Analisi dell’esercitazione del fine settimana

Il primo passo per la risoluzione del progetto è stato la realizzazione del modello Entità- Relazioni,

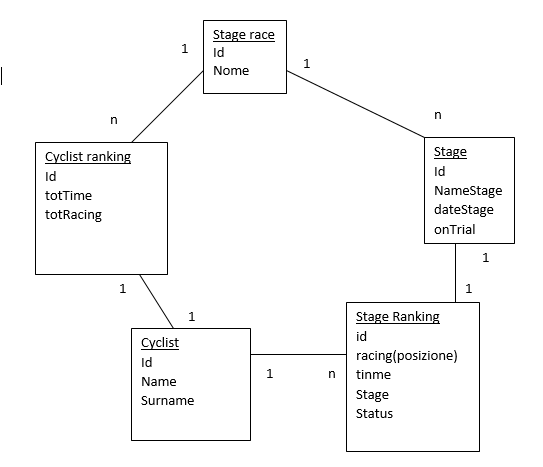
le entità coinvolte sono:

1. Stage race: campionato ciclistico
2. Stage: singola tappa
3. Stage Ranking: tempo e posizione di un ciclista in una singola tappa
4. Cyclist: informazioni sul ciclista
5. Cyclist Ranking: tempo e posizione totale di un ciclista nel campionato

Tra Cyclist e Stage ho pensato ad una relazione n-n, ovvero ogni ciclista partecipa a più Stage, e ad ogni Stage partecipano più ciclisti. Questa relazione l’ho sviluppata come una relazione 1-n tra Cyclist e StageRanking (che contiene le informazioni del ciclista riguardo la posizione ed il tempo per ogni Stage), ed 1-1 tra Stage Ranking e Stage, dove Stage Ranking contiene informazioni sullo Stage.

Un’altra relazione n-n tra Cyclist e Stage Race, in quanto ogni ciclista partecipa a più Stage Race e ad ogni Stage Race partecipano più ciclisti, anche in questo caso l’ho sviluppata come una relazione 1-1tra Cyclist e Cyclist Ranking (che contiene le informazioni riguardo il ciclista per il campionato) ed 1-n tra Cyclist Ranking e Stage Race, in quanto ad ogni Stage Race ci sono più piazzamenti di diversi ciclisti.

Inoltre ho aggiunto una relazion 1-n tra Stage Race e Stage in quanto ogni stage race è composto da più stage, questa relazione non è risultata utile ai fini del progetto, ma potrebbe servire per sviluppi futuri per facilitare le operazioni



Il progetto è stato sviluppato in due estensioni:

* cyclingracinguiip: contiene i dao ed i service
* cyclingracingfacadeuiip: contiene i facade

Per prima cosa nell’estensione cyclingracinguiip ho inserito nel file cyclingracinguiip-item.xml, gli item, le relazioni e due enum (Status (RITIRED, NOTRITIRED) , OnTrial(YES, NO) che verifica se uno stage è stato processato), dopo ho creato un file cyclingracing.impex per inizializzare i valori nel database.

Nella prima estensione ho creato per prima cosa i dao, in particolare:

* DefaultStageDao: restituisci tutti i stage non processati
* DefaultStageRankingDao: restituisci il tempo di uno stage per ogni ciclista (sarebbe stato più utile prendere la somma dei tempi dei vari stage per ogni ciclista, ma non riesco ad ottenere il valore sum(time) da service)
* DefaoultCyclistDao: restituisci tutti i ciclisti
* DefaultCyclistRankingDao: restituisci i Cyclist Ranking ordinati per time, utile per la creazione del file csv dei risultati

Dopo i dao ho creato i Service per ogni Dao, e successivamente il CronJob

Il CronJob viene schedulato ogni 10 minuti, controlla i Stage creati dopo l’ultimo CronJob, e che non siano stati processati, dopo di che per ogni ciclista calcolo la somma dei tempi dei relativi Stage non ancora processati e la sommo al tempo totale del relativo ciclista nella tabella CyclistRanking, infine setto a “YES” l’attributo OnTrial di questi stage in modo da risultare processati, per le varie operazioni e controlli del CronJob mi sono servito delle Dao create in precedenza.

Per ogni Dao, Service e CronJob ho creato i bean corrispondenti all’interno di cyclingracinguiip-spring.xml.

Non sono riuscito a creare l’event listener che aggiorna il file csv relativo alla classifica del campionato.

Nella seconda estensione (cyclingracingfacadeuiip) ho creato i bean data per ogni tabella, e i populator per la conversione da Model a Data, non sono riuscito a creare i Facade per ogni operazione perché non riconosce i Service creati nella prima estenzione ( i Model per i populator invece li riconosce ?!?)