TESI DI LAUREA IN INGEGNERIA INFORMATICA

Progetto ed implementazione del backend relazionale di una piattaforma HRM in D.B. Group S.p.A.

Relatore: Ch.mo Prof. Sergio Congiu

Correlatore: Dott. Enrico Rinolfi – D.B. Group S.p.A.

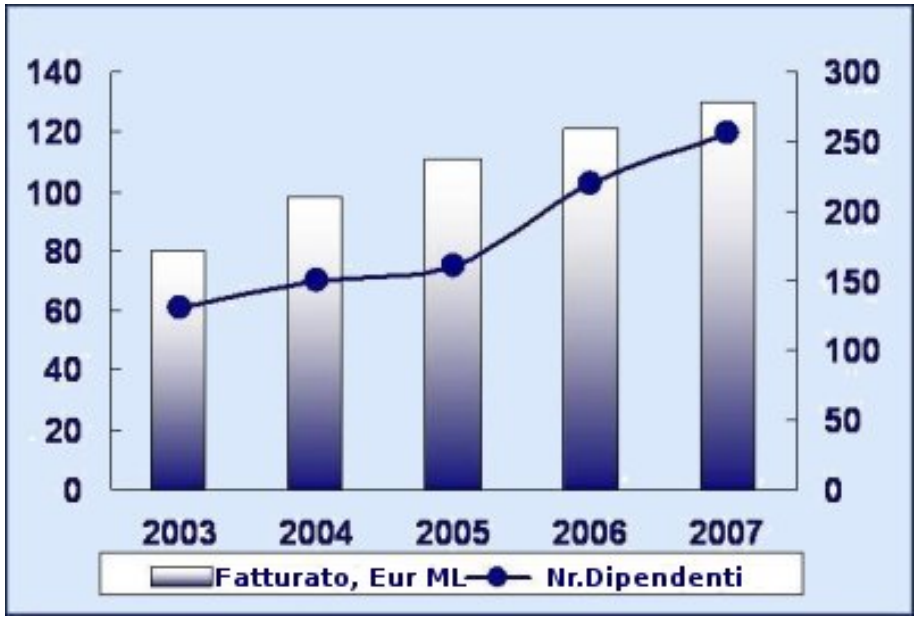
Laureando: Vincenzo Derobertis

A.A. 2008-2009

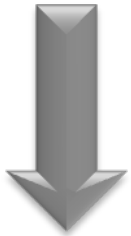
D.B. Group S.p.A.

Azienda operante nel settore della logistica e dei trasporti

- Fondata come azienda a conduzione familiare nel 1950;
- Oggi multinazionale presente in tutti i cinque continenti, con 7 sedi italiane, di cui la principale è a Montebelluna (TV);
- Oltre 250 dipendenti nel mondo;
- Più di 120 milioni di euro di fatturato annuale.



HRM: Human Resource Management



- **Controllo del rispetto del contratto lavorativo**
  - Acquisizione delle timbrature di ingresso e uscita
  - Verifica dello scostamento rispetto ai limiti imposti dall'orario lavorativo giornaliero
- **Gestione dei *leave* richiesti dal dipendente (ferie, permessi, ecc.)**

- Controllo del rispetto del contratto lavorativo

  - Gestito tramite software soggetto al pagamento di licenza;
  - Nessun controllo del rispetto dell'orario lavorativo, solo segnalazione mancanza timbrature;
  - La maggior parte dei compiti è delegata al Responsabile HRM;

- Gestione dei *leave* del dipendente

  - La richiesta e l'approvazione dei *leave* di ogni dipendente è effettuata tramite la piattaforma OrangeHRM ma la loro gestione è manuale;

OrangeHRM: è una *business solution* per la gestione delle Risorse Umane per aziende di medio-piccole dimensioni.
Rilasciato con licenza GNU, è multipiattaforma e si compone di diversi moduli di cui i più importanti sono:

- Modulo di amministrazione
- Modulo di gestione delle informazioni personali
- Modulo di reportistica
- Modulo di gestione delle timbrature giornaliere
- Modulo di gestione dei permessi

È sviluppato utilizzando:

- **MySQL (DBMS)**
- **PHP**
- **Apache HTTP Server**

OrangeHRM è la piattaforma *open* scelta da D.B. Group attorno alla quale costruire la propria soluzione di Gestione del Personale.

Problema: OrangeHRM non prevede alcun meccanismo di elaborazione delle informazioni da esso gestite: non prevede la definizione degli orari lavorativi dei dipendenti e, di conseguenza, non è in grado di effettuare alcun controllo sulle timbrature effettuate e sulla consistenza dei *leave* richiesti.

Obiettivo: sviluppare una soluzione software complementare ad OrangeHRM che sia in grado di automatizzare la maggior parte dei compiti svolti manualmente dal Responsabile HRM, eliminando la necessità di software terzi soggetti a licenze:

Problema: OrangeHRM non prevede alcun meccanismo di elaborazione delle informazioni da esso gestite: non prevede la definizione degli orari lavorativi dei dipendenti e, di conseguenza, non è in grado di effettuare alcun controllo sulle timbrature effettuate e sulla consistenza dei *leave* richiesti.

Obiettivo: sviluppare una soluzione software complementare ad OrangeHRM che sia in grado di automatizzare la maggior parte dei compiti svolti manualmente dal Responsabile HRM, eliminando la necessità di software terzi soggetti a licenze:

- Acquisizione timbrature dai dispositivi installati in azienda;
- Controllo e correzione dei *leave* richiesti presenti nel database di OrangeHRM, alla luce delle particolarità dei contratti lavorativi in D.B. Group (es: 39 ore lavorative settimanali e non 40);
- Analisi della giornata lavorativa del dipendente al fine di generare un report riepilogativo della stessa con il totale delle ore lavorate ordinarie, dei permessi richiesti e degli straordinari fatti.

Il cuore della suddetta soluzione software è chiamato *backend relazionale.*

- *Backend*, perché distinto dal *frontend* grafico che ne facilita l'amministrazione da parte del Responsabile HRM, sviluppato a parte;
- *Relazionale*, perché operante su molteplici database relazionali sia Access (JET) che MySql.

Il cuore della suddetta soluzione software è chiamato *backend relazionale.*

- *Backend*, perché distinto dal *frontend* grafico che ne facilita l'amministrazione da parte del Responsabile HRM, sviluppato a parte;
- *Relazionale*, perché operante su molteplici database relazionali sia Access (JET) che MySql.

Principi cardine del progetto:

- *Modularità*
- *Semplicità*
- *Portabilità*

Il cuore della suddetta soluzione software è chiamato *backend relazionale.*

- *Backend*, perché distinto dal *frontend* grafico che ne facilita l'amministrazione da parte del Responsabile HRM, sviluppato a parte;
- *Relazionale*, perché operante su molteplici database relazionali sia Access (JET) che MySql.

Principi cardine del progetto:

- *Modularità* ⇒
  - *MDBAccess* *(ril. timbrature)*
  - *LeaveAdj* *(correzione leave)*
  - *HRMFill* *(analisi complessiva)*
- *Semplicità*
- *Portabilità*

Il cuore della suddetta soluzione software è chiamato *backend relazionale.*

- *Backend*, perché distinto dal *frontend* grafico che ne facilita l'amministrazione da parte del Responsabile HRM, sviluppato a parte;
- *Relazionale*, perché operante su molteplici database relazionali sia Access (JET) che MySql.

Principi cardine del progetto:

- *Modularità* ⇒
  - *MDBAccess* (ril. timbrature)
  - *LeaveAdj* (correzione leave)
  - *HRMFill* (analisi complessiva)
- *Semplicità* ⇒
  - *Sviluppo in Visual C#*
  - *Filosofia KISS (Keep it short and simple)*
- *Portabilità*

Il cuore della suddetta soluzione software è chiamato *backend relazionale.*

- *Backend*, perché distinto dal *frontend* grafico che ne facilita l'amministrazione da parte del Responsabile HRM, sviluppato a parte;
- *Relazionale*, perché operante su molteplici database relazionali sia Access (JET) che MySql.

Principi cardine del progetto:

- *Modularità* ⇒
  - *MDBAccess* *(ril. timbrature)*
  - *LeaveAdj* *(correzione leave)*
  - *HRMFill* *(analisi complessiva)*
- *Semplicità* ⇒
  - *Sviluppo in Visual C#*
  - *Filosofia KISS (Keep it short and simple)*
- *Portabilità* ⇒
  - *Assenza di system calls*
  - *Eseguibile su più SO*

L'architettura dei tre moduli è simile e prevede tre fasi distinte:

1) *Parameter checking and error recovery*: acquisisce i parametri di inizializzazione del modulo, controllando lo stato della precedente esecuzione e, se erronea, ripristina lo stato precedente dei database modificati al fine di evitare inconsistenze. *Fault Tolerance* garantita in ognuno dei moduli.

2) *Data retrieval:* raccolta ed organizzazione in strutture dati interne dei dati utilizzati durante l'elaborazione.

3) *Data Processing:* elaborazione dati e scrittura risultati finali su database

Fasi ottimizzabili in presenza di più cores, nei moduli LeaveAdj e HRMFill

Il modulo MDBAccess

- Applicazione *single threaded*: responsabile della lettura del database Access scritto dal modulo di controllo delle centraline di rilevamento timbrature presenti in azienda:

Invio dati tramite rete Ethernet al computer di controllo

CARDEVENTDATE	CARDEVENTTIME	PERSONID	READERID
20090120	104006	600-1003	1
20090120	104008	600-1003	2
20090120	094006	600-1003	2
20090122	104020	600-1003	1
20090122	184021	600-1003	2
20090124	084029	600-1003	1
20090125	104035	600-1003	1
20090125	184038	600-1003	2
20090127	104041	600-1003	1
20090127	184808	600-1004	2
20090127	184830	600-1004	1
20090127	184900	600-1004	1
20090131	104818	600-1005	1
20090201	044818	600-1005	2
20090201	034818	600-1005	1

Ad ogni record della tabella corrisponde una timbratura di ingresso o uscita.
Nessuna associazione tra ingressi e corrispondenti uscite.

- Compito del modulo è riempire la tabella *hs_hr_attendance* nel database MySql di OrangeHRM, il cui record è invece una coppia di timbrature corrispondenti (*ingresso,uscita*):

Field	Type
attendance_id	int(11)
employee_id	int(11)
punchin_time	datetime
punchout_time	datetime
in_note	varchar(250)
out_note	varchar(250)
status	enum('0','1')

Per fare questo MDBAccess deve suddividere, dipendente per dipendente, i record letti dal database Access in due gruppi, uno comprendente le timbrature di ingresso e l'altro quelle di uscita, quindi ordinarli cronologicamente ed associarli logicamente. L'associazione avviene per mezzo di due variabili *pivot* che scandiscono gli elementi dei due gruppi.

Il modulo LeaveAdj

- Applicazione *multi threaded*: responsabile della correzione dei *leave* richiesti e salvati in *hs_hr_leave* al fine di garantirne la consistenza con i criteri imposti dalla politica aziendale.

leave_id	leave_date	leave_length_hours	leave_length_days	leave_status	leave_type_id	employee_id	start_time	end_time
1	2009-08-24	2.00	0.25	1	LTY001	1	07:30:00	09:30:00
2	2009-08-24	4.00	0.50	1	LTY001	1	11:30:00	15:30:00
3	2009-08-24	2.00	0.25	1	LTY001	1	19:30:00	21:30:00
4	2009-08-28	8.00	1.00	1	LTY001	1	08:00:00	19:00:00

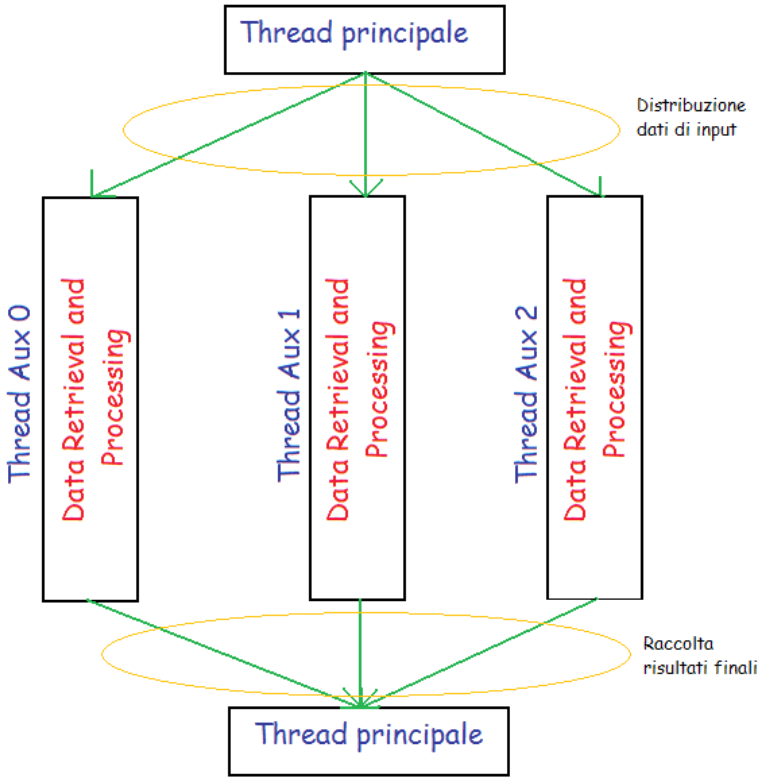
I vincoli da rispettare nell'analisi dei *leave* sono:

- Tutti i *leave* collocati al di fuori dell'orario lavorativo devono essere annullati;
- *Leave* parzialmente disposti al di fuori dell'orario lavorativo devono essere corretti al fine di eliminarne la parte in eccesso;
- Non sono ammesse sovrapposizioni tra *leave*;
- Nel caso di dipendenti in determinati stati lavorativi (maternità, allattamento), è necessario inserire automaticamente un *leave* di durata opportuna.

- L'analisi dei *leave* di un dipendente è svincolata da quella dei *leave* di altri dipendenti ➔ Le fasi di *data retrieval* (recupero orario lavorativo e *leave* da analizzare) e *data processing* presentano un elevato grado di parallelismo.

- Il thread principale legge dalla tabella *hs_hr_employee* l'elenco dei dipendenti e li "distribuisce" ai vari thread ausiliari (di numero pari al numero di cores rilevati):

```
for (int i = 0; i < employeeList.Count; i++)
{
        threadList[j].Add(employeeList[i]);
        j = (j + 1) % cores;
}
```

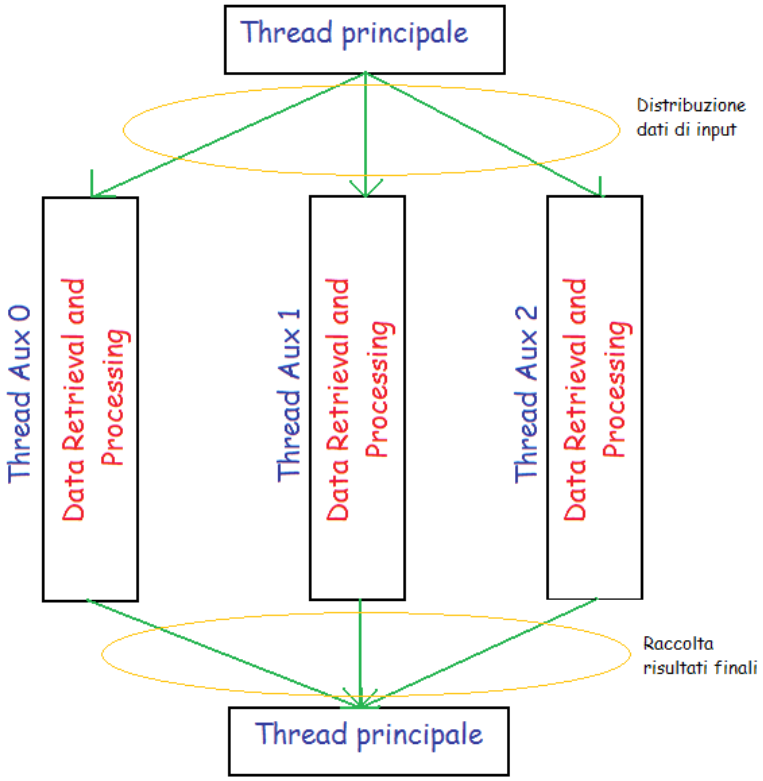


- L'analisi dei *leave* di un dipendente è svincolata da quella dei *leave* di altri dipendenti ➔ Le fasi di *data retrieval* (recupero orario lavorativo e *leave* da analizzare) e *data processing* presentano un elevato grado di parallelismo.

- Il thread principale legge dalla tabella *hs_hr_employee* l'elenco dei dipendenti e li "distribuisce" ai vari thread ausiliari (di numero pari al numero di cores rilevati):

```
for (int i = 0; i < employeeList.Count; i++)
{
        threadList[j].Add(employeeList[i]);
        j = (j + 1) % cores;
}
```

I thread ausiliari completano l'elaborazione e ritornano i dati finali da scrivere nel database al thread principale.

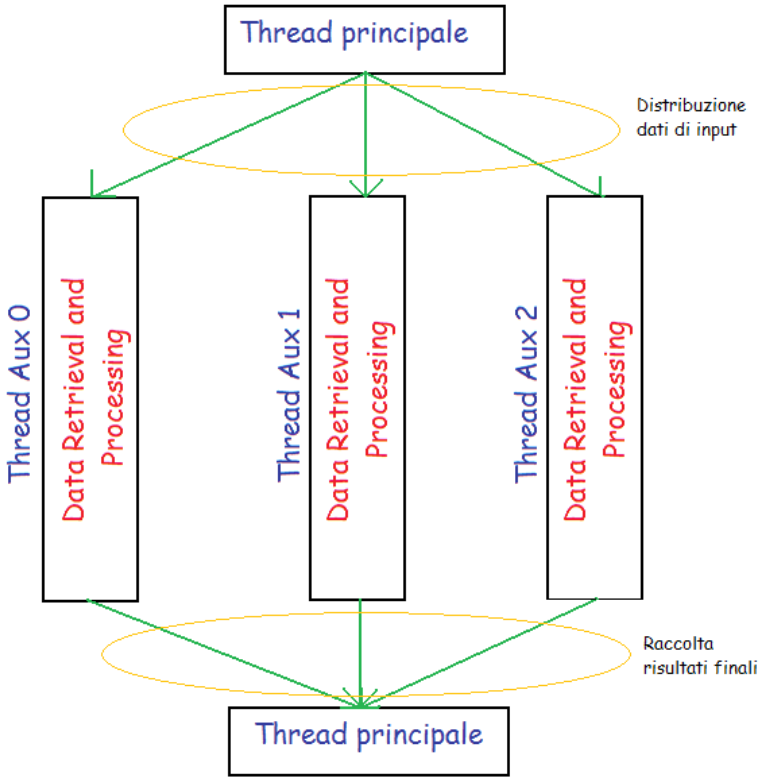


- L'analisi dei *leave* di un dipendente è svincolata da quella dei *leave* di altri dipendenti ➔ Le fasi di *data retrieval* (recupero orario lavorativo e *leave* da analizzare) e *data processing* presentano un elevato grado di parallelismo.

- Il thread principale legge dalla tabella *hs_hr_employee* l'elenco dei dipendenti e li "distribuisce" ai vari thread ausiliari (di numero pari al numero di cores rilevati):

```
for (int i = 0; i < employeeList.Count; i++)
{
        threadList[j].Add(employeeList[i]);
        j = (j + 1) % cores;
}
```

I thread ausiliari completano l'elaborazione e ritornano i dati finali da scrivere nel database al thread principale.



Esecuzione asincrona
Accesso a risorse condivise in sola lettura
Accesso in lettura al database concorrenziale (isolamento *Repeatable Read*) ➔ Nessun meccanismo di locking necessario

Il modulo HRMFill

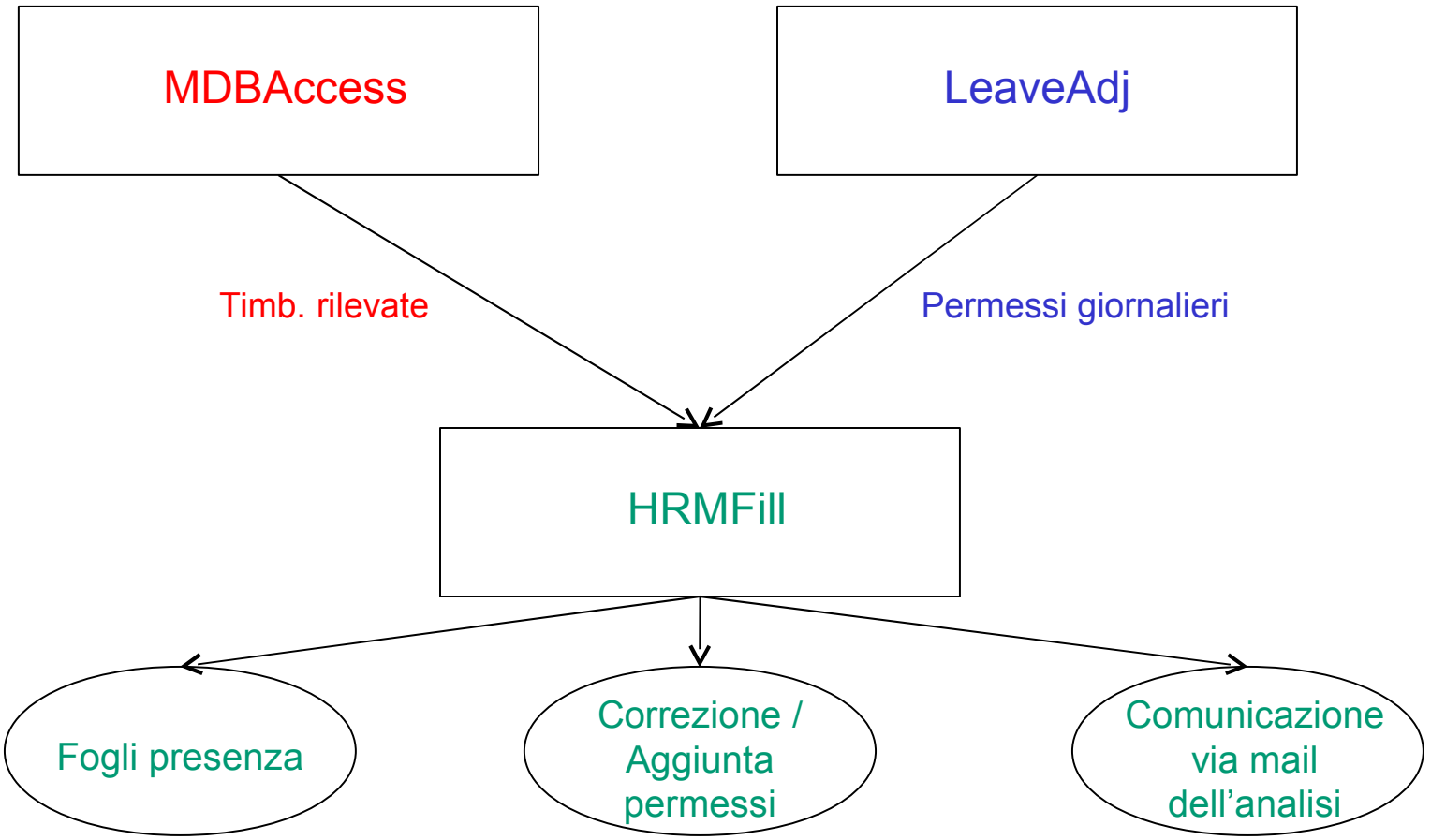
- Applicazione *multi threaded*: responsabile della generazione dei fogli di presenza giornalieri. Analizzando le timbrature rilevate ed i *leave* concessi si determinano:

- Ore ordinarie lavorate;
- Ritardi;
- Straordinari (al 30% o 50% della paga ordinaria in base a quando sono stati effettuati);
- Riepilogo permessi chiesti e goduti;
- Correzione automatica dei *leave* non goduti;
- Compensazione automatica permessi/straordinari.
- Comunicazioni finali via mail al dipendente e al Respnsabile HRM.

id	employee_id	day	number_of_punches	working_hours	punches	worked_hours
1	1	2009-08-24	4	8.00	|8:27:00 - 11:30:00||14:30:00 - 18:30:00|	7.00

LTY001	LTY002	LTY003	LTY004	LTY005	LTY006	EXTR30	EXTR50
1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

- L'impianto architetturale del modulo è molto simile a quello del modulo *LeaveAdj*, così come la gestione del parallelismo nelle fasi di *data retrieval* e *data processing* ➔ Esecuzione asincrona dei thread ausiliari.



Conclusioni

- Il software ha permesso di compensare le lacune di OrangeHRM e automatizzare molti aspetti dell'analisi ➔ semplificazione notevole del lavoro del Responsabile HRM;
- Notevole risparmio economico per l'azienda (quantificato in diverse migliaia di euro annuali) grazie all'analisi accurata delle timbrature giornaliere e all'assenza di costi di licenze e gestione di terze parti;
- Vantaggi per il dipendente per i minori errori relativi alla gestione dei *leave*.

Conclusioni

- Il software ha permesso di compensare le lacune di OrangeHRM e automatizzare molti aspetti dell'analisi ➔ semplificazione notevole del lavoro del Responsabile HRM;
- Notevole risparmio economico per l'azienda (quantificato in diverse migliaia di euro annuali) grazie all'analisi accurata delle timbrature giornaliere e all'assenza di costi di licenze e gestione di terze parti;
- Vantaggi per il dipendente per i minori errori relativi alla gestione dei *leave*.

Possibili upgrade futuri

- Rilevazione e analisi delle timbrature in *real time*;
- Miglioramento nella gestione del parallelismo ➔ scrittura finale su database eseguita, se possibile, dagli stessi thread ausiliari grazie ad appositi meccanismi di locking con conseguente aumento di efficienza ed eliminazione (almeno parziale) della serializzazione della fase di scrittura.