# AppStar – Relazione

In questa relazione descriveremo prima la progettazione concettuale e logica del modello di base di dati da noi definito e poi l'applicazione desktop AppStar, per interrogare e aggiornare il database.

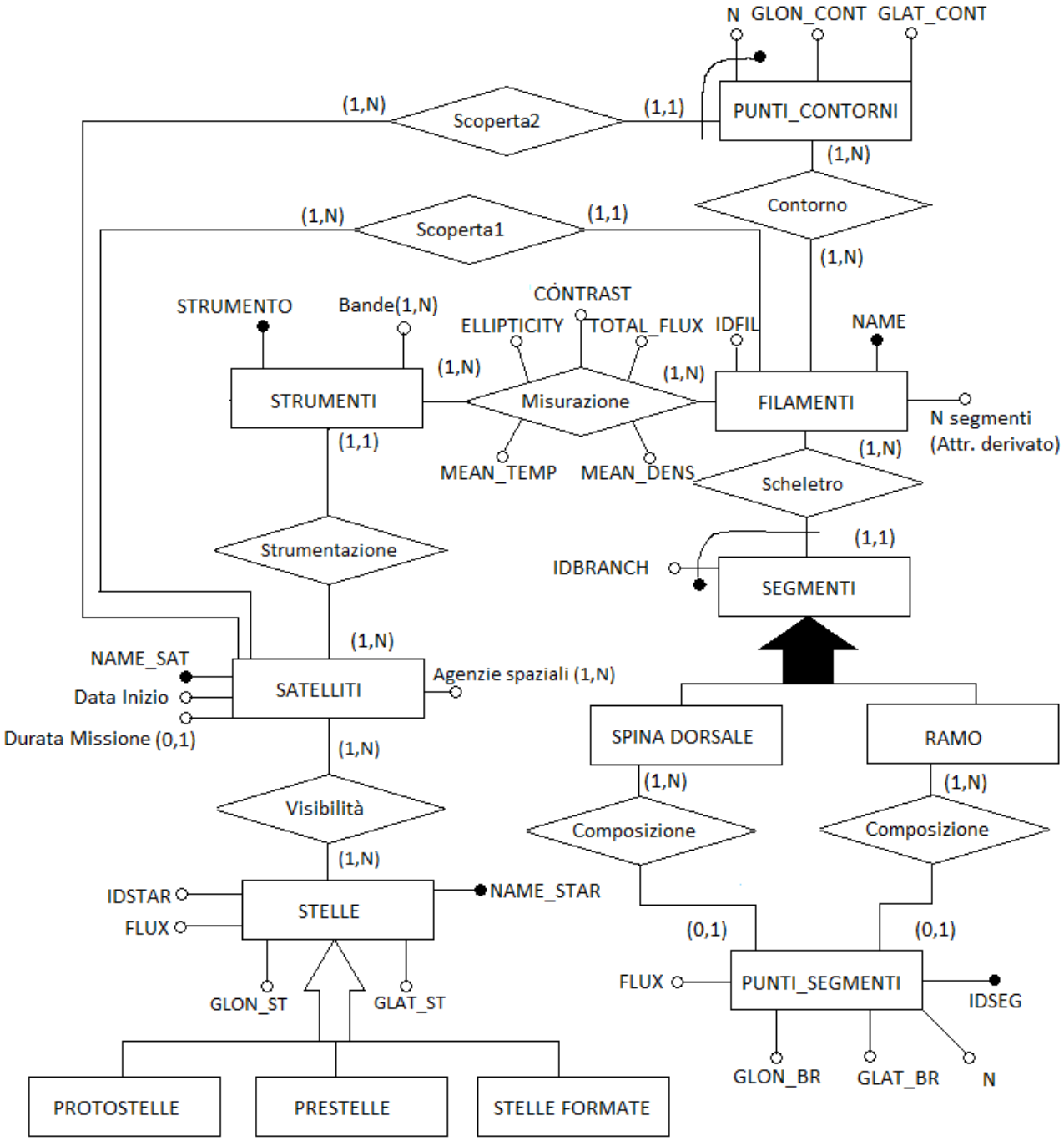
Per userid e password dell'applicazione vedere qui.

## Progettazione Concettuale – Glossario dei termini

Il seguente glossario contiene i termini ricorrenti nella descrizione del progetto. In verde sono evidenziate le entità, in azzurro le associazioni e in arancione gli attributi.

| Termine | Descrizione | Sinonimi | Collegamenti |
|---|---|---|---|
| Filamento | Struttura in cui i gas si raccolgono nello spazio, dalla forma filamentosa. Possiede uno scheletro, un asse principale e 0 o più rami secondari. Ha anche un contorno. | Struttura estesa | Scheletro, Contorno, Spina dorsale, Rami,Segmenti |
| Stelle | Oggetti luminosi puntiformi, con una posizione galattica. (a volte sono chiamate "punti" : OMONIMO) | Oggetti luminosi puntiformi, Sorgenti, Source | Posizione galattica |
| Contorno | Il contorno definisce il filamento ed è formato da **punti** distinti collegati. Ogni punto ha una posizione galattica. | Perimetro | Filamento, punti |
| Posizione galattica | Formata da Latitudine e longitudine galattica | Posizione, Posizione spaziale | Stelle, punti dei segmenti, spine dorsali e rami |
| Segmento | Insieme di punti(2 o più). I punti iniziali e finali si chiamano vertici. Sinonimo di spina dorsale e rami. | | Stelle, Scheletro, Spina dorsale |
| Satelliti | Permettono di vedere i filamenti. Dispongono di più strumenti. | | Strumenti, Bande |
| Bande | Lunghezza d'onda catturata dagli Strumenti. Si misurano in micrometri (indicati con u). | | Strumenti |
| Scheletro | Struttura interna al filamento che permette di identificarlo più facilmente. Formato da una spina dorsale e N rami. | | Spina dorsale, Rami, Filamento |
| Spina dorsale | Segmento principale che fa parte dello scheletro. E' formato da vari punti. | Asse principale, Spina centrale, segmento | Scheletro, punti, Segmenti |
| Strumenti | Propri dei satelliti, catturano le bande, e permettono di individuare oggetti caldi o freddi (stelle, filamenti e altri corpi celesti) | | Satelliti |
| Rami | Segmento secondario che fa parte dello scheletro. E' formato da vari punti. | Segmenti secondari, rami secondari, branch | Scheletro, Punti, Segmenti |
| Punti | I punti definiscono i contorni dei filamenti, oltre che i segmenti degli scheletri dei filamenti (ovvero le spine dorsali e i rami) | | Contorni,rami, Segmenti, Spine dorsali, Posizione gal. |

## Progettazione Concettuale – Schema E-R




**Descrizione – tra parentesi quadre si evidenziano le scelte nello schema E-R:**

I FILAMENTI, scoperti da un SATELLITE [1 a molti], hanno uno Scheletro formato da vari SEGMENTI [1 a molti]. Ogni SEGMENTO può essere una SPINA DORSALE o un RAMO[generalizzazione totale: non ci sono altre classificazioni], i quali sono a loro volta composti da vari punti (PUNTI_SEGMENTI) [1 a molti]. Su ogni FILAMENTO vengono eseguite delle Misurazioni di dati scientifici [molti a molti] da parte di uno STRUMENTO, che fa parte dell'equipaggiamento di un SATELLITE[1 a molti]; è possibile che ogni FILAMENTO sia misurato 2 volte dallo stesso STRUMENTO o che uno STRUMENTO misuri molti FILAMENTI diversi. Inoltre ogni FILAMENTO possiede un unico Contorno, formato da vari PUNTI_CONTORNI [molti a molti], calcolati da un unico SATELLITE; è possibile che i PUNTI_CONTORNI siano in comune tra 2 o piu' Contorni.

Le STELLE sono anche esse Visibili da un SATELLITE, ma è possibile in generale che la stessa STELLA sia visibile da SATELLITI diversi o, viceversa, che lo stesso SATELLITE veda diverse stelle [molti a molti]. Le

STELLE si possono suddividere in PROTOSTELLE, PRESTELLE e STELLE FORMATE, ma sono possibili anche altre classificazioni [perciò utilizziamo la generalizzazione parziale].
I SATELLITI possiedono uno o più STRUMENTI [attributo multivalore] e sono spediti in missione da 1 o più AGENZIE [1 a molti], mentre gli STRUMENTI possono misurare una o più BANDE [attributo multivalore]. Da notare che i SATELLITI possono ancora essere in missione [attributo opzionale: "Durata"].
I FILAMENTI hanno un id, chiamato "IDFIL", che però non li identifica univocamente, perciò nello schema del database abbiamo scelto come chiave primaria "NAME", che invece è univoca per ogni FILAMENTO. Nell'applicazione è spesso però utilizzata un'altra chiave: ("IDFIL", "SATELLITE") per rendere più semplice l'utilizzo della stessa.
Per i SEGMENTI vale un discorso simile, in quanto vi sono ripetizioni nell'attributo "IDBRANCH", perciò abbiamo preferito identificarlo in modo debole tramite("IDBRANCH", "NOME_FILAMENTO").
Abbiamo inoltre scelto di separare i PUNTI_SEGMENTI da PUNTI_CONTORNI, per i seguenti motivi:

- per evitare valori nulli su "FLUX"
- i punti del contorno non sono numerati da un numero progressivo "N".
- entrambi prevedono delle ripetizioni nelle longitudini e latitudini che quindi non li identificano univocamente, abbiamo usato quindi dei nuovi numeri progressivi ("IDSEG" su PUNTI_SEGMENTI e "NPCONT" in Contorni). Per maggiori dettagli consultare Descrizione Class Diagram AppStar.

Per rimanere coerenti, abbiamo scelto come chiave primaria "NAME_STAR" al posto di "IDSTAR", perché altri satelliti potrebbero usare gli stessi ID per identificare stelle diverse.

## Progettazione Concettuale – Dizionario dei dati

ENTITA'

| Nome Entità | Descrizione | Attributi | Identificatore |
|---|---|---|---|
| Filamenti | Struttura in cui i gas si raccolgono nello spazio, dalla forma filamentosa. Possiede uno scheletro, con almeno un asse principale e 0 o più rami secondari. Ha anche un contorno. | IDFIL, Numero Segmenti | NAME |
| Segmenti | Insieme di punti interni al filamento che permettono di definirlo più facilmente. L'insieme di tutti i segmenti di un filamento rappresenta lo scheletro. | | IDBRANCH, (NAME_FIL chiave esterna) |
| Spina Dorsale | Segmento che fa parte dello scheletro. E' formata da vari punti. | | IDBRANCH, (NAME_FIL chiave esterna) |
| Rami | Segmento secondario che fa parte dello scheletro. E' formato da vari punti. | | IDBRANCH,(NAME_FIL chiave esterna) |
| Punti_Segmenti | Rappresentano i punti delle mappe spaziali e definiscono i segmenti (ovvero le spine dorsali e i rami). | GLON_BR, GLAT_BR, FLUX ,N | IDSEG |
| Punti_contorni | Rappresentano i punti delle mappe spaziali e definiscono i contorni dei filamenti. | GLAT_CONT, GLON_CONT | N, (SATELLITE chiave esterna) |
| Stelle | Oggetti luminosi puntiformi, con una posizione galattica. | IDSTAR, GLAT_ST,GLON_ST, FLUX | NAME_STAR |

| | | | |
|---|---|---|---|
| Protostelle, Prestelle, Stelle formate | Stelle particolari rilevate a una certa banda | IDSTAR, GLAT_ST,GLON_ST, FLUX | NAME_STAR |
| Strumenti | Catturano le bande, che permettono di individuare oggetti caldi o freddi (stelle, filamenti e altri corpi celesti) | Bande(1,N) | STRUMENTO |
| Satelliti | Vengono lanciati da agenzie spaziali per studiare i filamenti. Dispongono di più strumenti. | Data inizio, Durata missione(0,1), Agenzie spaziali(1,N) | NAME_SAT |

RELAZIONI / ASSOCIAZIONI

| **Nome relazione** | **Descrizione** | **Entità coinvolte** | **Attributi associazione** |
|---|---|---|---|
| Contorno | Il contorno definisce il filamento ed è formato da punti distinti collegati. Associa ai punti, i filamenti che circondano. | Filamenti(1,N), Punti_contorni(1,N) | |
| Scheletro | Struttura interna al filamento che permette di identificarlo più facilmente. Associa un filamento ai segmenti di cui è composto. | Filamenti(1,N), Segmenti(1,1) | |
| Composizione | Associa a un segmento (ramo o spina dorsale) i punti di cui è composto | Spine Dorsali o Rami (1,N) Punti_Segmenti (0,1) | |
| Misurazione | Associa agli strumenti i filamenti che misurano. | Satelliti (1,N), Filamenti (1,N) | TOTAL_FLUX, MEAN_DENS, MEAN_TEMP, ELLIPTICITY, CONTRAST |
| Visibilità | Associa a uno strumento le stelle che rileva. | Strumenti (1,N), Stelle(1,N) | |
| Strumentazione | Associa a un satellite gli strumenti che possiede | Satelliti(1,N) Strumenti(1,1) | |
| Scoperta1 | Associa un filamento al satellite che lo scopre. | Satelliti(1,N), Filamenti (1,1) | |
| Scoperta2 | Associa un punto di un segmento al nome del satellite che lo rileva. | Satelliti(1,N), Punti_Segmenti (1,1) | |

## Progettazione Concettuale – Business Rules

Queste sono le regole di progetto (regole aziendali):

- Due segmenti appartenenti a filamenti diversi non devono avere punti in comune
- I punti di un segmento non devono sovrapporsi ai punti del contorno dello stesso filamento
- La distanza tra due punti p1 e p2(aventi latitudine e longitudine galattica) si ottiene calcolando la distanza euclidea $d(p1, p2) = \sqrt{(GLat1 - GLat2)^2 + (GLon1 - GLon2)^2}$
- Il valore contrasto data la percentuale di brillanza o di luminosità si ottiene da questa formula: 1 + %Brillanza/100

## Progettazione Logica – Analisi delle ridondanze – Numero segmenti

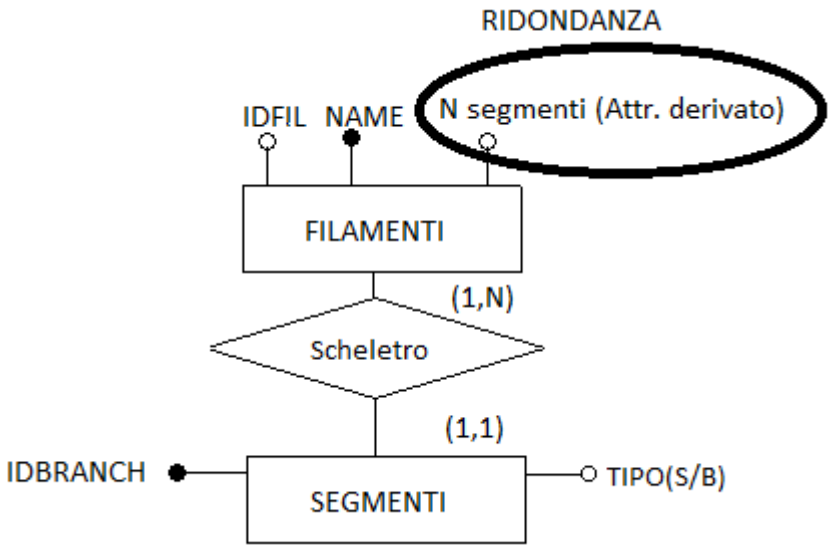
L'operazione 5 e l'operazione 7 richiedono il **numero dei segmenti**. Se non fosse noto a priori, ogni volta che viene utilizzata la funzione del requisito 5, bisognerebbe contare tutti i segmenti di un filamento (Ipotizziamo che in media ogni filamento abbia 5 segmenti).

Tavola dei volumi

| Concetto | Tipo | Volume atteso(Alcuni calcolati, altri ipotizzati) |
|---|---|---|
| Filamenti | E | 10 000 |
| Strumenti | E | 20 |
| Satelliti | E | 10 |
| Stelle | E | 60 000 |
| Punti(segmenti) | E | 2.5 milioni |
| Punti(contorni) | E | 3.0 milioni |
| Segmenti | E | 50 000 |
| Bande | E | 20 |
| Agenzie | E | 5 |
| Strumentazione | R | 20 |
| Misurabilità | R | 20 |
| Visibilità | R | 60 000 |
| Misurazione | R | 15 000 |
| Scheletro | R | 2.5 milioni |
| Contorno | R | 3.5 milioni |

Tavola delle operazioni

| Requisito funzionale | TipoOp | Frequenza attesa(IPOTESI) |
|---|---|---|
| 1 | I | 100 al giorno |
| 3 | I | 5 al giorno |
| 4 | I | 5 al giorno |
| 5 | I | 500 al giorno (ogni operazione) |
| 6 | I | 400 al giorno (ogni operazione) |
| 7 | I | 400 al giorno |
| 8 | I | 100 al giorno |
| 9 | I | 400 al giorno |
| 10 | I | 50 al giorno |
| 11 | I | 20 al giorno |
| 12 | I | 20 al giorno |



Senza ridondanza di N segmenti

- Non utilizziamo spazio aggiuntivo
- Op.2: bisogna contare in media 5 segmenti, quindi accedere 1 volta a filamenti, 5 volte a scheletro e 5 volte a segmenti. Quindi 11 accessi in lettura, per un totale di 10.500 accessi al giorno (trascurabili)
- Op. 7: Bisogna contare, per ogni filamento (10000 accessi), il suo numero di segmenti (50000 accessi, sia a Scheletro, sia a Segmenti) e controllare se i suoi valori sono compresi nel range specificato (0 accessi). Totale: 110.000 * 400 = 44 milioni di accessi al giorno

Con ridondanza di N segmenti:

- Lo spazio aggiuntivo per salvare un integer è 4 byte (IPOTESI), perciò per salvare il numero per tutti i filamenti abbiamo bisogno di 4 B*10 000 = 40 000 B = 40 KB
- Op. 2 bisogna fare un solo accesso in memoria secondaria, per leggere l'attributo. Totale: 500 accessi al giorno (dato trascurabile)
- Op. 7 bisogna fare 10.000 accessi per leggere N segmenti (1 per ogni filamento) . Totale: 10.000*400 = 4 milioni di accessi al giorno

CONVIENE MANTENERE LA RIDONDANZA

### Nota sulle stelle nei filamenti

I dati a noi disponibili per le stelle sono solo quelli di Heschel, quindi cercare le stelle in un filamento che non sia stato scoperto da Herschel non ha senso. E' stata comunque lasciata la scelta del satellite, nel caso siano disponibili dati di Spitzer sulle stelle. Questo discorso è valido per i requisiti 9, 10 e 12.

Inoltre, controllare che le stelle siano all'interno dei filamenti è molto costoso sia in termini di tempo che di spazio. Una stella può anche stare in più di un filamento (per come è fatta la formula del requisito 9), perciò un semplice attributo ridondante non basta. E' necessario costruire una tabella stelle_in_filamento_tmp con due colonne: STELLA e IN_FILAMENTO, con un occupazione di circa 50 byte per riga.

Poiché le stelle sono circa 60000 e i filamenti (di Herschel) sono 10000, se ipotizziamo che ogni filamento contenga in media 15000 stelle, la tabella dovrebbe avere circa 15000*10000 = 150 milioni di righe.

Tralasciando la questione del tempo necessario per popolare la tabella, elevatissimo, lo spazio occupato da essa dovrebbe essere di circa 50 byte * 150 milioni di righe = 6.98 GB, se contenesse ogni filamento e ogni stella.

Abbiamo quindi deciso di costruire ugualmente la tabella stelle_in_filamento_tmp ed escludere un buon numero di stelle dal conteggio: calcoliamo solo le stelle con "IDSTAR" dispari e non divisibile per 3, per lasciare un ragionevole numero di stelle e evitare che alcuni filamenti non ne abbiano nessuna all'interno. In questo modo lo spazio occupato e il tempo di esecuzione delle query diminuiscono notevolmente, anche per i filamenti con molti punti del contorno.

Quando si esegue la query del requisito 9 o 12, se necessario, prima riempiamo la tabella temporanea con i filamenti necessari, poi vengono eseguiti i rispettivi calcoli.

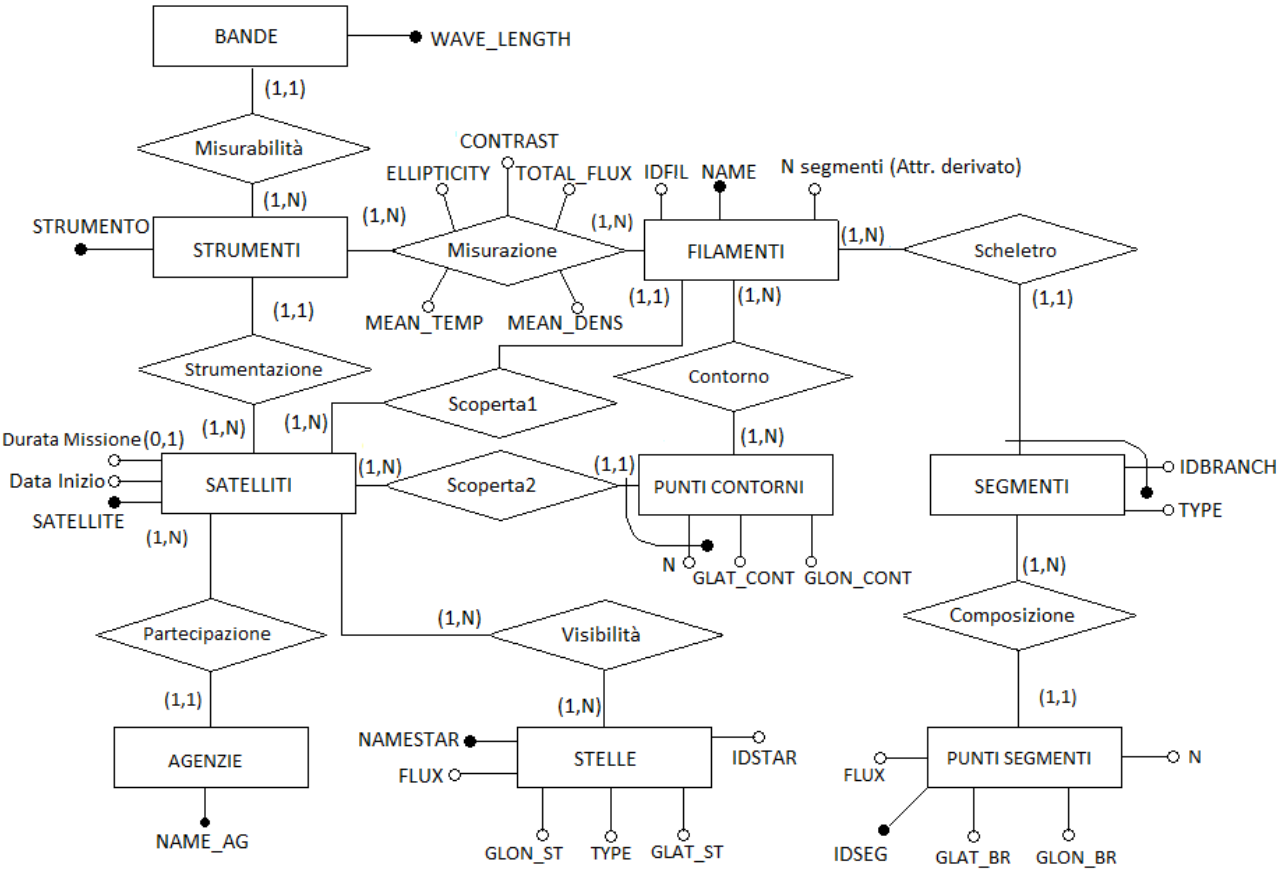
## Progettazione Logica – Ristrutturazione schema E-R

Per tradurre lo schema E-R nello schema logico, abbiamo scelto di effettuare le seguenti ristrutturazioni:

- Bande misurabili è un attributo multivalore quindi va trasformato in un entità identificata da un attributo "WAVE_LENGTH"
- Stesso discorso per l'attributo Agenzie di Satelliti, percio' abbiamo reificato l'attributo in una entità che avrà chiave "NAME_AG"

- La generalizzazione su stelle: abbiamo eliminato le 3 entità figlie e aggiunto l'attributo TYPE a STELLE. Infatti nei requisiti non sono richieste ulteriori informazioni sulle stelle di un particolare tipo.
- La generalizzazione su Segmenti: abbiamo eliminato Rami e spine dorsali, per poi collegare Composizione a segmenti. Infine, abbiamo aggiunto l'attributo tipo a Segmenti, che può essere solo 'B' o 'S'.
- Abbiamo aggiunto l'attributo "NPCONT" (integer autoincrement) a Punti_contorni per rendere univoche le righe (insieme a SATELLITE). Questo ci sarà utile sia in fase di importazione, sia per il requisito 9, che richiede una ripetizione della posizione galattica tra primo e ultimo punto di ogni segmento
- In Punti_Segmenti, "IDBRANCH" non è univoco, perciò abbiamo aggiunto la chiave "IDSEG", un semplice Intero che si autoincrementa, per risolvere il problema

## Progettazione Logica – Schema E-R Ristrutturato



## Progettazione Logica – Schema Logico e della base di dati

FILAMENTI(IDFIL, NAME, NUM_SEG, SATELLITE)
SEGMENTI(IDBRANCH, TYPE, NAME_FIL)
PUNTI_SEGMENTI(SEGMENTO, GLON_BR,GLAT_BR, N, FLUX, IDSEG, NAME_FIL)
PUNTI_CONTORNI(GLON_CONT,GLAT_CONT, N, SATELLITE)
STRUMENTI(STRUMENTO,SAT)
STELLE(IDSTAR, NAME_STAR, GLON_ST, GLAT_ST, FLUX, TYPE)
BANDE(WAVE_LENGTH, STRUMENTO)
SATELLITI(NAME_SAT, DATA_INIZIO, DURATA*) *opzionale
AGENZIE(NAME_AG, SAT)
CONTORNI(NAME_FIL, NPCONT, SATELLITE)
Misurazione(STRUMENTO,FILAMENTO, MEAN_DENS,MEAN_TEMP, ELLIPTICITY, CONTRAST, TOTAL_FLUX)
Visibilità(SATELLITE,STELLA)
Utenti(Nome, Cognome, Userid, Password, e-mail, isAmministratore)

NOTA1: la Tabella Utenti è stata aggiunta per salvare i dati degli utenti, richiesti nel requisito 1 e 2.
NOTA2: le tabelle che terminano per _imp servono per l'importazione e sono svuotate al suo termine.

## Descrizione Applicazione AppStar

Per accedere all'applicazione sono necessari id e password.

Esempio di utente amministratore: (id, password) = (lorzar, lorzar)
Esempio di utente semplice: (id, password) = (red_jack, redjack)
Se si vuole conoscere le altre credenziali, controllare la tabella Utenti, dal restore del dump allegato.

**Requisito 1: Login e Menu Home**
- Schermata Login: L'app permette l'accesso se i dati inseriti per user e password corrispondono alle credenziali di un utente registrato, salvate nella base di dati nella tabella Utenti.
- Menu Home: scelta dell'operazione da effettuare. Funzioni amministratore sono bloccate per gli utenti semplici. Cliccare il tasto Avanti per andare alla schermata della funzione scelta.

**Requisiti 2-4: Funzioni disponibili solo agli Utenti Amministratori**
- Registrazione di un nuovo utente: Permette di aggiungere l'utente al database
- Importazione di file csv: importa i file in una tra le tabelle "contorni_imp" , "filamenti_imp", "scheletri_imp", "stelle_imp", per poi smistarli nel database.
-Inserire dati dei satelliti e delle agenzie
-Inserire dati degli strumenti e delle bande

**Requisiti 5-12: Funzioni dell'applicazione, disponibili a tutti gli utenti registrati**
- Calcolo Centroide, Estensione e numero segmenti di un filamento
- Ricerca filamenti per contrasto e ellitticità
- Ricerca filamenti per numero di segmenti
- Ricerca di filamenti in regioni quadrate o circolari di mappa
- Ricerca delle stelle interne a un filamento
- Ricerca di stelle in regioni rettangolari, interne o esterne ai filamenti
- Calcolo della distanza minima degli estremi di un segmento di un filamento dal contorno
- Calcolo della distanza minima dalla spina dorsale per stelle interne a un filamento

## Descrizione Class Diagram e Test Report [FARE]

I class diagram allegati alla relazione sono stati sviluppati con il software gratuito StarUML e sono stati salvati tutti nel file ClassDiagram.mdj e anche nel file Class Diagram.pdf. Abbiamo diviso i class diagram per requisito funzionale, in modo da aumentarne la leggibilità. Per semplicità ogni class diagram, ad eccezione del primo, non contengono le classi LoginGUI e LoginController.

Tutti i test effettuati sono raggruppati per suite, in modo da essere eseguiti nell'ordine corretto. Abbiamo anche implementato la suite AllRF_TestSuite per eseguire tutti i test sequenzialmente.

### *Requisito 1 e 2[AGGIUNGERE LoginPopUp]*

**Descrizione Class Diagram:**

Il metodo start() del LoginController avvia l'applicazione e istanzia la schermata LoginGUI . Una volta inseiriti i dati, Il Login Controller verifica che le credenziali inserite siano presenti nella tabella utenti e viene istanziato il singleton UtenteConnesso. Tale istanza è di tipo UtenteRegistrato(che può specializzarsi in UtenteAmministratore). Il controllo del login viene effettuato dalla classe UtenteDAO che si connette al database Postgres(tramite il metodo connettiti() della classe Connessione) e chiama il metodo controlloAccount(). Se le credenziali sono corrette viene istanziata l'HomeGUI dall'HomeController, altrimenti non permette l'accesso.

**Test effettuati:**

1. Login dell'utente registrato lorzar, che è anche un amministratore. Ci aspettiamo che riesca ad accedere e che i suoi dati siano correttamente visualizzati.
   Valore atteso: [Lorenzo, Zara, lorzar, lorzar, lorenzo.zara96@gmail.com, amministratore]
   Valore ottenuto: [Lorenzo, Zara, lorzar, lorzar, lorenzo.zara96@gmail.com, amministratore]
2. Login dell'utente registrato red_jack. Ci aspettiamo che riesca ad accedere e che i suoi dati siano correttamente visualizzati.
   Valore atteso: [Giacomo, Rossi, red_jack, redjack, giacomo.redjack@gmail.com, notAmministratore]
   Valore ottenuto: [Giacomo, Rossi, red_jack, redjack, giacomo.redjack@gmail.com, notAmministratore]
3. Login di un utente non registrato. Ci aspettiamo che NON riesca ad accedere e che i suoi dati NON siano disponibili.
   Valore atteso: [] (array vuoto)
   Valore ottenuto: []

### *Requisito 3*

**Descrizione Class Diagram:**

Una volta effettuato l'accesso come utente amministratore, viene istanziato un oggetto da UtenteAmministratore che ha come metodi quelli relativi alle operazioni del requisito funzionale 3 che permettono di: registrare un nuovo utente, inserire nuovi dati di un satellite ed inserire un nuovo strumento. A seconda del radio button selezionato nella HomeGUI (relativo ad un'operazione permessa esclusivamente agli amministratori) verrà istanziato il corrispettivo controller che istanzierà la GUI dell'operazione desiderata. All'interno di tale GUI (InserisciFileDatiSatelliteGUI, InserisciDatiStrumentiGUI, ImportaFileSatelliteGUI, RegistraUtenteGUI) saranno permesse le varie operazioni, a patto che (grazie ad un ulteriore controllo sull'entità dell'istanza del Singleton) l'utente sia effettivamente un amministratore. Le funzioni vere e proprie sono eseguite da FileImportazioneDao e da UtenteDao che comunicano col database attraverso la classe Connessione. Nel caso dell'importazione dei CSV abbiamo scelto di salvarli in delle tabelle temporanee (riconoscibili dal suffisso "_imp"). Ogni volta che viene importato un file, la tabella corrispettiva viene troncata e riempita coi nuovi dati. Subito dopo i dati vengono distribuiti in tutte le tabelle che presentano nella loro query di inserimento la relazione scelta all'interno del JOIN.

**Test effettuati:**

Prima della test Suite (BeforeClass) viene cancellato dal database l'amministratore alberto, che sarà creato nel punto 1.

1. Creazione di un nuovo amministratore (albertone) .
   Valore atteso: InserisciAccount() ritorna un boolean: se vero, il nuovo utente viene creato con successo. Attendiamo true.
   Valore ottenuto: true
2. Creazione di un nuovo utente con UserID e Password di 2 caratteri
   Valore atteso: ci attendiamo false, perché user id e password devono avere almento 6 caratteri
   Valore ottenuto: false
3. Inserimento di nuovi dati di uno strumento da parte dell'amministratore albertone:
   nome: ANDREA, banda: 9.99, satellite: Herschel.
   Valore atteso: I dati non sono presenti nel database, quindi ci aspettiamo che inserisciNuoviDatiStrumento() restituisca true, ovvero che l'inserimento abbia successo.
   Valore ottenuto: true
4. Inserimento di uno strumento di un satellite non ancora inserito:
   nome: ANDREA, banda 3.0, satellite: nuovoSatellite
   Valore atteso: ci aspettiamo una violazione di foreign key, quindi non verrà inserito nulla e attendiamo false.
   Valore ottenuto: false.
5. Inserimento di nuovi dati di un satellite da parte dell'amministratore albertone:
   Nome: AGILE, agenzia: ISA, partenza 27/04/2007, durata: ancora in missione.
   Valore atteso: Ci aspettiamo che l'inserimento abbia successo e che inserisciNuoviDatiSatellite() restituisca true.
   Valore ottenuto: true

Al termine della testSuite (AfterClass), tutti i dati inseriti vengono subito eliminati, in modo da poter ripetere il test senza ottenere errori.

## *Requisito 4*

**Descrizione Class Diagram:**

Vedi descrizione del class diagram del requisito 3.

**Test effettuati:**

Poiché anche l'importazione è un operazione possibile solo agli amministratori,prima di eseguire i seguenti test (BeforeClass) viene eliminato l'amministratore albertone e vengono ripetuti i primi 2 test del requisito precedente.

1. Importiamo il file filamenti_Herschel.
   Valore atteso: la query di importazione è riusabile, grazie alla clausola ON CONFLICT, quindi che i dati siano presenti o no, ci aspettiamo che vengano inseriti e attendiamo true.
   Valore ottenuto: true
2. Importiamo il file stelle_Herschel.
   Valore atteso: la query di importazione è riusabile, grazie alla clausola ON CONFLICT, quindi che i dati siano presenti o no, ci aspettiamo che vengano inseriti e attendiamo true.
   Valore ottenuto: true
3. Proviamo a importare il file Herschel_nulla.txt, allegato al codice sorgente.

Valore atteso: la nostra applicazione non permette di importare un file che non contenga il nome del satellite o che non sia in formato csv. Visto che è in formato txt ( e non contiene dati), il file non dovrebbe essere importato e ci aspettiamo false.
Valore ottenuto: false

### *Requisito 5*

**Descrizione Class Diagram:**

Dal menu della HomeGUI viene istanziata il controller InfoFilamentiController che a sua volta istanzia l'interfaccia grafica InfoFilamentiGUI. Una volta inseriti i dati e avviata la richiesta di calcolo viene eseguita la funzione(calcolaCentroide(), calcola Estensione(), calcolaNumSeg()) propria dell'istanza della classe UtenteRegistrato ottenuta tramite getInstance() del Singleton UtenteConnesso. Le interrogazioni sul database vengono eseguite all'interno del FileDao dalle omonime funzioni.

**Test effettuati:**

L'applicazione permette di inserire il nome del filamento, o in alternativa l'idfil. Se si inseriscono entrambi, se l'id è sbagliato ha precedenza il nome, mentre se il nome è inventato e l'id è giusto, viene calcolato il filamento con l'id specificato.

1. Calcolo del centroide del filamento 409 con successo.
   Valore atteso: Attendiamo un valore non nullo in quanto il filamento è già nel database, con tutti i suoi dati
   Valore ottenuto: Not Null
2. Calcolo del centroide fallimentare di un filamento inesistente (0)
   Valore atteso: Attendiamo un valore nullo in quanto il filamento è inesistente. In questo caso la funzione calcolaCentroide() ritorna "NON TROVATO"
   Valore ottenuto: "NON TROVATO"
3. Calcolo dell'estensione con successo del filamento HiGALFil015.9322-1.0422 (e idfil 0)
   Valore atteso: Attendiamo un valore non nullo in quanto il filamento esiste nel database.
   Valore ottenuto: Not Null
4. Calcolo dell'estensione fallimentare di un filamento dal nome inesistente (filamentoACaso)
   Valore atteso: Attendiamo un valore nullo in quanto il filamento è inesistente. In questo caso la funzione calcolaEstensione() ritorna "NON TROVATO"
   Valore ottenuto: "NON TROVATO"
5. Calcolo del numero di segmenti del filamento 409. Il numero di segmenti viene calcolato in fase di importazione, quindi si tratta solamente di interrogare la tabella dato l'id e il satellite.
   Valore atteso: Ci aspettiamo il numero di segmenti del filamento 409, che è esattamente 21
   Valore ottenuto: 21
6. Calcolo del numero di segmenti fallimentare (filamento 0 di Herschel)
   Valore atteso: Attendiamo un valore nullo in quanto il filamento è inesistente. In questo caso la funzione calcolaNumSeg() ritorna -1
   Valore ottenuto: -1

### *Requisito 6*

**Descrizione Class Diagram:**

Dal menu della HomeGUI viene istanziata il controller RicercaFilamentoLumController che a sua volta istanzia l'interfaccia grafica RicercaFilamentoLumGUI. Una volta inseriti i dati e avviata la richiesta di calcolo

viene eseguita la funzione cercaFilamenti() propria dell'istanza della classe UtenteRegistrato ottenuta tramite getInstance() del Singleton UtenteConnesso. Le interrogazioni sul database vengono eseguite all'interno del FileDao dalla funzione cercaFilamenti(), che utilizza l'entità Filamento per creare dei record che vanno a riempire la tableView dell'interfaccia grafica.

**Test effettuati:**

Da notare che, per riutilizzare il codice, nei test seguenti sono stati usati valori nulli per i parametri riguardanti le tableView, in quanto in fase di test non sono necessarie.

1. Ricerca dei filamenti con ellitticità tra 2.3 e 9.9 e con luminosità maggiore del 40%
   Valore atteso: Ci aspettiamo un numero di filamenti trovati diverso da 0
   Valore ottenuto: 3687
2. Ricerca di filamenti data una percentuale negativa (-5%), che non deve essere accettata dal programma.
   Valore atteso: Ci aspettiamo un numero di filamenti trovati pari a 0 cercaFilamenti() ha come valore di ritorno il numero di segmenti.
   Valore ottenuto: 0
3. Ricerca di filamenti data una ellitticità compresa tra 1.0 e 11.5, che sono valori fuori range.
   Valore atteso: Ci aspettiamo un numero di filamenti trovati pari a 0.
   Valore ottenuto: 0

## *Requisito 7*

**Descrizione Class Diagram:**

Dal menu della HomeGUI viene istanziata il controller RicercaFilamentoSegController che a sua volta istanzia l'interfaccia grafica RicercaFilamentoSegGUI. Una volta inseriti i dati e avviata la richiesta di calcolo viene eseguita la funzione cercaFilamentiSeg() propria dell'istanza della classe UtenteRegistrato ottenuta tramite getInstance() del Singleton UtenteConnesso. Le interrogazioni sul database vengono eseguite all'interno del FileDao dalla funzione cercaFilamentiSeg(), che utilizza l'entità Filamento per creare dei record che vanno a riempire la tableView dell'interfaccia grafica.

**Test effettuati:**

1. Ricerca dei filamenti aventi un numero di segmenti compreso tra 20 e 22.
   Valore atteso: La dimensione minima del range è 3 ma in questo caso è minore. Perciò ci aspettiamo che non vengano mostrati filamenti e che cercaFilamentiSeg() ritorni 0.
   Valore ottenuto: 0
2. Ricerca dei filamenti aventi un numero di segmenti tra 5 e 30.
   Valore atteso: Il range è accettato dal programma quindi ci aspettiamo che cercaFIlamentiSeg ritorni un numero diverso da 0.
   Valore ottenuto: 8300.

## *Requisito 8*

**Descrizione Class Diagram:**

Dal menu della HomeGUI viene istanziata il controller RicercaFilamentoRegioneController che a sua volta istanzia l'interfaccia grafica RicercaFilamentoRegioneGUI. Una volta inseriti i dati e avviata la richiesta di

calcolo viene eseguita la funzione cercaFilamentiInRegione() propria dell'istanza della classe UtenteRegistrato ottenuta tramite getInstance() del Singleton UtenteConnesso. Le interrogazioni sul database vengono eseguite all'interno del FileDao dalla funzione cercaInRegione(), che utilizza l'entità Filamento per creare dei record che vanno a riempire la tableView dell'interfaccia grafica.

**Test effettuati:**

1. Ricerca di filamenti in un quadrato di lato 20 e centroide in (0,0).
   Valore atteso: Questo quadrato contiene all'interno dei filamenti, inoltre cercaFilamentiInRegione() restituisce 1 se ha successo, 0 se fallisce. Ci attendiamo il valore 1.
   Valore ottenuto: 1
2. Ricerca di filamenti in un quadrato con raggio negativo
   Valore atteso: Non esistono lunghezze negative, quindi ci aspettiamo il valore 0
   Valore ottenuto: 0
3. Ricerca di filamenti in un cerchio di raggio 50.0 e centro in (0,0)
   Valore atteso: Questo cerchio contiene all'interno dei filamenti, inoltre cercaFilamentiInRegione() restituisce 1 se ha successo, 0 se fallisce. Ci attendiamo il valore 1.
   Valore ottenuto: 1
4. Ricerca di filamenti in un cerchio fallimentare
   Ricerca di filamenti in un cerchio con raggio negativo
   Valore atteso: Non esistono lunghezze negative, quindi ci aspettiamo il valore 0
   Valore ottenuto: 0

### *Requisito 9*

**Descrizione Class Diagram:**

Dal menu della HomeGUI viene istanziata il controller RicercaStelleInFilamentoController che a sua volta istanzia l'interfaccia grafica RicercaStelleInFilamentoGUI. Una volta inseriti i dati e avviata la richiesta di calcolo viene eseguita la funzione cercaInFilamento() propria dell'istanza della classe UtenteRegistrato ottenuta tramite getInstance() del Singleton UtenteConnesso. Le interrogazioni sul database vengono eseguite all'interno del FileDao dalla funzione cercaInFilamento(), che utilizza l'entità Stella per creare dei record che vanno a riempire la tableView dell'interfaccia grafica.

**Test effettuati:**

Da notare che non sono state contate tutte le stelle ma solo quelle con idstar dispari e non divisibile per 3, per rendere piu' rapida la ricerca. Inoltre poiché ogni nuovo filamento di cui vogliamo sapere le stelle interne viene aggiunto alla tabella temporanea stelle_in_filamenti_tmp insieme al le sue stelle, usando questo accorgimento limitiamo anche lo spazio utilizzato.

1. Controllo delle stelle interne al filamento 409
   Valore atteso: Sappiamo che il filamento 409 di Herscel contiene al suo interno 5349 stelle.
   Valore ottenuto: 5349
2. Ricerca delle stelle interne al filamento 45. Vogliamo controllare che le stelle interne al filamento 45 siano diverse da 0.

Valore atteso: un valore diverso da 0. (La somma tra le stelle trovate di ogni tipo)
Valore ottenuto: 4378

3. Ricerca delle stelle interne al filamenti 0 (inesistente)
   Valore atteso: Quel filamento non esiste perciò non dovrebbero essere state trovate stelle.
   Valore ottenuto: 0

### *Requisito 10*

**Descrizione Class Diagram:**

Dal menu della HomeGUI viene istanziata il controller RicercaStelleInRegioneController che a sua volta istanzia l'interfaccia grafica RicercaStelleInRegioneGUI. Una volta inseriti i dati e avviata la richiesta di calcolo viene eseguita la funzione cercaInRegione() propria dell'istanza della classe UtenteRegistrato ottenuta tramite getInstance() del Singleton UtenteConnesso. Le interrogazioni sul database vengono eseguite all'interno del FileDao dalla funzione cercaInRegione(), che utilizza l'entità Stella per creare dei record che vanno a riempire la tableView dell'interfaccia grafica.

**Test effettuati:**

1. Controllo del numero di stelle interne a un rettangolo di base 12 e altezza 4, centrato in (0,0)
   Valore atteso: In questo rettangolo sono presenti stelle, perciò ci aspettiamo che la somma dei tipi delle stelle sia pari a 702
   Valore ottenuto: 702
2. Ricerca fallimentare nel rettangolo di base 4 e altezza 12, , centrato in (0,0)
   Valore atteso: In questo rettangolo NON sono presenti stelle, perciò ci aspettiamo che la somma dei tipi delle stelle sia pari a 0
   Valore ottenuto: 0

### *Requisito 11*

**Descrizione Class Diagram:**

Dal menu della HomeGUI viene istanziata il controller CalcolaDistanzeSegConController che a sua volta istanzia l'interfaccia grafica CalcolaDistanzeSegConGUI. Una volta inseriti i dati e avviata la richiesta di calcolo viene eseguita la funzione calcolaDistanze() propria dell'istanza della classe UtenteRegistrato ottenuta tramite getInstance() del Singleton UtenteConnesso. Inoltre abbiamo utilizzato la funzione trova Segmenti() per recuperare e mostrare dentro la choiceBox tutti i segmenti relativi al filamento inserito. Le interrogazioni sul database vengono eseguite all'interno del FileDao dalle funzioni calcola Distanze() e trovaSegmenti().

**Test effettuato**: Controlliamo che per il segmento 257 del filamento 409 di Herschel vengano calcolate distanze esatte. Non è possibile scegliere un segmento non appartenente al filamento specificato, perché nell'applicazione vengono automaticamente elencati i segmenti del filamento in una choiche box.
Valori attesi: primo punto dista: 0.01469, ultimo punto dista: 0.01472 Entrambi con un errore di 0.00002.
Valori ottenuti: 0.01469, 0.01472

### *Requisito 12*

**Descrizione Class Diagram:**

Dal menu della HomeGUI viene istanziata il controller CalcolaDistanzeStellaSpinaController che a sua volta istanzia l'interfaccia grafica CalcolaDistanzeStellaSpinaGUI. Una volta inseriti i dati e avviata la richiesta di

calcolo viene eseguita la funzione calcolaDistanze() propria dell'istanza della classe UtenteRegistrato ottenuta tramite getInstance() del Singleton UtenteConnesso. Le interrogazioni sul database vengono eseguite all'interno del FileDao dalla funzione calcolaDistanze(), che utilizza l'entità StellaSpina(generalizzazione della classe Stella) per creare dei record che vanno a riempire la tableView dell'interfaccia grafica.

**Test effettuati:**

1. Calcolo delle distanze delle stelle interne a un filamento rispetto alla spina dorsale. Vengono calcolate le distanze delle stelle nel filamento 409 e ordinate per distanza.
   Valore Attesi: calcolaDistStellaSpina() ritorna un valore boolean vero se i calcoli hanno successo, falso se falliscono. il filamento esiste quindi ci aspettiamo true.
   Valore Ottenuto: true
2. Calcolo fallito delle distanze delle stelle interne a un filamento rispetto alla spina dorsale. Cerchiamo le distanze delle stelle interne al filamento 0, inesistente.
   Valore Atteso: poiché il filamento non esiste non dovrebbe essere trovata nessuna stella e attendiamo il valore false.
   Valore ottenuto: false