "True", e infine tutti gli indici vengono uniti in un unico array, che viene restituito come risultato. Per gestire la nuova tipologia di calcolo del percorso, è stata introdotta la classe `PredictionService`, dedicata esclusivamente alla gestione del modello. Questa classe contiene solo i metodi necessari per l'istanziazione del modello e l'esecuzione della funzione `predict`[4] sul grafo fornito.

```
def predict(tensorGraph, graph_nx, startNode, endNode):
    global build_initial_hidden_state
    global trainable_gnn
    if build_initial_hidden_state == "":
        build_initial_hidden_state = tfgnn.keras.layers.MapFeatures(
            node_sets_fn=_set_initial_node_state,
            edge_sets_fn=_set_initial_edge_state,
            context_fn=_set_initial_context_state)

        trainable_gnn = ShortestPathModel(
            edge_output_size=2,
            node_output_size=2,
            context_output_size=None,
            num_message_passing_steps=2,
            num_mlp_hidden_layers=2,
            mlp_hidden_size=64,
            latent_size=64,
            use_layer_norm=True,
            shared_processors=True,
            )
        try:
            trainable_gnn.load_weights(weights_path)
        except Exception as error:
            print("Weights not found", error)

    nodes_data = [data for _, data in graph_nx.nodes(data=True)]
```

[4]Nel contesto dell'intelligenza artificiale si riferisce al processo in cui un modello utilizza i dati di input per generare un'ipotesi o una stima su un risultato sconosciuto, basandosi sui pattern appresi durante la fase di addestramento.

```
27      tensor_temp = GraphTensorUtility(tensorGraph, nodes_data)
28      g_tensor = tensor_temp.setStartEndCity(
29                              Graph.string_to_int_map[startNode],
30                              Graph.string_to_int_map[endNode])
31      input_graph = build_initial_hidden_state(g_tensor)
32      output_graph = trainable_gnn(input_graph)
33      predicted_task_graph = predict_from_final_hidden_state(
34          g_tensor, output_graph)
35      tempGraph = GraphTensorUtility(predicted_task_graph, nodes_data)
36
37      all_indices = tempGraph.printNodesInPath()
38      keys = [key for index in all_indices
39                  for key, value in Graph.string_to_int_map.items()
40                      if value == index]
41      return keys
```

Utilizziamo variabili di tipo global per mantenere il valore delle variabili tra le diverse esecuzioni richieste da client diversi. In queste variabili salviamo un'istanza del `ShortestPathModel`, in cui andiamo a caricare i pesi del modello precedentemente allenato. Salviamo anche la procedura `build_initial_hidden_state` utilizzata per impostare lo stato iniziale su tutti i nodi e archi del grafo. Entrambi vengono creati solamente alla prima esecuzione, mentre nelle esecuzioni successive si passa direttamente alla fase di predict. Il codice relativo al processo di predict vero e proprio inizia quando vengono recuperati tutti i nodi del grafo e si istanzia un oggetto di tipo `GraphTensorUtility`, passando il grafo convertito e i nodi. A questo punto, si impostano immediatamente i nodi di partenza e di arrivo e si procede a inizializzare lo stato di tutti i nodi, rendendoli pronti per l'inferenza. Una volta ottenuto il grafo con le informazioni relative al percorso, utilizziamo il metodo `printNodesInPath` per estrarre gli indici dei nodi che appartengono al percorso calcolato. Successivamente, grazie al dizionario creato nella fase di costruzione del grafo, otteniamo una corrispondenza tra l'indice e l'ID del nodo. Tale relazione, in un secondo momento, consente di recuperare tutte le informazioni pertinenti a ciascun nodo.

## 3.6 Integrazione nel sistema

Per affrontare le problematiche menzionate in questo capitolo, è stato necessario apportare modifiche al servizio API precedentemente sviluppato. La classe `Graph` ha subito i cambiamenti principali per integrare il nuovo tipo di grafo necessario al calcolo e alla gestione dello stesso. Al contempo, la classe `NavigationService` è stata leggermente modificata per quanto riguarda il calcolo del percorso e la creazione dell'istanza di tipo `Navigation`.

### 3.6.1 Modifiche del codice

La classe `NavigationService` ha subito alcuni cambiamenti nel metodo `calculateRoute` che come suggerisce il nome si occupa di calcolare il percorso. Una volta ottenuto il grafo dell'edificio, si accede alla sua versione convertita per procedere con l'operazione di predict descritta in precedenza. Una volta ottenuta la lista con gli id dei nodi presenti nel percorso, andiamo a recuperare tutte le informazioni su ciascuno di essi, selezionando un nodo ogni tre, con particolare attenzione a quelle riguardanti la loro posizione. Ciascun punto di interesse selezionato viene aggiunto ad un array componente i checkpoint del percorso. Viene creata un'istanza di tipo `Navigation` la quale contiene il punto di interesse di partenza, di arrivo e la lista dei checkpoint da attraversa per seguire il percorso ottimale.

```
def calculateRoute(self, idSource: str, idTarget: str) -> Navigation:
    intPoint = self.navRepo.getInterestPointById(idSource)
    floor = self.navRepo.getFloorById(intPoint.idFloor)
    nodes = self.navRepo.getNodesOfBuilding(floor.idBuilding)
    converted_graph = nodes.get_dataset()

    task_dataset = iter(converted_graph)
    for _, task_graph in enumerate(task_dataset):
        allCheckpoints = predict(task_graph,
                                 nodes.graph_nx,
                                 idSource,
                                 idTarget)
    route: List[Checkpoint] = []
    for i in range(0, (len(allCheckpoints) - 1), 3):
        intPointCheck = self.navRepo.
                        getInterestPointById(allCheckpoints[i])
        check = Checkpoint(intPointCheck, i)
        route.append(check)

    target = self.navRepo.getInterestPointById(idTarget)
    check = Checkpoint(target, i)
    route.append(check)
    nav = Navigation(str(uuid.UUID), intPoint, target, route)
    return nav
```

Mentre la classe `Graph` è stata arricchita con i metodi necessari per la conversione ed utilizzo del grafo convertito. La funzione che si occupa di restiruire il grafo convertito è `get_dataset`, la quale se presente ritorna il grafo precedentemente convertito, altrimenti inizia la conversione. Il metodo procede creando una specifica del grafo utilizzando la funzione `graph_tensor_spec_from_sample_graph`, che si basa sulla conversione del grafo effettuata tramite il metodo `_convert_to_graph_tensor`. Successivamente, viene generato un dataset TensorFlow utilizzando `tf.data.Dataset.from_generator`, per mantenere la stessa struttura dell'oggetto utilizzato nel training, definendo la struttura del dataset tramite la specifica del grafo. Infine, il primo, ed unico, grafo generato viene salvato nella variabile `convertedGraph` per un uso futuro e restituito.

```
def task_graph_generator(self):
    while True:
        yield self._convert_to_graph_tensor()

def get_dataset(self):
    if self.convertedGraph != {}:
        print("graph present")
        return self.convertedGraph

    def generator_fn():
        return self.task_graph_generator()
    graph_spec = graph_tensor_spec_from_sample_graph(
                                 self._convert_to_graph_tensor())
    generator = tf.data.Dataset.from_generator(
        generator_fn, output_signature=graph_spec)
```

```
16      self.convertedGraph = generator.take(1)
17      return self.convertedGraph
```

Il metodo `_convert_to_graph_tensor` trasforma un grafo tradizionale, creato con la libreria networkx, in un formato compatibile con TensorFlow, chiamato `GraphTensor`. Prima di tutto, viene generato un grafo di base attraverso il metodo `_generate_base_graph`, che viene approfondito in seguito. Il metodo estrae i dati relativi ai nodi (come caratteristiche e attributi) e li converte in un formato utilizzabile da TensorFlow, usando la funzione `tf.nest.map_structure`. Oltre ai nodi e agli archi, viene raccolto un set di dati generali (o contesto) che rappresenta le informazioni globali dell'intero grafo. Tutte queste informazioni vengono poi combinate in un oggetto `GraphTensor`, che rappresenta l'intero grafo, con i nodi etichettati come "cities" e gli archi come "roads". Successivamente, questo oggetto `GraphTensor` viene arricchito con funzionalità specifiche attraverso la classe `GraphTensorUtility`, permettendo di collegare i dati originali del grafo ai nodi convertiti, in modo da mantenere una corrispondenza tra il grafo iniziale e quello trasformato. Infine, il grafo convertito viene restituito pronto per essere utilizzato nell'inferenza.

```
1  def _convert_to_graph_tensor(self):
2      """Converts the graph to a GraphTensor."""
3      graph_nx = self._generate_base_graph()
4
5      number_of_nodes = graph_nx.number_of_nodes()
6      nodes_data = [data for _, data in graph_nx.nodes(data=True)]
7      node_features = tf.nest.map_structure(
8          lambda *n: np.stack(n, axis=0), *nodes_data)
9
10     number_of_edges = graph_nx.number_of_edges()
11     source_indices, target_indices, edges_data =
12                             zip(*graph_nx.edges(data=True))
13     s_index = []
14     t_index = []
15     for index in source_indices:
16       s_index.append(string_to_int(index))
17
18     for index in target_indices:
19       t_index.append(string_to_int(index))
20     source_indices = np.array(s_index, dtype=np.int32)
21     target_indices = np.array(t_index, dtype=np.int32)
22     edge_features = tf.nest.map_structure(
23         lambda *e: np.stack(e, axis=0), *edges_data)
24     context_features = dict(graph_nx.graph)
25     self.tensorGraph = tfgnn.GraphTensor.from_pieces(
26         node_sets={
27             "cities": tfgnn.NodeSet.from_fields(
28                 sizes=[number_of_nodes],
29                 features=node_features,
30                 )},
31         edge_sets={
32             "roads": tfgnn.EdgeSet.from_fields(
33                 sizes=[number_of_edges],
34                 features=edge_features,
```

```
                adjacency=tfgnn.Adjacency.from_indices(
                    source=("cities", source_indices),
                    target=("cities", target_indices))
                )},
            context=tfgnn.Context.from_fields(features=context_features),
            )

        self.tensorGraph.__class__ = GraphTensorUtility
        self.tensorGraph.set_graph_nodes(self.graph_nx.nodes(data=True))
        return self.tensorGraph
```

Il metodo `_generate_base_graph` crea un grafo di base utilizzando la libreria networkx, aggiungendo nodi, archi e attributi personalizzati. I nodi vengono aggiunti al grafo utilizzando i dati memorizzati nella variabile `self.graph`, dove ogni nodo è connesso ai suoi vicini attraverso archi con un peso specificato ma senza la presenza di attributi personalizzati. Dopo aver aggiunto i nodi e gli archi, il metodo assegna gli attributi ai nodi, quali "is_start", "is_end", e "is_in_path", inizialmente impostati a "False". Questi attributi saranno utili in seguito per identificare il nodo di partenza, quello di arrivo e i nodi che appartengono al percorso ottimale. Anche agli archi viene assegnato l'attributo "is_in_path" per indicare se fanno parte del percorso calcolato. Infine, il grafo con tutti i nodi, archi e attributi personalizzati viene memorizzato nella variabile `self.graph_nx` e restituito per essere convertito in un `GraphTensor`.

```
def _generate_base_graph(self):
    rand = np.random.RandomState(0)
    dimensions = 2
    geo_graph = nx.Graph()

    # Add nodes to the graph.
    geo_graph.add_nodes_from(self.graph.keys())
    for node, neighbors in self.graph.items():
        for neighbor, weight in neighbors.items():
            geo_graph.add_edge(node, neighbor, weight=weight)

    # Generate random positions for nodes.
    pos = {node: rand.uniform(size=dimensions)
                for node in geo_graph.nodes()}


    # Add "weight" and "pos" features to the "cities" node set.
    weight_data = {node: 1 for node in geo_graph.nodes()}
    pos_data = {node: pos[node] for node in geo_graph.nodes()}

    # Add features to the graph
    nx.set_node_attributes(geo_graph, values=weight_data,
                           name='weight')
    nx.set_node_attributes(geo_graph, values=pos_data, name='pos')

    # Add custom attributes to nodes and edges.
    for node in geo_graph.nodes():
        geo_graph.nodes[node]['is_start'] = False
```

```
            geo_graph.nodes[node]['is_end'] = False
            geo_graph.nodes[node]['is_in_path'] = False

        for edge in geo_graph.edges(data=True):
            data = edge[2]
            data['is_in_path'] = False

        self.graph_nx = geo_graph
        return geo_graph
```

La funzione `graph_tensor_spec_from_sample_graph` è progettata per generare una specifica del grafo a partire da un grafo esemplificativo fornito come input. Questo processo è essenziale per garantire che il grafo possa essere utilizzato correttamente nelle operazioni di apprendimento automatico che utilizzano grafi, in particolare con la libreria TensorFlow Graph Neural Networks (TF-GNN). La funzione procede a elaborare le specifiche dei set di nodi, creando un dizionario vuoto chiamato `node_sets_spec`. Per ogni set di nodi, viene creata una nuova specifica per le funzionalità, in cui ogni nome di funzionalità e la relativa specifica vengono convertiti tramite la funzione `_to_none_leading_dim`. Questo passaggio assicura che le dimensioni siano corrette per l'elaborazione successiva. Ogni set di nodi viene quindi memorizzato in `node_sets_spec` utilizzando `tfgnn.NodeSetSpec.from_field_specs`. Allo stesso modo, la funzione gestisce i set di archi creando un dizionario chiamato `edge_sets_spec`. La specifica del contesto del grafo viene recuperata e utilizzata per costruire la specifica finale del grafo tramite `tfgnn.GraphTensorSpec.from_piece_specs`. Se l'array `array_nodes` è vuoto, viene riempito con i nodi del grafo fornito. La funzione restituisce quindi la specifica del grafo, che sarà utilizzata nelle fasi successive per assicurarsi che tutte le strutture dati siano conformi alle aspettative.

```
def graph_tensor_spec_from_sample_graph(sample_ut_graph):
    sample_graph = sample_ut_graph
    tfgnn.check_scalar_graph_tensor(sample_graph)
    sample_graph_spec = sample_graph.spec
    node_sets_spec = {}
    for node_set_name, node_set_spec in
                    sample_graph_spec.node_sets_spec.items():
        new_features_spec = {}
        for feature_name, feature_spec in
                        node_set_spec.features_spec.items():
            new_features_spec[feature_name] =
                            _to_none_leading_dim(feature_spec)
        node_sets_spec[node_set_name] =
                            tfgnn.NodeSetSpec.from_field_specs(
                        features_spec=new_features_spec,
                        sizes_spec=tf.TensorSpec(shape=(1,),
                        dtype=tf.int32))

    edge_sets_spec = {}
    for edge_set_name, edge_set_spec in
                    sample_graph_spec.edge_sets_spec.items():
      new_features_spec = {}
      for feature_name, feature_spec in
```

```
                        edge_set_spec.features_spec.items():
            new_features_spec[feature_name] =
                        _to_none_leading_dim(feature_spec)

        adjacency_spec = tfgnn.AdjacencySpec.from_incident_node_sets(
            source_node_set=edge_set_spec.adjacency_spec.source_name,
            target_node_set=edge_set_spec.adjacency_spec.target_name,
            index_spec=_to_none_leading_dim(
                                edge_set_spec.adjacency_spec.target))

        edge_sets_spec[edge_set_name] =
                    tfgnn.EdgeSetSpec.from_field_specs(
                            features_spec=new_features_spec,
                            sizes_spec=tf.TensorSpec(shape=(1,),
                            dtype=tf.int32),
                            adjacency_spec=adjacency_spec)

    context_spec = sample_graph_spec.context_spec

    graph_spec = tfgnn.GraphTensorSpec.from_piece_specs(
        node_sets_spec=node_sets_spec,
        edge_sets_spec=edge_sets_spec,
        context_spec=context_spec,
    )
    if len(array_nodes) == 0:
        array_nodes.append(sample_ut_graph.get_graph_nodes())
    return graph_spec
```

CAPITOLO 4

# Valutazione del sistema

*Questo capitolo tratta l'addestramento del modello e la valutazione delle sue prestazioni. Esamineremo i grafi utilizzati, l'infrastruttura hardware impiegata per l'ottimizzazione del training, e discuteremo i risultati ottenuti in termini di loss e accuracy.*

## 4.1 Addestramento

L'addestramento di un modello di intelligenza artificiale (IA) è il processo attraverso il quale il modello apprende dai dati. Durante questa fase, il modello viene esposto a un insieme di dati di apprendimento, utilizzando algoritmi specifici per ottimizzare le sue capacità predittive. Questo processo può essere visto come un insegnamento: proprio come un essere umano impara attraverso l'esperienza, un modello di IA impara dai dati che gli vengono forniti. L'importanza dell'addestramento è fondamentale per garantire una buona performance del modello. Un modello ben allenato è in grado di generalizzare, cioè di applicare ciò che ha appreso a dati mai visti prima, effettuando previsioni accurate. Al contrario, un modello mal addestrato rischia di soffrire di problemi come l'overfitting, dove si adatta eccessivamente ai dati di addestramento senza riuscire a performare altrettanto bene su dati nuovi. In questo contesto, l'apprendimento non è solo un passo necessario, ma rappresenta il cuore stesso della creazione di modelli di IA efficaci, in grado di risolvere problemi complessi in diversi ambiti.

### 4.1.1 Caratteristiche del grafo

Quando si progetta un grafo per un caso d'uso specifico, è cruciale considerare alcune caratteristiche chiave che ne ottimizzano la funzionalità e l'efficacia dell'algoritmo di apprendimento. In questa situazione è necessario avere un grafo in cui ogni nodo sia collegato da 2 a 3 archi. Questa limitazione è importante perché garantisce una sufficiente somiglianza con i grafi reali costruti a partire dai punti di interesse. connettività tra i nodi, permettendo al modello di esplorare diverse rotte senza complicare eccessivamente la struttura del grafo. Un numero moderato di archi per nodo favorisce infatti l'efficienza nell'elaborazione e nella gestione della memoria, riducendo il rischio di overfitting e mantenendo il grafo relativamente semplice e facilmente interpretabile. Un'altra caratteristica fondamentale è che il peso degli archi deve rappresentare la distanza effettiva tra i nodi, calcolata secondo la formula "x+y". Questa scelta è cruciale poiché i pesi forniscono un'indicazione quantitativa della difficoltà

di percorrere un arco specifico. Utilizzando la distanza effettiva, il modello può effettuare valutazioni più accurate sui percorsi, facilitando la selezione del percorso ottimale.

### 4.1.2 Algoritmo per la costruzione dei grafi

Viste le particolari esigenze della struttura del grafo, è stato implementato un algoritmo ad-hoc per la generazione del dataset da utilizzare per l'addestramento. La generazione dei dati utili per l'allenamento si divide in step:

- Nel primo step si va a generare il grafo di base con le caratteristiche richieste in precedenza.
- Nel secondo step si procede a duplicare il grafo andando ad avere un istanza per ogni coppia di nodi che ricoprono il punto di partenza e di arrivo. Questa peculiarità serve per generalizzare ancora di più il modello.

Attraverso la funzione `generate_task_graph_all_streets` ha il compito di generare un numero finito di grafi indicato nel parametro `num_graph` e per ciascuno di essi, calcolare tutti i possibili percorsi più brevi tra le coppie di nodi. Restituisce una lista contenente i dati dei grafi generati in forma di `GraphTensor`. Il parametro `random_state` rappresenta un generatore di numeri casuali, utilizzato per garantire una casualità controllata nella creazione degli elementi del grafo. Questo permette di ottenere risultati riproducibili. `num_nodes_min_max`, invece, è una coppia di valori che specifica il numero minimo e massimo di nodi che possono essere inclusi nel grafo. Il numero effettivo di nodi viene scelto casualmente all'interno di questo intervallo.

```
def generate_task_graph_all_streets(
    num_graph: int,
    random_state,
    num_nodes_min_max: Tuple[int, int],
    dimensions: int = 2,
    rate: float = 1.0) -> tfgnn.GraphTensor:
  graph_tensor = []
  i = 0
  while(i < num_graph):
    graph = _generate_base_graph(
        random_state, num_nodes_min_max=num_nodes_min_max,
        dimensions=dimensions,
        rate=rate)
    graph_tensor.extend(_add_shortest_path_all_combinations(graph))
    i = i+1

  return graph_tensor
```

La funzione `_generate_base_graph` crea un grafo geometrico utilizzando alcuni parametri specifici. Il parametri ricevuti come input, quali `rand` e `num_nodes_min_max` sono stati illustrati in precedenza. La funzione inizia creando un numero casuale di nodi all'interno del grafo. Ogni nodo viene poi collegato a un numero di nodi vicini, garantendo che ciascuno abbia da 2 a 3 archi (o collegamenti) usando una metrica di distanza (calcolata come distanza euclidea tra i nodi). Viene poi aggiunta una caratteristica ai nodi chiamata "weight" (peso), che è un valore generato casualmente per ogni nodo e arco. Per assicurare che il grafo sia connesso, la funzione controlla se ci sono componenti disconnesse e, in tal caso, aggiunge degli archi per collegarle. Successivamente, vengono aggiunti attributi specifici ai nodi e agli

archi per marcare se fanno parte del percorso, che inizialmente sono impostati su "False". La funzione restituisce un grafo completo e connesso.

```
def _generate_base_graph(rand,
                         num_nodes_min_max,
                         dimensions,
                         rate):
  num_nodes = rand.randint(*num_nodes_min_max)
  geo_graph = nx.Graph()
  geo_graph.add_nodes_from(range(num_nodes))

  pos = {node: rand.uniform(size=dimensions) for node in
                                              geo_graph.nodes()}
  for i in geo_graph.nodes():
    remaining_edges = rand.randint(2, 4) #Ensure 2-3 edges per node
    distances = np.linalg.norm(pos[i] - np.array(
                                          list(pos.values())),
                                          axis=1)
    sorted_indices = np.argsort(distances)
    for j in sorted_indices[1:]:
      geo_graph.add_edge(i, j, weight=distances[j])
      remaining_edges -= 1
      if remaining_edges == 0:
        break

  weight_data = {node: rand.exponential(rate) for node in
                                              geo_graph.nodes()}
  pos_data = {node: pos[node] for node in geo_graph.nodes()}

  nx.set_node_attributes(geo_graph, values=weight_data,
                         name='weight')
  nx.set_node_attributes(geo_graph, values=pos_data, name='pos')

  # Ensure connectivity by adding edges to connect components.
  while not nx.is_connected(geo_graph):
    for component in nx.connected_components(geo_graph):
      if len(component) <= 30:
        central_node = rand.choice(list(geo_graph.nodes))
        for node in component:
          if node != central_node:
            geo_graph.add_edge(central_node, node, weight=5)
            break

  # Add custom attributes to nodes and edges.
  for node in geo_graph.nodes():
    geo_graph.nodes[node]['is_start'] = False
    geo_graph.nodes[node]['is_end'] = False
    geo_graph.nodes[node]['is_in_path'] = False

  for edge in geo_graph.edges(data=True):
    data = edge[2]
```

```
49      data['is_in_path'] = False
50
51    return geo_graph
```

La funzione `_add_shortest_path_all_combinations` implementa il secondo step della generazione del dataset, andando a calcolare i percorsi più brevi tra tutte le coppie di nodi all'interno di un grafo e di memorizzarli in un formato adatto per ulteriori elaborazioni. Inizia generando una lista di tutte le possibili coppie di nodi presenti nel grafo, utilizzando la funzione `combinations()`. Queste coppie rappresentano i punti di partenza e di arrivo per i percorsi da calcolare. Successivamente, per ciascuna coppia di nodi, la funzione utilizza `nx.shortest_path()` per determinare il percorso più breve, tenendo conto dei pesi associati agli archi del grafo. Una volta calcolato il percorso, la funzione crea un grafo diretto, trasformando il grafo originale in modo che ogni arco abbia una direzione ben definita. Questo passaggio è importante per rappresentare il percorso in modo chiaro e preciso. Dopo aver creato il grafo diretto, la funzione procede ad aggiungere attributi specifici ai nodi e agli archi. Il nodo di partenza viene contrassegnato con l'attributo `is_start=True`, mentre il nodo di arrivo riceve l'attributo `is_end=True`. Per i nodi che fanno parte del percorso più breve, viene aggiunto l'attributo `is_in_path=True`, mentre gli altri nodi e archi che non fanno parte del percorso mantengono l'attributo `is_in_path=False`. Questo approccio consente di distinguere facilmente quali elementi sono coinvolti nel percorso. Una volta che tutti questi attributi sono stati assegnati, il grafo diretto con i percorsi viene convertito in un formato idoneo per l'uso in modelli di machine learning tramite la funzione `_convert_to_graph_tensor()`. Infine, tutti i grafi diretti che rappresentano i percorsi più brevi vengono memorizzati in una lista, la quale viene restituita come risultato della funzione.

```
1  def _add_shortest_path_all_combinations(graph):
2    node_pairs = list(combinations(graph.nodes(), 2))
3    graph_array = []
4    for pairs in node_pairs:
5      start = pairs[0]
6      end = pairs[1]
7      path = nx.shortest_path(
8          graph, source=start, target=end, weight="length")
9
10     # Creates a directed graph, to store the directed path
11     # from start to end.
12     digraph = graph.to_directed()
13
14     # Add the "start", "end", and "solution" attributes to
15     #the nodes and edges.
16     digraph.add_node(start, is_start=True)
17     digraph.add_node(end, is_end=True)
18     digraph.add_nodes_from(_set_diff(digraph.nodes(), [start]),
19                            is_start=False)
20     digraph.add_nodes_from(_set_diff(digraph.nodes(), [end]),
21                            is_end=False)
22     digraph.add_nodes_from(_set_diff(digraph.nodes(), path),
23                            is_in_path=False)
24     digraph.add_nodes_from(path, is_in_path=True)
25     path_edges = list(_pairwise(path))
26     digraph.add_edges_from(_set_diff(digraph.edges(), path_edges),
27                            is_in_path=False)
```

```
28      digraph.add_edges_from(path_edges, is_in_path=True)
29      graph_array.append(_convert_to_graph_tensor(digraph))
30
31    return graph_array
```

### 4.1.3 Infrastruttura Hardware e Ottimizzazione del Training

Per l'addestramento di un modello di IA è necessaria una notevole potenza di calcolo, poiché i processi coinvolti, come il calcolo dei gradienti e l'aggiornamento dei parametri del modello, richiedono un elevato utilizzo di risorse computazionali. In questo caso, ci siamo affidati a un server dotato di una scheda grafica con 12GB di memoria DDR5. Questa potente GPU ci ha permesso di eseguire calcoli paralleli in modo estremamente efficiente, riducendo i tempi di addestramento e permettendoci di utilizzare batch più grandi durante il processo.

La quantità di memoria sulla GPU ha un impatto diretto sulla dimensione del batch che può essere utilizzato durante l'allenamento. I batch rappresentano il numero di campioni che vengono processati in una singola iterazione dell'algoritmo di ottimizzazione. Un batch più grande consente di calcolare medie più precise e ridurre il rumore durante l'aggiornamento dei pesi del modello, il che porta generalmente a un processo di allenamento più stabile e, in molti casi, a una convergenza più veloce del modello. Grazie ai 12 GB di memoria della nostra scheda grafica DDR5, è stato possibile incrementare significativamente le dimensioni del batch rispetto a sistemi dotati di GPU con meno memoria. Questo ha permesso di caricare più campioni di dati nella memoria della GPU contemporaneamente, riducendo il numero di iterazioni necessarie per completare un'epoca di addestramento. Un altro vantaggio derivante dall'uso di batch più grandi è la riduzione del tempo complessivo di allenamento. Infatti, l'aggiornamento dei pesi del modello avviene meno frequentemente, il che significa che il modello richiede meno tempo per passare attraverso l'intero dataset. L'uso di batch di dimensioni più grandi non solo ha migliorato l'efficienza in termini di tempo, ma ha anche contribuito a una maggiore stabilità durante l'allenamento. Uno degli aspetti critici dell'addestramento di modelli di intelligenza artificiale è evitare oscillazioni o aggiornamenti instabili dei pesi, specialmente nei modelli più complessi o quando si lavora con set di dati rumorosi. Batch più grandi tendono a fornire un gradiente meno rumoroso, il che significa che gli aggiornamenti dei pesi sono meno soggetti a fluttuazioni casuali causate da campioni particolarmente difficili o atipici nel set di dati. Questo ha portato a un miglioramento generale nella qualità del modello, riducendo il rischio di overfitting, soprattutto nelle fasi iniziali dell'allenamento, dove un gradiente meno stabile può facilmente portare a modelli subottimali.

## 4.2 Valutazione del grafo

La valutazione del modello di intelligenza artificiale è una fase cruciale che segue l'addestramento e consente di determinare l'efficacia del modello nel risolvere il problema per cui è stato progettato. Durante questa fase, vengono utilizzate diverse metriche per misurare le prestazioni del modello. Due delle metriche più comuni impiegate in questa fase sono:

- "accuracy": Rappresenta la percentuale di predizioni corrette fatte dal modello rispetto al totale degli esempi. Si utilizza comunemente nei problemi di classificazione e fornisce una misura diretta delle prestazioni del modello, ovvero quanto spesso il modello ha indovinato la classe giusta.

- "losses": Misurano quanto le predizioni del modello si discostano dai valori reali. La loss serve a guidare il processo di addestramento del modello, poiché l'algoritmo cerca di minimizzare questa misura d'errore.

Ciascuna delle quali fornisce informazioni importanti sulla qualità del modello e su come migliorarlo ulteriormente. Questa valutazione viene effettuata testando il modello su dati che non ha mai visto prima, noti come dati di validazione o di test, con l'obiettivo di valutare la sua capacità di generalizzare, ossia di fare previsioni accurate su dati nuovi.

Nel nostro caso, per valutare il modello, abbiamo utilizzato i grafi generati in precedenza come dati di test, mantenuti separati durante tutto il processo di addestramento. Questo approccio garantisce una valutazione oggettiva.

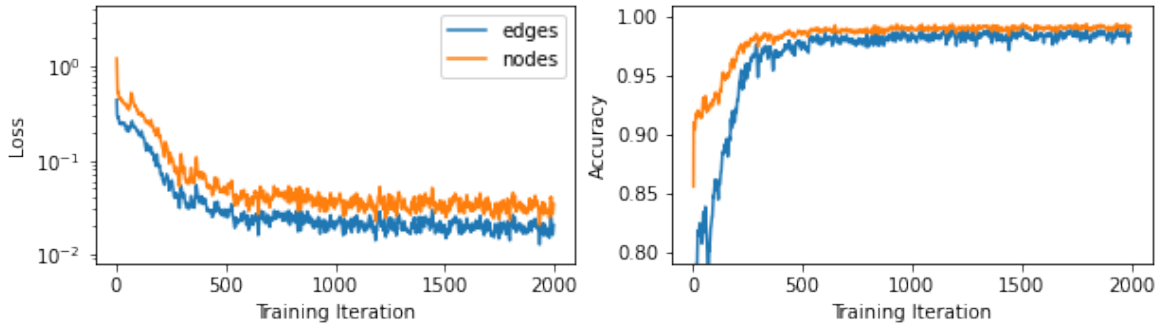


**Figura 4.1:** Grafico di loss e accuracy

Il grafico mostra l'andamento della loss e dell'accuracy durante il processo di addestramento di un modello su due diversi set di dati: "edges" (rappresentato in blu) e "nodes" (rappresentato in arancione).

Nel grafico di sinistra, possiamo osservare l'andamento della loss (funzione di perdita) lungo le iterazioni di addestramento. All'inizio, entrambe le curve di loss sono piuttosto alte, indicando un grande errore nelle predizioni iniziali del modello. Con il progredire dell'addestramento, le loss decrescono, segnalando che il modello sta imparando e migliorando le sue predizioni. Tuttavia, notiamo che la curva per "nodes" tende a stabilizzarsi su valori di loss leggermente più alti rispetto a "edges", il che potrebbe suggerire che il modello ha più difficoltà a ridurre l'errore su quel set di dati.

Nel grafico di destra, invece, si vede l'andamento dell'accuracy durante l'addestramento. Inizialmente, entrambe le curve partono con valori di accuracy bassi, ma rapidamente crescono fino a stabilizzarsi attorno al 95-99%. La curva per "nodes" raggiunge un'accuracy stabile più velocemente rispetto a "edges", il che indica che le predizioni corrette su "nodes" aumentano più rapidamente nelle fasi iniziali di addestramento. Tuttavia, entrambe le curve sembrano convergere su valori elevati di accuracy, segnalando che il modello è in grado di predire con grande precisione su entrambi i set di dati verso la fine del processo di addestramento.

# CAPITOLO 5

# Conclusione

Sulla base delle analisi condotte e delle implementazioni sviluppate, emergono alcune considerazioni conclusive di rilievo.

Un aspetto cruciale da sottolineare è l'efficace applicazione delle Graph Neural Networks (GNN) nel contesto di questo specifico problema reale. L'implementazione è stata condotta con successo, dimostrando la capacità del modello di gestire e risolvere situazioni complesse basate su strutture di grafi. Questo risultato acquisisce particolare importanza se si considera che le reti neurali per grafi rappresentano una tecnologia ancora relativamente nuova e sperimentale, la cui diffusione in applicazioni commerciali è attualmente limitata. Nonostante il loro potenziale teorico sia stato ampiamente riconosciuto nella ricerca accademica, la loro adozione pratica in scenari di produzione reale è ancora in fase di sviluppo.

L'applicazione di una GNN per questo progetto ha permesso di superare diverse sfide tecniche, tra cui la modellazione di relazioni complesse tra nodi e la capacità di estrarre informazioni rilevanti da strutture di dati non lineari, come i grafi. Il fatto che questa rete neurale sia stata in grado di fornire soluzioni di qualità evidenzia non solo la versatilità di questa architettura, ma anche la sua potenziale efficacia in una vasta gamma di domini applicativi che vanno oltre quello specifico analizzato in questa implementazione.

Un aspetto che merita particolare attenzione riguarda la velocità di esecuzione, soprattutto a confronto con gli algoritmi esatti tradizionali. In molti problemi basati su grafi, gli algoritmi esatti, come quelli utilizzati per trovare il percorso minimo o per la risoluzione di problemi combinatori complessi, possono risultare molto lenti, specialmente quando il numero di nodi e di archi cresce. Questi algoritmi, pur garantendo soluzioni precise, soffrono spesso di tempi di calcolo esponenziali in relazione alla complessità del problema. Le GNN, invece, grazie alla capacità di elaborare grandi quantità di dati in parallelo e di sfruttare le connessioni tra nodi in modo efficiente, offrono un compromesso tra accuratezza e rapidità di esecuzione. In questo senso, le Graph Neural Network rappresentano un'alternativa più scalabile, in grado di produrre risultati approssimati in tempi molto più ridotti, pur mantenendo un alto livello di precisione. Questo vantaggio di velocità rende le GNN particolarmente interessanti in applicazioni che richiedono risposte rapide, come i sistemi real-time. Tuttavia, va anche riconosciuto che, essendo una tecnologia ancora in fase di sperimentazione, non sono prive di sfide. Nonostante ciò, l'efficace implementazione in questo progetto rappresenta un importante passo avanti verso una maggiore maturità tecnologica e apre nuove prospettive per l'integrazione delle Graph Neural Network in contesti operativi concreti.

Un altro fattore rilevante riguarda la generalizzabilità della tecnologia, che ci permette di affrontare problemi complessi in vari settori, grazie alla sua natura intrinsecamente flessibile e adattabile. Uno dei settori in cui le GNN possono essere applicate è quello della salute. Le reti neurali per grafi possono essere utilizzate per analizzare le interazioni tra i diversi elementi biologici, come proteine e geni, permettendo di identificare potenziali bersagli per terapie e migliorare la comprensione delle malattie. Attraverso l'analisi delle reti biologiche possono contribuire a scoprire nuove relazioni tra patologie e trattamenti, supportando così la ricerca biomedica. Le reti sociali rappresentano un altro esempio di applicazione. Vengono analizzate le interazioni tra utenti per identificare gruppi, tendenze e influenze, supportando così le strategie di marketing e di comunicazione. Inoltre, questa tecnologia può essere utilizzata per combattere la disinformazione, analizzando le relazioni tra contenuti e utenti per identificare fonti di notizie affidabili. Nel settore finanziario, possono essere utilizzate per rilevare frodi e anomalie nelle transazioni, analizzando le relazioni tra utenti e transazioni per identificare comportamenti sospetti. Inoltre, possono aiutare nella modellazione dei rischi e nella previsione dei movimenti del mercato, fornendo alle istituzioni finanziarie strumenti più sofisticati per prendere decisioni informate. Infine, un altro ambito in cui mostrano grande potenziale è l'industria dei giochi e dell'intrattenimento. Queste reti possono essere utilizzate per sviluppare giochi più coinvolgenti, analizzando le interazioni tra i giocatori e creando dinamiche di gioco più complesse. Inoltre, possono facilitare la creazione di mondi di gioco interattivi e personalizzati, migliorando l'esperienza dell'utente.

La capacità delle Graph Neural Networks di adattarsi e generalizzarsi a diversi ambiti, unita alla loro velocità rispetto agli algoritmi esatti, ne fanno uno strumento promettente per affrontare problemi complessi in tempo reale. La loro applicazione nei settori della salute, dei trasporti, delle reti sociali, della finanza e dell'intrattenimento sottolinea il potenziale innovativo di questa tecnologia, contribuendo a creare soluzioni più efficienti e sostenibili.

In conclusione, gli sviluppi futuri di questo sistema aprono interessanti prospettive. Una possibile direzione potrebbe essere l'integrazione delle Graph Neural Networks con la logica del primo ordine, permettendo di arricchire il processo decisionale con capacità deduttive e ragionamenti logici complessi, migliorando così la comprensione delle relazioni strutturali tra i nodi. Questo sviluppo consentirebbe di affrontare problemi ancora più articolati e con un livello di astrazione superiore. Un ulteriore potenziale risiede nella collaborazione con istituzioni accademiche e università, con l'obiettivo di adottare questo sistema come piattaforma per applicazioni di navigazione intelligente e in tempo reale. Progetti di ricerca congiunti potrebbero esplorare nuove applicazioni pratiche, affinare ulteriormente la tecnologia e promuoverne l'uso in contesti urbani, trasportistici e scientifici. Questi sforzi congiunti potrebbero accelerare la diffusione e l'evoluzione di questo approccio innovativo in diversi settori industriali e accademici.

# Bibliografia


ALBAWI, S., MOHAMMED, T. A. e AL-ZAWI, S. (2017), «Understanding of a convolutional neural network», in «2017 international conference on engineering and technology (ICET)», p. 1–6, Ieee. (Cited at page 8)

BESSEN, J. (2018), «AI and jobs: The role of demand», Rap. tecn., National Bureau of Economic Research. (Cited at page 17)

BUCHANAN, B. G. (2019), «Artificial intelligence in finance», . (Cited at page 11)

CHOI, R. Y., COYNER, A. S., KALPATHY-CRAMER, J., CHIANG, M. F. e CAMPBELL, J. P. (2020), «Introduction to machine learning, neural networks, and deep learning», *Translational vision science & technology*, vol. 9 (2), p. 14–14.

ELI-CHUKWU, N. C. (2019), «Applications of artificial intelligence in agriculture: A review.», *Engineering, Technology & Applied Science Research*, vol. 9 (4). (Cited at page 16)

FAN, W., MA, Y., LI, Q., HE, Y., ZHAO, E., TANG, J. e YIN, D. (2019), «Graph neural networks for social recommendation», in «The world wide web conference», p. 417–426. (Cited at page 10)

FLOWERS, J. C. (2019), «Strong and Weak AI: Deweyan Considerations.», in «AAAI spring symposium: Towards conscious AI systems», vol. 2287. (Cited at page 4)

FRANCHETTO, F. (????), «Impianti geotermici, ecco il decreto di riassetto normativo», *Nextville.it – Energie rinnovabili ed efficienza energetica*.

GOODFELLOW, I., BENGIO, Y. e COURVILLE, A. (2016), *Deep learning*, MIT press. (Cited at page 17)

GUO, T., DONG, J., LI, H. e GAO, Y. (2017), «Simple convolutional neural network on image classification», in «2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)», p. 721–724, IEEE.

MEI, Q., XIE, Y., YUAN, W. e JACKSON, M. O. (2024), «A Turing test of whether AI chatbots are behaviorally similar to humans», *Proceedings of the National Academy of Sciences*, vol. 121 (9), p. e2313925 121. (Cited at pages iv, 5 e 6)

MOOR, J. (2006), «The Dartmouth College artificial intelligence conference: The next fifty years», *Ai Magazine*, vol. 27 (4), p. 87–87. (Cited at page 1)

RAJPURKAR, P., CHEN, E., BANERJEE, O. e TOPOL, E. J. (2022), «AI in health and medicine», *Nature medicine*, vol. 28 (1), p. 31–38. (Cited at page 12)

SANCHEZ-GONZALEZ, A., GODWIN, J., PFAFF, T., YING, R., LESKOVEC, J. e BATTAGLIA, P. (2020), «Learning to simulate complex physics with graph networks», in «International conference on machine learning», p. 8459–8468, PMLR. (Cited at page 49)

SZABADFÖLDI, I. (2021), «Artificial intelligence in military application–opportunities and challenges», *Land Forces Academy Review*, vol. 26 (2), p. 157–165. (Cited at pages iv, 8 e 12)

TURING, A. M., GEIRSSON, H. e LOSONSKY, M. (1950), «Computing machinery and intelligence», . (Cited at pages 1 e 4)

## Websites consulted

- AVG Technologies – `https://www.avg.com/`
- Braincomputing – `https://www.braincomputing.com/`
- Geopop – `https://www.geopop.it/`
- Harvard – `https://sitn.hms.harvard.edu/`
- Intelligenza Artificiale – `https://www.intelligenzaartificiale.it/`
- Ionos – `https://www.ionos.it`
- Microsoft – `https://microsoft.com`
- Parlamento Europeo – `https://www.europarl.europa.eu/`
- Sap – `https://www.sap.com`
- Talend – `https://www.talend.com/`
- Wikipedia – `https://it.wikipedia.org`

# Ringraziamenti

Desidero esprimere la mia gratitudine più sincera a tutte le persone che mi hanno supportato durante questo percorso. Un ringraziamento speciale va alla mia ragazza Eli, per il suo costante sostegno e pazienza, così come ai miei familiari e amici, che mi hanno sempre incoraggiato e motivato. Un sentito riconoscimento va anche al mio relatore, Dott. Enrico Corradini, per la sua guida e il suo prezioso supporto lungo tutto questo percorso. Vorrei anche ringraziare l'azienda per cui lavoro, doIT, che mi ha permesso di mettere in pratica quanto appreso, dimostrando grande flessibilità e dandomi l'opportunità di crescere professionalmente. Infine, un ringraziamento a me stesso per la pazienza e la perseveranza, che mi hanno permesso di superare le sfide e raggiungere questo importante traguardo.