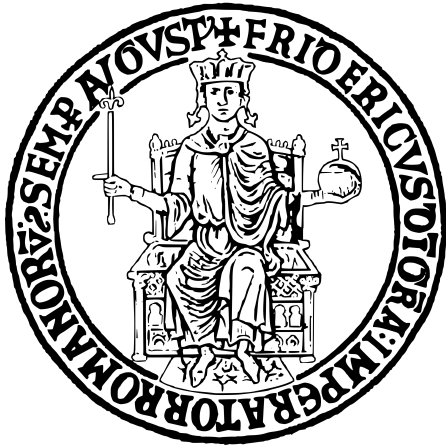


Elaborato di Basi di Dati

“Artemis Cartesio”



Professore: Vincenzo Moscato

Autori: Alessandro Cioffi N46006940
Antonio Cirino N46006930

a.a. 2024/2025

Contents

1	Introduzione	3
1.1	ArtemisCartesio	3
1.2	Specifiche del Sistema	3
1.3	Struttura del Progetto	3
1.4	Obiettivi	4
2	Creazione DB	5
2.1	Progettazione Concettuale	5
2.1.1	Modello E/R portante	5
2.1.2	Modello E/R completo	6
2.2	Progettazione Logica	8
2.2.1	Trasformazione	8
2.2.2	Traduzione	9
2.2.3	Modello E/R avanzato	10
2.3	Progettazione Fisica	12
2.4	Occupazione tabelle	12
2.4.1	Tabelle	13
2.4.2	Chiavi primarie	16
2.4.3	Chiavi esterne	16
2.4.4	Vincoli di check	17
3	Ottimizzazione DB	18
3.1	Indici	18
3.2	Concorrenza	18
3.3	Affidabilità	19
3.3.1	Backup	20
3.3.2	Recovery	20
4	Stored Procedures	22
4.1	Query	22
4.2	Views	22
4.3	Procedures	22
4.4	Triggers	22
5	Oracle APEX	23
5.1	Introduzione	23
5.2	Home	23
5.3	Procedure	24
5.4	Analisi dati	25

5.5	Altre views	27
5.6	Gestione della sicurezza	28

1 Introduzione

1.1 ArtemisCartesio

ArtemisCartesio rappresenta un'iniziativa pionieristica nel campo dell'esplorazione spaziale, sviluppata da un'agenzia spaziale internazionale per supportare missioni lunari avanzate. L'obiettivo principale è sfruttare tecnologie all'avanguardia per raccogliere, analizzare e gestire i dati provenienti da sensori, robot autonomi e membri dell'equipaggio impegnati in operazioni sulla superficie lunare. Questa piattaforma è progettata per affrontare le complessità logistiche e tecniche di missioni scientifiche, garantendo al contempo efficienza, sicurezza e affidabilità.

Il nome **Artemis Cartesio** unisce il simbolismo della dea Artemide, legata alla luna e all'esplorazione, con il razionalismo scientifico di Cartesio, rappresentando così l'equilibrio tra avventura e metodo. Inoltre, le iniziali **A.C.** omaggiano i creatori del progetto, conferendo un tocco personale al nome.

1.2 Specifiche del Sistema

Il progetto prevede lo sviluppo di una piattaforma informatica che integri:

- **Gestione delle missioni:** Archiviazione di dettagli quali obiettivo, stato, data di inizio e fine, e gestione delle risorse coinvolte.
- **Monitoraggio dei sensori:** Registrazione delle informazioni relative ai sensori (coordinate, tipo, stato operativo) e delle rilevazioni effettuate.
- **Gestione delle anomalie:** Identificazione e registrazione di anomalie, con dati relativi a data, ora, livello di priorità e causa.
- **Interventi e manutenzione:** Programmazione e tracciamento degli interventi per risolvere anomalie, con dettagli quali esito e descrizione delle operazioni effettuate.
- **Reportistica e analisi statistiche:** Compilazione di report da parte dei membri dell'equipaggio e analisi statistiche per ottimizzare le operazioni.

1.3 Struttura del Progetto

Per implementare il sistema, il progetto include le seguenti fasi principali:

- **Progettazione della base di dati:**

- *Concettuale*: Definizione delle entità, delle relazioni e degli attributi principali.
 - *Logica*: Traduzione dello schema concettuale in un modello relazionale.
 - *Fisica*: Ottimizzazione dello schema logico per l'implementazione in Oracle DBMS.
- **Ottimizzazione delle prestazioni:**
 - Creazione di indici per velocizzare le operazioni.
 - Progettazione di strategie di backup, recovery e replicazione per garantire l'affidabilità.
 - **Implementazione SQL:**
 - Creazione di stored procedure, trigger, query e viste per gestire le operazioni e supportare l'automazione.
 - **Interfaccia web-based:**
 - Sviluppo di un'interfaccia utente tramite Oracle APEX per semplificare l'interazione con il sistema.

1.4 Obiettivi

Il sistema è progettato per:

- Migliorare la gestione e il coordinamento delle missioni lunari.
- Consentire un monitoraggio avanzato e in tempo reale delle operazioni.
- Supportare l'analisi statistica dei dati raccolti per prendere decisioni informate.

Conclusione

Questo progetto non solo rappresenta un passo avanti nella gestione delle missioni spaziali, ma si pone come esempio di integrazione tra tecnologie avanzate e necessità operative. L'approccio metodologico adottato garantisce una base solida per affrontare sfide future nell'esplorazione lunare e spaziale.

2 Creazione DB

In questa sezione verrà descritta la progettazione e la creazione del database. Il database è stato progettato per soddisfare i requisiti del sistema di monitoraggio e controllo di missioni spaziali e in modo da garantire la corretta memorizzazione e gestione dei dati relativi a missioni, membri dell'equipaggio, sensori, robot, anomalie, rilevazioni e report.

2.1 Progettazione Concettuale

La progettazione concettuale è la fase iniziale del processo di progettazione di un database, in cui si definiscono i requisiti del sistema e si identificano le entità coinvolte, le relazioni tra di esse e gli attributi che le caratterizzano. Questa fase è indipendente dal modello logico e si concentra sulla rappresentazione dei dati in modo astratto, senza considerare i dettagli implementativi.

La progettazione concettuale è stata realizzata attraverso la modellazione del sistema tramite il modello Entità/Relazione (E/R). Questo modello consente di rappresentare in modo chiaro e intuitivo le entità coinvolte, le relazioni tra di esse e gli attributi che le caratterizzano.

2.1.1 Modello E/R portante

Come prima fase della progettazione concettuale, è stato realizzato uno schema E/R portante. Lo schema portante rappresenta il nucleo centrale del modello Entità-Relazione (E/R) per il sistema in esame. Esso evidenzia le principali entità coinvolte: **RISORSA**, **MISSIONE** e **MEMBRO EQUIPAGGIO**, connesse tra loro tramite relazioni chiave.

- La **RISORSA** rappresenta gli strumenti o gli elementi utilizzati nelle missioni.
- La **MISSIONE** è il fulcro operativo, dove vengono assegnate risorse e membri dell'equipaggio.
- Il **MEMBRO EQUIPAGGIO** indica le persone coinvolte nelle missioni, con specifici ruoli e compiti.

Le relazioni descrivono i collegamenti logici tra queste entità, evidenziando l'assegnazione e l'utilizzo di risorse e personale in contesto missione. Questo schema costituisce la base per la progettazione logica e fisica del sistema.



Figure 1: Schema portante del modello E/R

2.1.2 Modello E/R completo

A partire dallo schema portante descritto in precedenza, è stato realizzato uno schema E/R completo che include tutte le entità e le relazioni coinvolte nel sistema. Lo schema completo rappresenta in modo esaustivo tutte le entità, le relazioni e gli attributi che caratterizzano il sistema di monitoraggio e controllo di missioni spaziali.

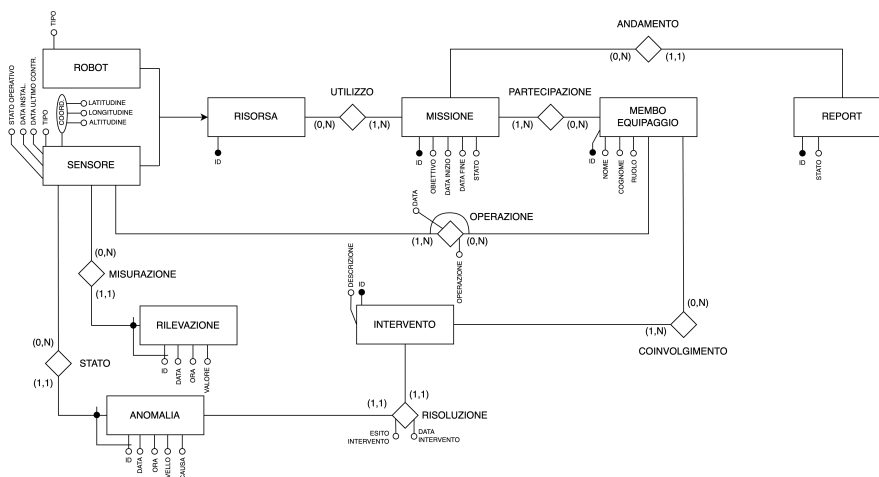


Figure 2: Modello E/R completo

Specifiche di Progettazione

- **MISSIONE:** L'entità **MISSIONE** è identificata da un *ID* e caratterizzata dagli attributi: *Obiettivo*, *Data di inizio*, *Data di fine* e *Stato*. Ogni **MISSIONE** è in relazione con i **MEMBRI DELL'EQUIPAGGIO** tramite l'associazione **PARTECIPAZIONE** con cardinalità $(1, N)$, poiché una missione può coinvolgere uno o più membri dell'equipaggio, mentre un membro può partecipare a zero o più missioni $(0, N)$. L'associazione è di tipo molti-a-molti. Ogni **MISSIONE** è anche in una relazione di **UTILIZZO** con **RISORSA**. Una **MISSIONE** può utilizzare $(1, N)$ risorse (non $(0, N)$ in quanto una missione non avrebbe senso di esistere se non vi fosse associata almeno una risorsa) e una **RISORSA** può essere usata da $(0, N)$ missioni.

- **MEMBRO DELL'EQUIPAGGIO:** L'entità **MEMBRO DELL'EQUIPAGGIO** è identificata da un *Codice univoco* e caratterizzata dagli attributi: *Nome*, *Cognome* e *Ruolo*. Ogni membro dell'equipaggio è responsabile della manutenzione e riparazione dei sensori, partecipando all'associazione **OPERAZIONE** con i **SENSORI** secondo cardinalità $(0, N)$ da parte dell'equipaggio e $(1, N)$ da parte dei sensori. Il tipo di operazione è specificato tramite l'attributo *Operazione* della relazione **OPERAZIONE**. L'attributo *Data* fa sì che un **MEMBRO DELL'EQUIPAGGIO** possa effettuare più operazioni dello stesso tipo sullo stesso sensore nel tempo. Un **MEMBRO DELL'EQUIPAGGIO** può inoltre essere coinvolto tramite la relazione **COINVOLGIMENTO** in $(0, N)$ **INTERVENTO**. Ogni **INTERVENTO** può coinvolgere $(1, N)$ membri.
- **INTERVENTO:** Ogni **INTERVENTO** è avviato per risolvere una specifica **ANOMALIA**. Gli attributi chiave di **INTERVENTO** includono un *ID*, una *Descrizione*, l'*Esito dell'intervento* e la *Data dell'intervento*. La relazione tra **ANOMALIA** e **INTERVENTO**, denominata **RISOLUZIONE**, ha cardinalità $(1, 1)$ su entrambi i lati, garantendo che ogni **INTERVENTO** risolva un'unica **ANOMALIA** e che ogni **ANOMALIA** sia risolta da un unico **INTERVENTO**.
- **REPORT:** Ad ogni **MISSIONE** possono essere associati $(0, N)$ **REPORT** tramite la relazione **ANDAMENTO**. Un **REPORT** è relativo a un'unica **MISSIONE** ed ha dunque cardinalità $(1, 1)$. Gli attributi di **REPORT** includono un *ID* ed uno *Stato*.
- **RILEVAZIONE:** Ogni **RILEVAZIONE** è associata esattamente a un $(1, 1)$ **SENSORE** tramite la relazione denominata **MISURAZIONE**. Un **SENSORE**, tuttavia, può effettuare più rilevazioni, con una cardinalità pari a $(0, N)$. L'entità **RILEVAZIONE** include una chiave esterna che la collega all'entità **SENSORE** tramite la relazione **MISURAZIONE**, basandosi sugli attributi *Data* e *Ora*. Questi attributi, combinati con l'*ID* del sensore, garantiscono l'identificazione univoca di ciascuna **RILEVAZIONE** effettuata da un determinato **SENSORE**. Si assume che un **SENSORE** non possa eseguire più rilevazioni nello stesso istante temporale definito dalla coppia $(Data, Ora)$.
- **ANOMALIA:** Ogni **ANOMALIA** è associata esattamente a un $(1, 1)$ **SENSORE** tramite la relazione denominata **STATO**. Un **SENSORE**, tuttavia, può presentare più anomalie, con una cardinalità pari a $(0, N)$. L'entità **ANOMALIA** include una chiave esterna che la collega all'entità **SENSORE** tramite la relazione **STATO**, basandosi sugli attributi *Data* e

Ora. Questi attributi, combinati con l'*ID* del sensore, garantiscono l'identificazione univoca di ciascuna **ANOMALIA** verificatasi in un determinato **SENSORE**. Si assume che in un **SENSORE** non possano verificarsi più anomalie nello stesso istante temporale definito dalla coppia (*Data*, *Ora*).

2.2 Progettazione Logica

La progettazione logica si articola in due fasi:

1. **Trasformazione:** in questa fase, vengono rimossi tutti i costrutti del modello Entità/Relazione (E/R) che non sono direttamente traducibili nel modello logico, come gli attributi composti e gli attributi multi-valore. Gli attributi multi-valore vengono associati direttamente all'entità di partenza, mentre gli attributi composti vengono scomposti nei loro componenti e, se necessario, trasferiti a una nuova entità collegata all'entità originale.
2. **Traduzione:** lo schema risultante dalla trasformazione viene convertito nel modello logico attraverso un insieme di regole predeterminate, che possono essere implementate anche tramite strumenti automatizzati. Questa fase non considera direttamente la semantica dei dati, ma si concentra sulla loro struttura.

2.2.1 Trasformazione

Durante la fase di trasformazione, vengono eliminati tutti gli attributi che non sono direttamente traducibili nel modello logico. Di seguito vengono descritti i casi specifici presenti nello schema:

- **Attributi multi-valore:** non sono presenti in questo caso, quindi non si rende necessaria alcuna operazione di trasformazione relativa a questa tipologia di attributi.
- **Attributi composti:** l'unico attributo composto identificato è **Coordinate**, associato all'entità *SENSORE*. Per conformarsi ai requisiti del modello logico, questo attributo è stato scomposto nei suoi componenti: **Latitudine**, **Longitudine** e **Altitudine**. Tali componenti sono stati direttamente associati all'entità di partenza senza creare una nuova entità.

Di seguito è riportato lo schema trasformato per l'entità *SENSORE*:

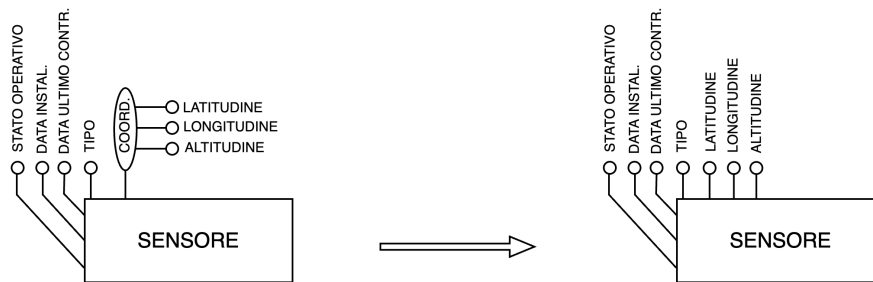


Figure 3: Entità *SENSORE* trasformata

2.2.2 Traduzione

La fase di traduzione consiste nella conversione dello schema Entità/Relazione (E/R), risultato della trasformazione, in uno schema relazionale conforme al modello logico relazionale. Questa operazione garantisce la corretta implementazione delle entità, delle relazioni e degli attributi all'interno del database relazionale.

Traduzione Entità Ogni entità del modello E/R diventa una relazione/tabella.

- **Nome della tabella:** corrisponde al nome dell'entità.
- **Campi della tabella:** corrispondono agli attributi dell'entità.

Risultato della Traduzione delle Entità:

```
MISSIONI(ID, Obiettivo, Data_Inizio, Data_Fine, Stato);
MEMBI(ID, Nome, Cognome, Ruolo);
REPORT(ID, Stato);
INTERVENTI(ID, Descrizione);
ANOMALIE(ID, Data, Ora, Livello, Causa);
RILEVAZIONI(ID, Data, Ora, Valore);
ROBOT(ID, Tipo);
SENSORI(ID, Data_Installazione, Data_Ultimo_Controllo, Tipo,
        Stato_Operativo, Latitudine, Longitudine, Altitudine)
```

Traduzione Relazioni

1. Relazioni N a N

- Ogni associazione N a N diventa una tabella con:
 - **Nome:** corrisponde al nome dell'associazione, al plurale.
 - **Campi:** includono gli identificatori delle due entità che collega, più eventuali attributi dell'associazione.

- **Chiave primaria:** composta dalla coppia dei due identificatori.
- **Vincoli di integrità referenziale:** garantiscono la consistenza con le entità collegate.

```
UTILIZZO_SENSORI (Sensore: Sensori, Missione: Missioni);
UTILIZZO_ROBOT (Robot: Robot, Missione: Missioni);
PARTECIPAZIONI (Missione: Missioni, Membro_Equipaggio:
    Membri_Equipaggio);
OPERAZIONI (Membro_Equipaggio: Membri_Equipaggio, Sensore
    : Sensori, Operazione, Data);
COINVOLGIMENTI (Membro_Equipaggio: Membri_Equipaggio,
    Intervento: Interventi)
```

2. Relazioni 1 a N

- Gli **attributi dell'entità lato 1** e gli **attributi della relazione** vengono aggiunti come campi all'entità lato N.
- **Chiave primaria:** rimane quella dell'entità lato N.
- Questa scelta consente di ridurre il numero di tabelle, evitando join complessi a 3 tabelle.

```
REPORT (ID, Stato, Data, Missione: Missioni);
RILEVAZIONI (ID, Data, Ora, Valore, Sensore: Sensori);
ANOMALIE (ID, Data, Ora, Livello, Causa, Sensore: Sensori
    );
```

3. Relazioni 1 a 1

- Ogni associazione 1 a 1 diventa una tabella con:
 - **Campi:** includono gli identificatori delle entità che collega, più eventuali attributi.
 - **Chiave primaria:** si sceglie l'identificatore dell'entità con cardinalità minima e partecipazione obbligatoria, per evitare valori NULL.

```
RISOLUZIONI (Intervento: Interventi, Anomalia: Anomalie,
    Esito_Intervento, Data_Intervento)
```

2.2.3 Modello E/R avanzato

Le gerarchie di generalizzazione/specializzazione non possono essere direttamente rappresentate nel modello logico relazionale, poiché quest'ultimo non prevede un

costrutto equivalente. Per superare questa limitazione, il modello E/R è stato esteso con l'introduzione del costrutto di **generalizzazione/specializzazione**.

Il costrutto di generalizzazione può essere trasformato in schemi traducibili nel modello logico seguendo tre modalità principali.

Per questo progetto è stata adottata la modalità di **accorpamento della super-classe nelle sottoclassi**:

- **Eliminazione dell'entità padre:** l'entità padre (superclasse) viene eliminata dallo schema.
- **Eredità degli attributi e delle relazioni:** grazie alla proprietà dell'eredità, gli attributi, l'identificatore e le relazioni a cui partecipava l'entità padre vengono trasferiti integralmente alle entità figlie (sottoclassi).

In particolare la specializzazione è di tipo totale-disgiunta:

- **Totale:** Ogni istanza della superclasse deve appartenere ad almeno una sottoclasse. Ogni risorsa deve essere necessariamente o un "sensore" o "robot".
- **Disgiunta:** Un'istanza della superclasse può appartenere a una sola sottoclasse alla volta. Una risorsa può essere un "sensore" o "robot", ma non entrambi contemporaneamente.

In figura (Figura 4) è riportato il risultato dell'accorpamento relativo alla super-classe *RISORSA* e alle sottoclassi *ROBOT* e *SENSORE*.

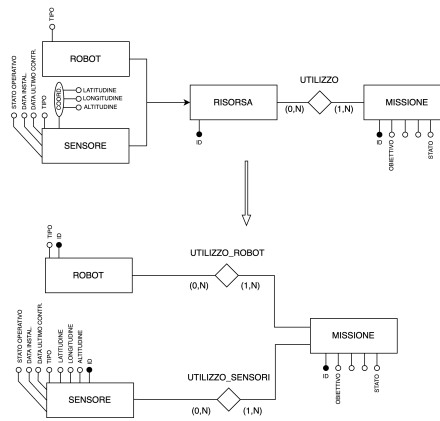


Figure 4: accorpamento di *RISORSA* in *ROBOT* e *SENSORE*

2.3 Progettazione Fisica

I tipi di dato utilizzati per la realizzazione di questo progetto sono:

- **DATE**: Utilizzato per tutte le date presenti.
- **INTEGER**: per tutti gli ID che quindi sono formati esclusivamente da numeri e non da lettere;
- **VARCHAR2(N)**: per tutti gli attributi di tipo testuali, come per esempio nome, cognome, descrizione, ecc..
- **FLOAT**: Utilizzato per rappresentare valori numerici con la necessità di precisione decimale, ad esempio coordinate geografiche (latitudine, longitudine, altitudine) o valori rilevati dai sensori.
- **TIMESTAMP**: Utilizzato per rappresentare data e ora in modo dettagliato, includendo secondi e frazioni di secondo, ad esempio per registrare eventi come rilevazioni o anomalie con un'accurata marcatura temporale.

2.4 Occupazione tabelle

Viene ora presentata una stima dell'occupazione in byte per le tabelle di un database progettato per un DBMS Oracle. Ogni tabella è analizzata in base ai tipi di dati delle sue colonne.

Tipi di Dati e Dimensioni

- **INT**: 4 byte
- **VARCHAR2(n)**: fino a n byte
- **DATE**: 7 byte
- **TIMESTAMP**: 11 byte
- **FLOAT**: 8 byte

Le dimensioni calcolate sono espresse in byte per riga e in MB per il numero stimato di righe.

- **MISSIONI**: $4 + 255 + 7 + 7 + 50 = 323$ byte, 1.000 righe $\approx 0,31$ MB
- **MEMBRI**: $4 + 100 + 100 + 100 = 304$ byte, 500 righe $\approx 0,14$ MB

- **SENSORI:** $4 + 7 + 7 + 100 + 50 + 8 + 8 + 8 = 192$ byte, 300 righe $\approx 0,05$ MB
- **ROBOT:** $4 + 100 = 104$ byte, 100 righe $\approx 0,01$ MB
- **ANOMALIE:** $4 + 7 + 11 + 50 + 255 + 4 = 331$ byte, 2.000 righe $\approx 0,63$ MB
- **INTERVENTI:** $4 + 255 = 259$ byte, 1.000 righe $\approx 0,25$ MB
- **RISOLUZIONI:** $4 + 4 + 255 + 7 = 270$ byte, 1.000 righe $\approx 0,26$ MB
- **RILEVAZIONI:** $4 + 7 + 11 + 8 + 4 = 34$ byte, 10.000 righe $\approx 0,32$ MB
- **REPORT:** $4 + 50 + 7 + 4 = 65$ byte, 500 righe $\approx 0,03$ MB
- **UTILIZZO_ROBOT:** $4 + 4 = 8$ byte, 1.000 righe $\approx 0,01$ MB
- **UTILIZZO_SENSORI:** $4 + 4 = 8$ byte, 1.000 righe $\approx 0,01$ MB
- **COINVOLGIMENTI:** $4 + 4 = 8$ byte, 500 righe $\approx 0,004$ MB
- **OPERAZIONI:** $4 + 4 + 255 + 7 = 270$ byte, 2.000 righe $\approx 0,51$ MB
- **PARTECIPAZIONI:** $4 + 4 = 8$ byte, 1.000 righe $\approx 0,01$ MB

Le dimensioni teoriche sono calcolate considerando i tipi di dati e ignorando eventuali overhead di gestione. Per una stima complessiva, è necessario moltiplicare queste dimensioni per il numero di righe previste in ciascuna tabella, includendo un margine del 10-20% per l'indicizzazione e altri metadati.

2.4.1 Tabelle

```
-- Tabella MISSIONI
CREATE TABLE MISSIONI (
    ID INT,
    Obiettivo VARCHAR2(255) NOT NULL,
    Data_Inizio DATE NOT NULL, --NOT NULL o no ?
    Data_Fine DATE,
    Stato VARCHAR2(50) NOT NULL
);

-- Tabella MEMBRI
CREATE TABLE MEMBRI (
    ID INT,
    Nome VARCHAR2(100) NOT NULL,
    Cognome VARCHAR2(100) NOT NULL,
    Ruolo VARCHAR2(100) NOT NULL
);
```

```

-- Tabella SENSORI
CREATE TABLE SENSORI (
    ID INT,
    Data_Installazione DATE NOT NULL,
    Data_Ultimo_Controllo DATE,
    Tipo VARCHAR2(100) NOT NULL,
    Latitudine FLOAT NOT NULL,
    Longitudine FLOAT NOT NULL,
    Altitudine FLOAT NOT NULL
);

-- Tabella ROBOT
CREATE TABLE ROBOT (
    ID INT,
    Tipo VARCHAR2(100) NOT NULL
);

-- Tabella ANOMALIE
CREATE TABLE ANOMALIE (
    ID INT,
    Data DATE NOT NULL,
    Ora TIMESTAMP NOT NULL,
    Livello VARCHAR2(50) NOT NULL,
    Causa VARCHAR2(255) NOT NULL,
    Sensori INT
);

-- Tabella INTERVENTI
CREATE TABLE INTERVENTI (
    ID INT,
    Descrizione VARCHAR2(255) NOT NULL
);

-- Tabella RISOLUZIONI
CREATE TABLE RISOLUZIONI (
    Anomalie INT,
    Interventi INT,
    Esito_Intervento VARCHAR2(255),
    Data_Intervento DATE
);

-- Tabella RILEVAZIONI
CREATE TABLE RILEVAZIONI (
    ID INT,
    Data DATE NOT NULL,
    Ora TIMESTAMP NOT NULL,
    Valore FLOAT NOT NULL,
    Sensori INT
);

```

```

);

-- Tabella REPORT
CREATE TABLE REPORT (
    ID INT,
    Stato VARCHAR2(50) NOT NULL,
    Missioni INT
);

-- Tabella UTILIZZO_ROBOT
CREATE TABLE UTILIZZO_ROBOT (
    Robot INT,
    Missioni INT
);

-- Tabella UTILIZZO_SENSORI
CREATE TABLE UTILIZZO_SENSORI (
    Sensori INT,
    Missioni INT
);

-- Tabella COINVOLGIMENTI
CREATE TABLE COINVOLGIMENTI (
    Membri INT,
    Interventi INT
);

-- Tabella OPERAZIONI
CREATE TABLE OPERAZIONI (
    Membri INT,
    Sensori INT,
    Stato_Operativo VARCHAR2(50) NOT NULL,
    Operazione VARCHAR2(255)
);

-- Tabella PARTECIPAZIONI
CREATE TABLE PARTECIPAZIONI (
    Missione INT,
    Membri INT
);

-- Tabella SENSORI_MISSIONI
CREATE TABLE SENSORI_MISSIONI (
    Sensore_ID INT,
    Missione_ID INT,
    PRIMARY KEY (Sensore_ID, Missione_ID),
    FOREIGN KEY (Sensore_ID) REFERENCES SENSORI(ID),
    FOREIGN KEY (Missione_ID) REFERENCES MISSIONI(ID)
);

```



```
-- Tabella MEMBRI_MISSIONI
CREATE TABLE MEMBRI_MISSIONI (
    Membro_ID INT,
    Missione_ID INT,
    PRIMARY KEY (Membro_ID, Missione_ID),
    FOREIGN KEY (Membro_ID) REFERENCES MEMBRI(ID),
    FOREIGN KEY (Missione_ID) REFERENCES MISSIONI(ID)
);
```

2.4.2 Chiavi primarie

```
ALTER TABLE MISSIONI ADD CONSTRAINT PK_MISSIONI PRIMARY KEY (ID);
ALTER TABLE MEMBRI ADD CONSTRAINT PK_MEMBRI PRIMARY KEY (ID);
ALTER TABLE SENSORI ADD CONSTRAINT PK_SENSORI PRIMARY KEY (ID);
ALTER TABLE ROBOT ADD CONSTRAINT PK_ROBOT PRIMARY KEY (ID);
ALTER TABLE ANOMALIE ADD CONSTRAINT PK_ANOMALIE PRIMARY KEY (ID);
ALTER TABLE INTERVENTI ADD CONSTRAINT PK_INTERVENTI PRIMARY KEY (
    ID);
ALTER TABLE RILEVAZIONI ADD CONSTRAINT PK_RILEVAZIONI PRIMARY KEY
    (ID);
ALTER TABLE REPORT ADD CONSTRAINT PK_REPORT PRIMARY KEY (ID);
ALTER TABLE RISOLUZIONI ADD CONSTRAINT PK_RISOLUZIONI PRIMARY KEY
    (Anomalie, Interventi);
ALTER TABLE UTILIZZO_ROBOT ADD CONSTRAINT PK_UTILIZZO_ROBOT
    PRIMARY KEY (Robot, Missioni);
ALTER TABLE UTILIZZO_SENSORI ADD CONSTRAINT PK_UTILIZZO_SENSORI
    PRIMARY KEY (Sensori, Missioni);
ALTER TABLE COINVOLGIMENTI ADD CONSTRAINT PK_COINVOLGIMENTI
    PRIMARY KEY (Membri, Interventi);
ALTER TABLE OPERAZIONI ADD CONSTRAINT PK_OPERAZIONI PRIMARY KEY (
    Membri, Sensori);
ALTER TABLE PARTECIPAZIONI ADD CONSTRAINT PK PARTECIPAZIONI
    PRIMARY KEY (Missione, Membri);
```

2.4.3 Chiavi esterne

```
ALTER TABLE ANOMALIE ADD CONSTRAINT FK_ANOMALIE_SENSORI FOREIGN
    KEY (Sensori) REFERENCES SENSORI(ID);
ALTER TABLE RISOLUZIONI ADD CONSTRAINT FK_RISOLUZIONI_ANOMALIE
    FOREIGN KEY (Anomalie) REFERENCES ANOMALIE(ID);
ALTER TABLE RISOLUZIONI ADD CONSTRAINT FK_RISOLUZIONI_INTERVENTI
    FOREIGN KEY (Interventi) REFERENCES INTERVENTI(ID);
ALTER TABLE RILEVAZIONI ADD CONSTRAINT FK_RILEVAZIONI_SENSORI
    FOREIGN KEY (Sensori) REFERENCES SENSORI(ID);
```

```

ALTER TABLE REPORT ADD CONSTRAINT FK_REPORT_MISSIONI FOREIGN KEY (
    Missioni) REFERENCES MISSIONI(ID);
ALTER TABLE UTILIZZO_ROBOT ADD CONSTRAINT FK_UTILIZZO_ROBOT_ROBOT
    FOREIGN KEY (Robot) REFERENCES ROBOT(ID);
ALTER TABLE UTILIZZO_ROBOT ADD CONSTRAINT
    FK_UTILIZZO_ROBOT_MISSIONI FOREIGN KEY (Missioni) REFERENCES
    MISSIONI(ID);
ALTER TABLE UTILIZZO_SENSORI ADD CONSTRAINT
    FK_UTILIZZO_SENSORI_SENSORI FOREIGN KEY (Sensori) REFERENCES
    SENSORI(ID);
ALTER TABLE UTILIZZO_SENSORI ADD CONSTRAINT
    FK_UTILIZZO_SENSORI_MISSIONI FOREIGN KEY (Missioni) REFERENCES
    MISSIONI(ID);
ALTER TABLE COINVOLGIMENTI ADD CONSTRAINT FK_COINVOLGIMENTI_MEMBRI
    FOREIGN KEY (Membri) REFERENCES MEMBRI(ID);
ALTER TABLE COINVOLGIMENTI ADD CONSTRAINT
    FK_COINVOLGIMENTI_INTERVENTI FOREIGN KEY (Interventi)
    REFERENCES INTERVENTI(ID);
ALTER TABLE OPERAZIONI ADD CONSTRAINT FK_OPERAZIONI_MEMBRI FOREIGN
    KEY (Membri) REFERENCES MEMBRI(ID);

ALTER TABLE OPERAZIONI ADD CONSTRAINT FK_OPERAZIONI_SENSORI
    FOREIGN KEY (Sensori) REFERENCES SENSORI(ID);

ALTER TABLE PARTECIPAZIONI ADD CONSTRAINT
    FK_PORTECIPAZIONI_MISSIONE FOREIGN KEY (Missione) REFERENCES
    MISSIONI(ID);

ALTER TABLE PARTECIPAZIONI ADD CONSTRAINT FK_PORTECIPAZIONI_MEMBRI
    FOREIGN KEY (Membri) REFERENCES MEMBRI(ID);

```

2.4.4 Vincoli di check

```

ALTER TABLE SENSORI ADD CONSTRAINT CK_SENSORI_TIPO CHECK (Tipo
    IN ('Temperatura', 'Pressione', 'Gas', 'Radiazioni', '
    Geologia'));

ALTER TABLE OPERAZIONI ADD CONSTRAINT CK_OPERAZIONI_STATO
    CHECK (Stato_Operativo IN ('Attivo', 'Standby', '
    Manutenzione', 'Malfunzionante'));

ALTER TABLE ANOMALIE ADD CONSTRAINT CK_ANOMALIE_LIVELLO CHECK
    (Livello IN ('Bassa', 'Media', 'Alta', 'Critica'));

ALTER TABLE MISSIONI ADD CONSTRAINT CK_MISSIONI_STATO CHECK (
    Stato IN ('Pianificata', 'In corso', 'Completata', '
    Annullata'));

```

3 Ottimizzazione DB

3.1 Indici

Un **indice** in un sistema di gestione di database (DBMS) è una struttura dati organizzata che consente di individuare rapidamente un determinato record all'interno di un file di dati. Uno dei principali vantaggi dell'utilizzo degli indici è il miglioramento delle prestazioni delle query: gli indici permettono di ridurre il tempo necessario per cercare i dati, evitando una scansione completa della tabella.

Creare indici sui campi che vengono frequentemente utilizzati nei filtri delle query (ad esempio, con condizioni `WHERE`, `JOIN`, `ORDER BY` o `GROUP BY`) può velocizzare significativamente l'elaborazione delle richieste, ottimizzando il sistema nel suo complesso.

Tuttavia, è importante bilanciare l'uso degli indici per evitare costi aggiuntivi durante le operazioni di scrittura come `INSERT`, `UPDATE` e `DELETE`, poiché gli indici devono essere aggiornati ogni volta che i dati della tabella vengono modificati.

Tralasciando gli indici sulla chiave primaria, in quanto il DBMS crea un indice per ogni chiave primaria della tabella, abbiamo pensato di aggiungere questi indici:

```
CREATE INDEX idx_missioni_stato ON MISSIONI(Stato);
-- utile per filtrare missioni sullo stato

CREATE INDEX idx_membri_ruolo ON MEMBRI(Ruolo);
-- utile per filtrare sul ruolo dei membri

CREATE INDEX idx_robot_tipo ON ROBOT(Tipo);
-- utile per filtrare sul tipo di robot

CREATE INDEX idx_report_data ON REPORT(Data);

CREATE INDEX idx_rilevazioni_sensori_data ON RILEVAZIONI(Sensori,
    Data);
```

3.2 Concorrenza

Per la gestione della concorrenza abbiamo scelto di adottare il protocollo **2PL stretto**, una variante del **2PL**. Entrambi i protocolli si basano sul meccanismo del **lock**, che consente di gestire la concorrenza e garantire la **serializzabilità delle transazioni**.

Il suo funzionamento prevede:

- `lock()`: ogni oggetto è protetto da un lock;

- **read_lock()**: se una transazione vuole effettuare una lettura su uno oggetto. Più transazioni possono leggere contemporaneamente lo stesso oggetto (condivisione).
- **write_lock()**: è esclusivo e consente a una sola transazione di modificare l'oggetto per volta.
- **unlock()**: Ogni lock, una volta terminata l'operazione, deve essere rilasciato (unlock).

Gli oggetti possono trovarsi in tre stati: **libero**, **bloccato in lettura**, o **bloccato in scrittura**.

La scelta del **2PL stretto** rispetto al **2PL** è dovuta al fatto che il **2PL stretto** evita l'anomalia delle **letture sporche** (dirty reads), che possono verificarsi in altri approcci di gestione della concorrenza.

Il 2PL stretto prevede due fasi:

- **fase crescente**: è la fase in cui una transazione acquisisce mediante **read_lock** e **write_lock** tutte le risorse su cui dovrà effettuare operazioni di lettura e scrittura.
- **fase decrescente**: è la fase in cui attraverso l'**unlock** si vanno a rilasciare le risorse. In particolare questa fase nel caso di 2PL stretto può essere effettuata solo se la transazione termina con una commit o con un abort.

Questo approccio garantisce che le transazioni siano serializzabili, impedendo conflitti e mantenendo la consistenza dei dati.

3.3 Affidabilità

Il controllo di **affidabilità** in un sistema di basi di dati ha come obiettivo principale il **ripristino** dello stato corretto del sistema (recovery) in seguito a guasti accidentali o intenzionali, che possano compromettere la funzionalità del sistema stesso. I guasti possono essere legati sia a malfunzionamenti hardware (ad esempio, guasti su disco o memoria) che software (ad esempio, crash di applicazioni o errori di sistema).

Il sistema di affidabilità si basa sulla gestione delle **transazioni**, che sono le unità fondamentali delle operazioni nel database, garantendo **atomicità** (le transazioni sono eseguite in modo completo o non eseguite affatto) e **persistenza** (i dati delle transazioni devono essere memorizzati in modo permanente una volta che la transazione è stata completata correttamente).

3.3.1 Backup

Per garantire un alto livello di affidabilità, il nostro sistema di database implementa la strategia di backup RAID 1, che offre una soluzione di mirroring. In un sistema RAID 1, ogni dato scritto sul disco primario viene duplicato in tempo reale su un disco secondario, chiamato "mirror". Questa tecnica garantisce che, in caso di guasto di uno dei dischi, i dati siano ancora disponibili sull'altro disco, riducendo il rischio di perdita di informazioni e migliorando la disponibilità del sistema.

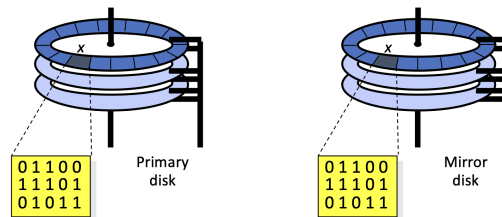


Figure 5: RAID 1 Mirroring

3.3.2 Recovery

Il gestore dell'affidabilità deve gestire l'esecuzione dei comandi transazionali di `begin transaction`, `commit`, `rollback` e tutte le operazioni di ripristino dopo i guasti.

Per poter effettuare ciò, il gestore deve possedere un file di log: un file presente su memoria stabile che registra tutte le operazioni svolte dalle transazioni nel loro ordine di esecuzione.

Il log è quindi una sorta di "diario di bordo" che, in un qualsiasi istante, permette di ricostituire il contenuto corretto della base dei dati a seguito di mal-funzionamenti.

```
DP, B(T1,-, -, -), U(T1, -, qtaP, 100, 90), U(T1, -, qtaC, NULL, 10), C(T1, -, -, -),
B(T2, -, -, -, -), CK(T2), U(T2, -, qtaP, 90, 70), U(T1, -, qtaC, NULL, 20), C(T2, -, -, -),
B(T3, -, -, -, -), U(T3, -, qtaP, 100, 90), U(T3, -, qtaC, NULL, 10), C(T3, -, -, -),
...
```

Figure 6: File di log

Tecniche di recovery

- **Ripresa a freddo:** Nel caso di guasti hard sui dispositivi di memoria di massa (es. guasti ai dischi rigidi) si perde sia la memoria centrale che quella

secondaria, ma la memoria stabile (come i dispositivi di backup) rimane intatta. In queste situazioni, viene effettuata la ripresa a freddo (cold restart), che richiede un ripristino più approfondito, attingendo ai backup e ai log per recuperare i dati persi.

- **Ripresa a caldo:** Nel caso di guasti soft (es. errori di programma, crash di sistema, caduta di tensione, ecc.) si perde il contenuto della sola memoria centrale (mentre rimangono intatte la memoria secondaria e quella stabile). In tali situazioni, viene effettuata la cosiddetta ripresa a caldo (warm restart).

In entrambi i casi, la procedura di ripristino avviene nelle seguenti fasi (modello fail-stop):

1. si forza l'arresto completo delle transazioni attive sul sistema di basi di dati;
2. viene ripristinato il corretto funzionamento del sistema operativo;
3. viene effettuata la procedura di ripristino.

4 Stored Procedures

4.1 Query

4.2 Views

4.3 Procedures

4.4 Triggers

5 Oracle APEX

5.1 Introduzione

Oracle APEX (Application Express) è una piattaforma di sviluppo applicativo low-code che consente di creare applicazioni web scalabili, sicure e altamente performanti utilizzando il database Oracle come base.

Con Oracle APEX, è possibile sfruttare strumenti integrati per la creazione di interfacce utente, la gestione dei dati e la personalizzazione delle funzionalità, rendendo il processo di sviluppo rapido ed efficiente. È particolarmente utile per automatizzare processi aziendali, creare report interattivi e implementare soluzioni personalizzate su misura per le esigenze delle organizzazioni.

5.2 Home

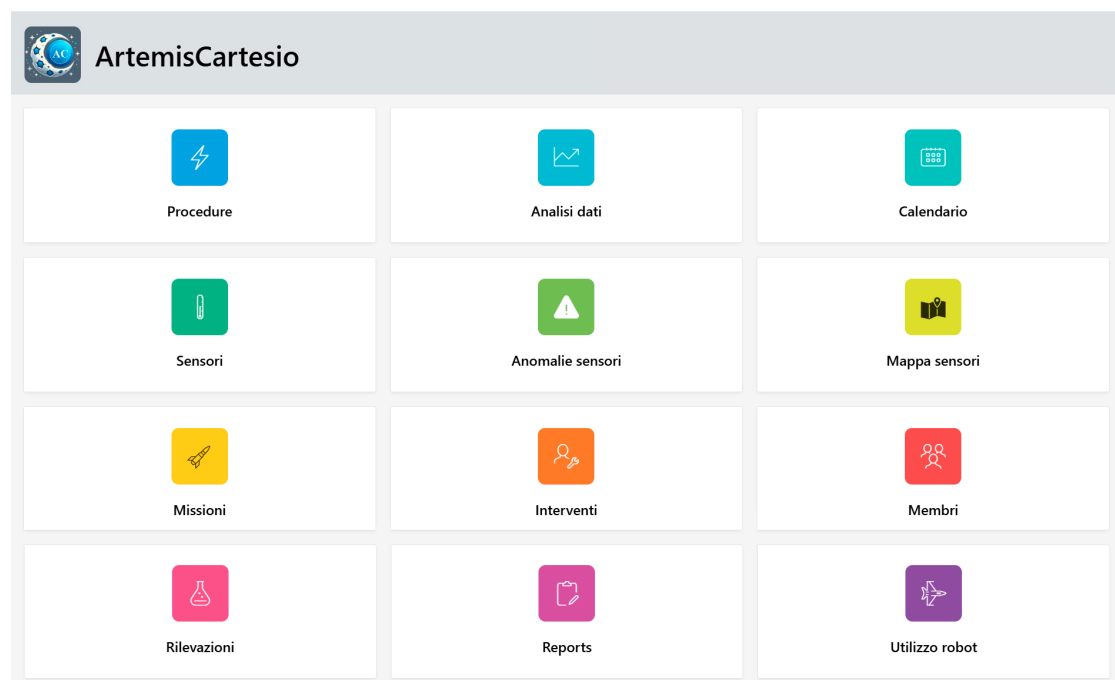


Figure 7: Home page dell'applicazione

In figura (Figura 7) è rappresentata la schermata principale dell'applicazione, progettata per gestire e monitorare ogni aspetto di una missione spaziale. La home page presenta una serie di funzionalità principali, ciascuna accessibile tramite icone intuitive e facilmente identificabili, che semplificano la navigazione e l'interazione con il sistema.

L'interfaccia è user-friendly e ben strutturata, progettata per garantire l'efficienza e la facilità d'uso per i membri dell'equipaggio della missione. Grazie all'organizzazione intuitiva, si può accedere rapidamente alle informazioni necessarie e ai moduli specifici per completare i propri compiti.

5.3 Procedure

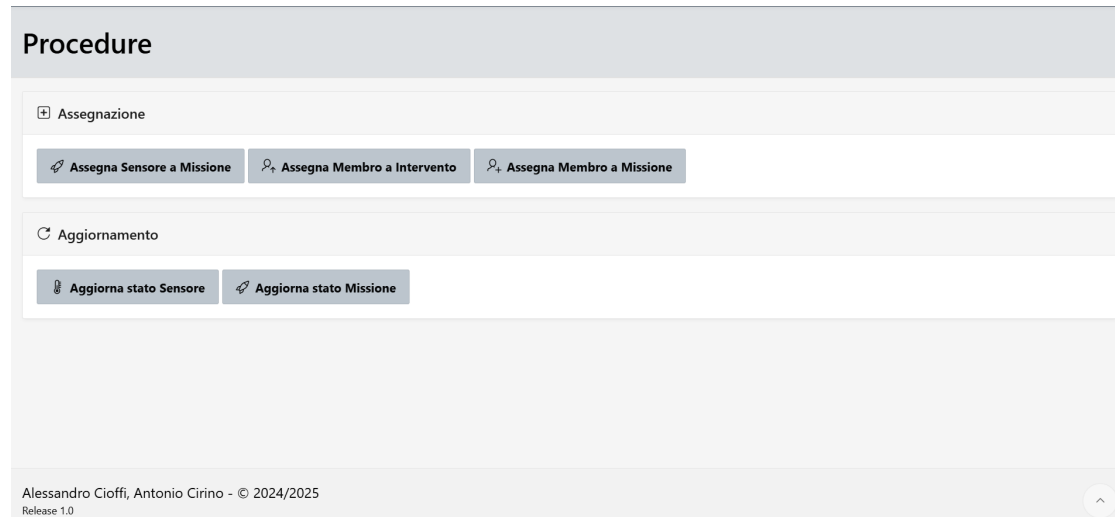


Figure 8: Schermata della sezione *Procedure*

La schermata *Procedure* è divisa in due sezioni: **Assegnazione** e **Aggiornamento**.

Assegnazione Questa sezione permette di gestire l'allocazione delle risorse e del personale per le varie attività. Le funzionalità principali includono:

- **Assegna Sensore a Missione:** consente di associare specifici sensori a una missione attiva.
- **Assegna Membro a Intervento:** permette di designare un membro del team per un determinato intervento tecnico (Figura 9).
- **Assegna Membro a Missione:** utilizzato per assegnare personale a una missione specifica.

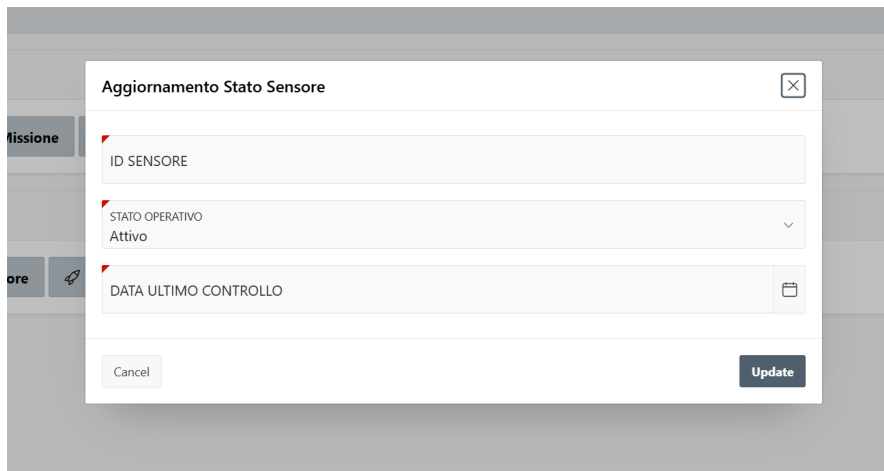
Aggiornamento In questa sezione si possono effettuare aggiornamenti sullo stato dei sensori e delle missioni, garantendo che le informazioni siano sempre accurate e aggiornate. Le opzioni includono:

- **Aggiorna stato Sensore:** modifica dello stato operativo di un sensore (attivo, malfunzionante, manutenzione, stand by) e della data dell'ultimo controllo (Figura 10).
- **Aggiorna stato Missione:** consente di aggiornare lo stato di avanzamento di una missione (pianificata, in corso, completata, annullata).



The dialog box is titled "Assegna Membro a Intervento". It contains two text input fields: "ID MEMBRO" and "ID INTERVENTO". Below these fields are two buttons: "Cancel" on the left and "Create" on the right.

Figure 9: Finestra di dialogo per l'assegnazione di un membro ad un intervento



The dialog box is titled "Aggiornamento Stato Sensore". It contains three input fields: "ID SENSORE", "STATO OPERATIVO" (which is a dropdown menu currently showing "Attivo"), and "DATA ULTIMO CONTROLLO" (which has a calendar icon to its right). Below these fields are two buttons: "Cancel" on the left and "Update" on the right.

Figure 10: Finestra di dialogo per l'aggiornamento di un sensore

5.4 Analisi dati

L'immagine (Figura 11) mostra la sezione **Analisi Dati** dell'applicazione "Missione Lunare", che fornisce una panoramica visiva sulle principali informazioni relative alla missione. La schermata include diversi grafici, tra cui:

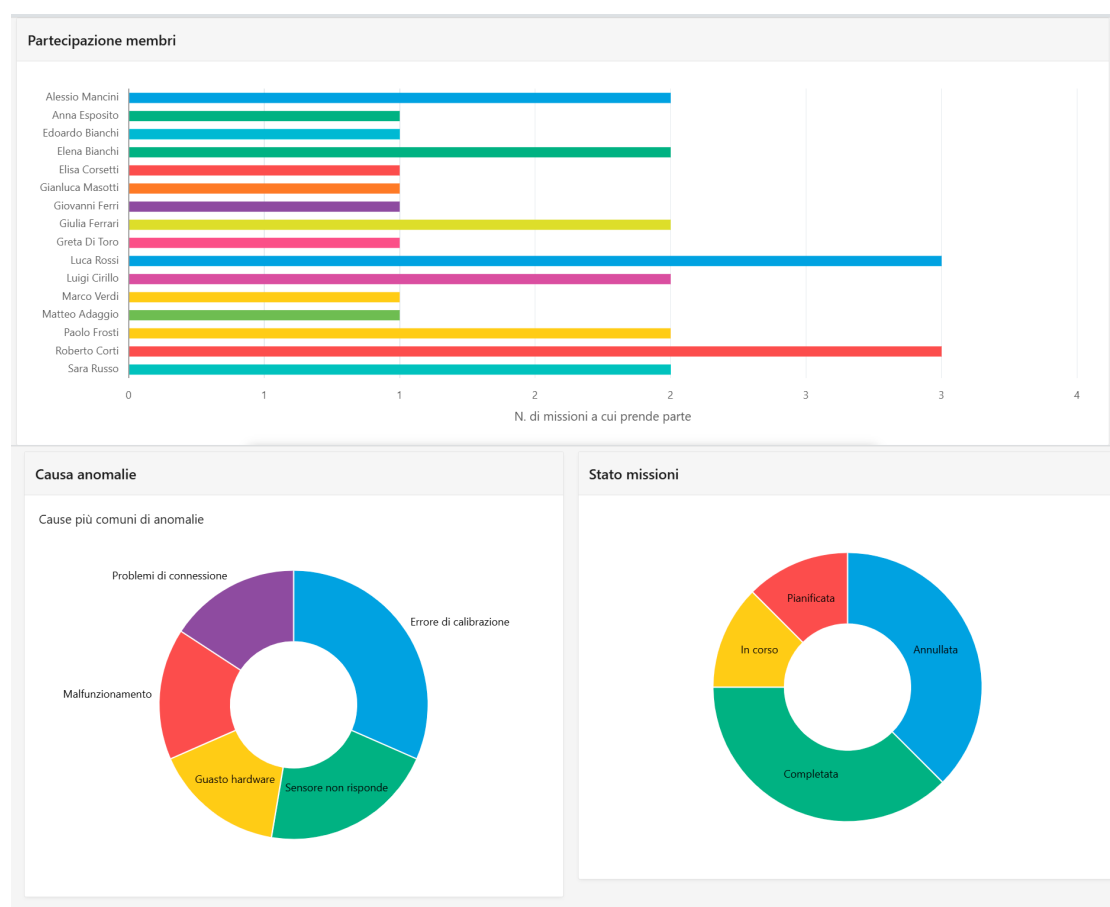


Figure 11: Schermata della sezione *Analisi dati*

Partecipazione membri Un grafico a barre orizzontali che mostra il numero di missioni a cui ciascun membro del team partecipa. Questo consente di visualizzare rapidamente il grado di coinvolgimento dei membri nelle attività.

Causa anomalie Un grafico a ciambella che rappresenta le cause più comuni di anomalie registrate durante le missioni.

Stato missioni Un secondo grafico a ciambella che evidenzia la distribuzione delle missioni in base al loro stato attuale.

Questa sezione offre una visione chiara e intuitiva per monitorare il progresso e le criticità delle missioni, supportando un processo decisionale informato.

5.5 Altre views

Di seguito sono riportate altre sezioni presenti nell'applicazione:

Missioni

Stato

☐ Annullata (3)

☐ Completata (3)

☐ In corso (1)

☐ Pianificata (1)

Total Row Count 8

Id

Obiettivo

Data Inizio

Data Fine

Stato

1	Esplorazione Polo Sud Lunare	15/06/2025	20/12/2025	Annullata
2	Studio Crateri Lunari	10/03/2024		Completata
3	Raccolta Campioni di Rocce	05/11/2023	15/02/2024	Completata
4	Installazione Base Avanzata	01/05/2024		Annullata
5	Esplorazione del suolo marziano	01/01/2024	30/06/2024	Completata
6	Monitoraggio della pressione atmosferica	01/07/2024		In corso
7	Ricerca perdite di gas	01/01/2025		Pianificata
8	Controllo radiazioni	01/07/2025		Annullata

Alessandro Cioffi, Antonio Cirino - © 2024/2025

Release 1.0

Figure 12: Schermata della sezione *Missioni*

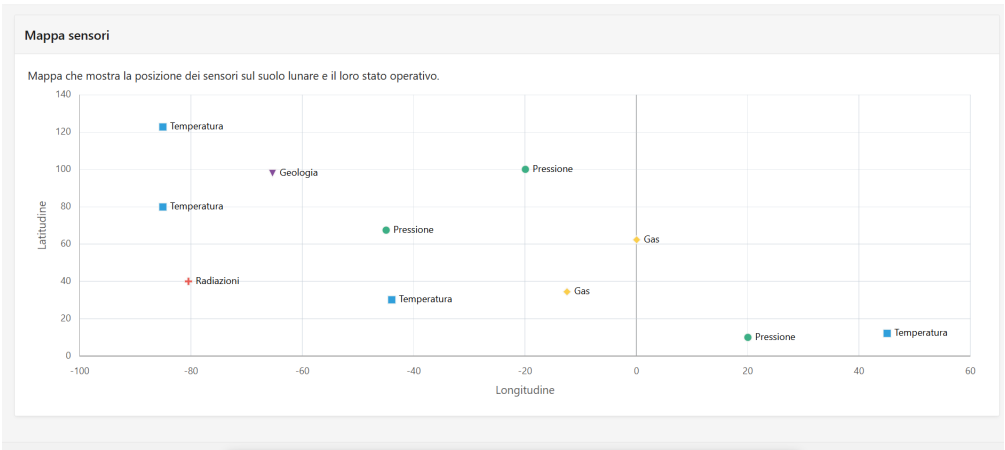


Figure 13: Schermata della sezione *Mappa*

Anomalie sensori							
<div> <div> <div>Q Search...</div> </div> <div> <div>Tipo Sensore</div> <div> <input type="checkbox"/> Temperatura (9) <input type="checkbox"/> Gas (4) <input type="checkbox"/> Pressione (4) <input type="checkbox"/> Geologia (1) <input type="checkbox"/> Radiazioni (1) </div> </div> <div> <div>Stato Sensore</div> <div> <input type="checkbox"/> Attivo (11) <input type="checkbox"/> Manutenzione (4) <input type="checkbox"/> Malfunzionante (2) <input type="checkbox"/> Standby (2) </div> </div> <div> <div>Livello Anomalia</div> <div> <input type="checkbox"/> Alta (6) <input type="checkbox"/> Critica (6) </div> </div> </div>							
<div> <div>Total Row Count 19</div> <div>Reset</div> </div>							
Id Sensore ↑	Tipo Sensore	Stato Sensore	Id Anomalia	Livello Anomalia	Causa Anomalia	Data	Orario
1	Temperatura	Attivo	1	Alta	Errore di calibrazione	01/12/2024	10:15:00
1	Temperatura	Attivo	18	Media	Errore di calibrazione	01/02/2024	08:30:00
1	Temperatura	Attivo	7	Media	Errore di calibrazione	12/12/2024	08:00:00
2	Pressione	Attivo	19	Critica	Malfunzionamento	15/01/2024	09:45:00
2	Pressione	Attivo	8	Alta	Problemi di connessione	18/11/2024	16:00:00
3	Gas	Manutenzione	2	Media	Sensore non risponde	15/11/2024	14:45:00
3	Gas	Manutenzione	9	Critica	Sensore non risponde	30/10/2024	12:30:00
4	Radiazioni	Malfunzionante	10	Alta	Errore di calibrazione	01/10/2024	09:15:00
5	Temperatura	Attivo	3	Critica	Guasto hardware	20/10/2024	08:30:00
5	Temperatura	Attivo	11	Media	Guasto hardware	22/09/2024	14:20:00
6	Pressione	Standby	12	Bassa	Problemi di connessione	15/08/2024	18:45:00
7	Temperatura	Attivo	4	Alta	Errore di calibrazione	10/09/2024	12:00:00
7	Temperatura	Attivo	13	Critica	Sensore non risponde	01/07/2024	11:00:00

Figure 14: Schermata della sezione *Anomalie*

Membri				
<div> <div> <div>Q Search...</div> </div> <div> <div>Nome</div> <div> <input type="checkbox"/> Luca Rossi (3) <input type="checkbox"/> Roberto Corti (3) <input type="checkbox"/> Alessio Mancini (2) <input type="checkbox"/> Elena Bianchi (2) <input type="checkbox"/> Giulia Ferrari (2) <input type="checkbox"/> Luigi Cirillo (2) <input type="checkbox"/> Paolo Frosti (2) Show All </div> </div> <div> <div>Obiettivo Missione</div> <div> <input type="checkbox"/> Controllo radiazioni (4) <input type="checkbox"/> Esplorazione Polo Sud Lunare (4) <input type="checkbox"/> Esplorazione del suolo marziano (3) <input type="checkbox"/> Installazione Base Avanzata (3) <input type="checkbox"/> Monitoraggio della pressione </div> </div> </div>				
<div> <div>Total Row Count 26</div> <div>Reset</div> </div>				
Id ↑	Nome	Obiettivo Missione	Id Missione	Stato Missione
1	Luca Rossi	Esplorazione Polo Sud Lunare	1	Annullata
1	Luca Rossi	Raccolta Campioni di Rocce	3	Completata
1	Luca Rossi	Controllo radiazioni	8	Annullata
2	Anna Esposito	Raccolta Campioni di Rocce	3	Completata
3	Marco Verdi	Installazione Base Avanzata	4	Annullata
4	Elisa Corsetti	Installazione Base Avanzata	4	Annullata
5	Giovanni Ferri	Installazione Base Avanzata	4	Annullata
6	Sara Russo	Esplorazione del suolo marziano	5	Completata
6	Sara Russo	Controllo radiazioni	8	Annullata
7	Gianluca Masotti	Esplorazione del suolo marziano	5	Completata
8	Greta Di Toro	Esplorazione del suolo marziano	5	Completata
9	Edoardo Bianchi	Monitoraggio della pressione atmosferica	6	In corso
10	Matteo Adaggio	Monitoraggio della pressione atmosferica	6	In corso

Figure 15: Schermata della sezione *Membri*

5.6 Gestione della sicurezza

Grazie alla sua profonda integrazione con il database Oracle, APEX garantisce sicurezza, affidabilità e un'elevata scalabilità, rendendolo ideale per progetti di qualsiasi dimensione, dalle piccole imprese alle grandi organizzazioni.

L'immagine (Figura 16) mostra la schermata di login dell'applicazione che implementa le best practice per garantire la sicurezza e la protezione dei dati. Questa schermata rappresenta il primo livello di sicurezza dell'applicazione, garantendo che solo utenti autorizzati possano accedere alle funzionalità e ai dati critici del sistema. L'accesso all'applicazione è protetto da un sistema di login che richiede

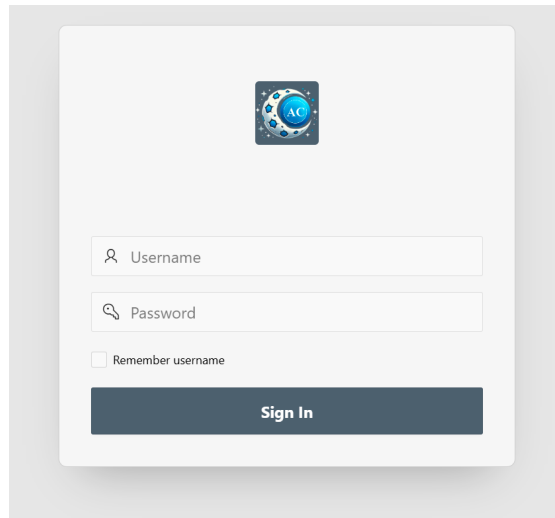


Figure 16: Schermata di login

l'inserimento di username e password.

Oracle APEX utilizza inoltre meccanismi di crittografia per garantire la protezione delle credenziali durante la trasmissione dei dati e il sistema è progettato per prevenire vulnerabilità come SQL injection e brute force.