***ANALISI ESERCITAZIONE HYBRIS WEEKEND FRANCESCO CANIGLIA***

In questo documento mi limiterò ad elencare i punti più particolari dell’esecuzione dell’esercizio.

1. Per l’interazione delle due extensions ho modificato il file extensioninfo.xml dell’extension *cyclingraceFacade*, inserendo la riga **<requires-extension name="cyclingraceDaoService"/>,** anche se nel Facade stesso, nonostante riesca ad implementare i Converter/Populator grazie alla visibilità dei Model presenti nella prima extension, non riesca ad implementare i metodi per interrogare il DB, in quanto non riesco ad importare le classi che implementano i suddetti all’interno del layer Service.
2. Per quanto riguarda l’Interceptor, ho scelto di implementare l’interfaccia InitDefaultsInterceptor, utilizzando la classe Calendar per il set della data.
3. La classe che implementa il CronJob è presente nel package my.cyclingrace.Stage.Service.Impl (ho pensato di inserirlo in questo pacchetto il quanto il CornJob è comunque inerente allo StageModel). Per effettuare il controllo della presenza di nove Stages all’interno del DB, ho utilizzato il metodo *getCreationTime* (restituisce un oggetto di tipo Date) per conoscere la data e ora di creazione del record nel DB, e *getEndTime* per data e ora della fine del precedente CronJob (ho supposto che questo metodo funzionasse così, anche perché per il CronJob in esecuzione non è possibile conoscere il suo EndTime). Con il metodo *getTime* su entrambi gli oggetti di tipo Date, trasformo tutto in ms (millisecondi), potendo così effettuare un confronto utile. Se tale controllo risulta positivo (presenza di una o più nuove Stages) effettuo l’aggiornamento dei tempi totali per ogni ciclista, assicurandomi di metchare le StageRanking con le opportune Stages (quelle nuove), con lo StageRace di competenza. Inoltre è stato facilitante implementare nel Dao relativo al CyclistRanking una FlexibleSearchQuery che restituisse solo gli elementi relativi ad un dato StageRace (coerentemente alle nuove Stages); questo ha appunto facilitato la procedura di ordinamento delle posizioni dei vari ciclisti all’interno della classifica, ordinamento implementato in un metodo esterno (static). Infine ho utilizzato il modelService.save() sulla lista ordinata (ho supposto che così verrà effettuato un INSERT\_UPDATE all’interno della table CyclistRanking, anche se non ne sono sicuro, ma non ho trovato nulla online).
4. Per quanto riguarda l’implementazione dell’Event/Listener, ho lanciato l’evento dalla classe che implementa il CronJob solo se questo rileva nuove Stage, ho definito le due classi per l’Event e per il Listener, senza però riuscire ad effettuare l’export del file csv. Ho comunque settato il path tramite chiave valore all’interno del *project.properties.*

29/10/2017 FRANCESCO CANIGLIA