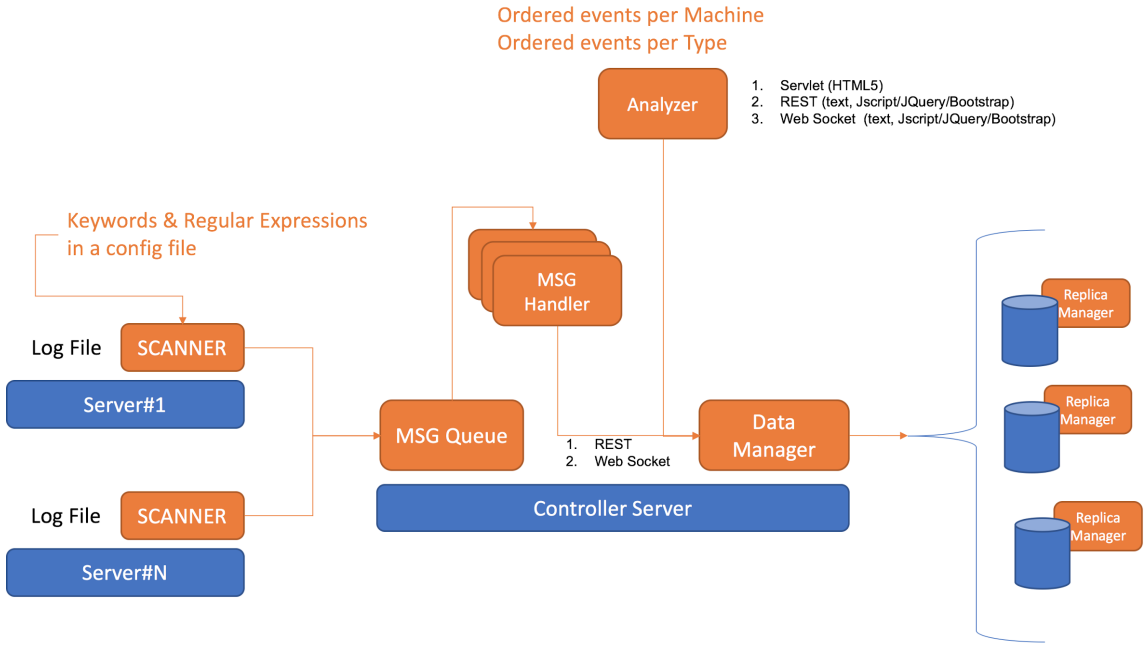
Architettura di riferimento



Configurazione DOCKER per RabbitMQ

Detto HOSTNAME l'hostname del container, ad esempio "my-rabbit".
Detto NAME il nome del container, ad esempio "rabbit".
Detto PATH_LOCALE il percorso in cui verrano salvati i dati di RabbitQM nel file system locale, ad esempio "/Users/utente/DockerImage/RabbitMQ"

Il comando per l'esecuzione di RabbitMQ è:

docker run -d --hostname HOSTNAME --name NAME -p 15672:15672 -p 5672:5672 -v PATH_LOCALE:/var/lib/rabbitmq rabbitmq:3-management

All'indirizzo
http://localhost:15672 si accederà ai servizi di management della coda (guest/guest)
Da API bisogna usare la porta 5672 per accedere alla coda.

Configurazione DOCKER per MYSQL

Detto NAME il nome del container, ad esempio "mysql".

Detto PATH_LOCALE il percorso in cui verrano salvati i database nel file system locale, ad esempio “/Users/utente/DockerImage/MySQL”

Il comando per l’esecuzione di MySQL è:

docker run --name NAME -v PATH_LOCALEL:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=passRoot -p 3306:3306 -d mysql:5.7

E’ possibile usare JDBC per l’accesso a mysql utilizzando root/passRoot e 3306 come porta.

Di seguito il dettaglio sui diversi home work.

Lo studente fornisca come risultato finale

- Il codice delle classi implementate
- Un documento nel quale vengono spiegate le scelte progettuali adottate
- Una rappresentazione architetturale dell’applicazione distribuita nella quale vengono evidenziati i vari livelli coinvolti (Presentation Layer, Business Layer, Legacy Layer)
- La rappresentazione UML delle clbassi principali
- Eventuali README

ATTENZIONE!!
I diversi gruppi da 2 studenti potranno scegliere l’homework che preferiscono rispettando la precedenza dettata dall’ordine con cui sono inseriti nell’elenco che mi è stato fornito

per lo studente singolo (**Andrea Claudio Sebastian Susinna**) ho invece predisposto un compito specifico

Home Work 1

SCANNER#1: programma Java che analizza ogni nuova riga di uno specifico log file alla ricerca di specifiche keywords o **espressioni regolari** (da file di conf/properties).
Per ogni match, ID macchina, timestamp e riga sono pubblicati sul P/S Queue.

MSG QUEUE: riceve i dati (asynch) dagli SCANNER e li invia ai vari MSG Handler

- MSG Handler **Content-based:** salva sul db usando i servizi del data manager (synch).

DB: Mysql/MongoDB

IL **DATA MANAGER**, sviluppato a componenti EJB, rappresenta l'interfaccia Read (per Analyser) / Write (MSG Handler) per i database.
Il Data Manager espone i propri servizi stateless mediante interfaccia REST.

5 repliche

- **Replicazione attiva con lettura da db più vicino** e fault detector basato su **gossip**
- **Ciascuna replica ha un ejb posto nel nodo sul quale risiede il DB il cui compito è quello di eseguire tutte le query e gli update richiesti dal DataManager**

ANALYSER "acts as CONTROLLER".
Mette a disposizione all'utente finale query relative ai dati memorizzati.
Esempio

- Eventi ordinati nel tempo per tutte le macchine
- Eventi specifici ordinati nel tempo per tutte le macchine
- Eventi ordinati nel tempo per una singola macchina
- Eventi specifici ordinati nel tempo per una singola macchina
- (Tempo, evento, macchina come parametri principali)
- Dati su info per macchina
- Dati su info per l'intero sistema
- Dati su specifiche finestre temporali

Home Work 8

SCANNER#2: programma Java che esegue uno script per il recupero di informazioni relative alle performance (Cpu/Mem, disk I/O, network I/O)

MSG QUEUE: riceve i dati (asynch) dagli SCANNER e li invia ai vari MSG Handler

1. MSG Handler Topic-based: salva sul db usando i servizi del data manager (synch).
2. MSG Handler Content-based: salva sul db usando i servizi del data manager (synch).

DB: Mysql/MongoDB

IL **DATA MANAGER**, sviluppato a componenti EJB, rappresenta l'interfaccia Read (per Analyser) / Write (MSG Handler) per i database.
Il Data Manager espone i propri servizi stateless mediante interfaccia WebSocket.
5 repliche

- **Replicazione attiva con lettura bilanciata (un db diverso ogni volta) e fault detector basato su gossip**
- **Ciascuna replica ha un ejb posto nel nodo sul quale risiede il DB il cui compito è quello di eseguire tutte le query e gli update richiesti dal DataManager**

ANALYSER "acts as CONTROLLER".
Mette a disposizione all'utente finale query relative ai dati memorizzati.
Esempio

- Eventi ordinati nel tempo per tutte le macchine
- Eventi specifici ordinati nel tempo per tutte le macchine
- Eventi ordinati nel tempo per una singola macchina
- Eventi specifici ordinati nel tempo per una singola macchina
- (Tempo, evento, macchina come parametri principali)
- Dati su info per macchina
- Dati su info per l'intero sistema

Dati su specifiche finestre temporali

Home Work 2

SCANNER#1: programma Java che analizza ogni nuova riga di uno specifico log file alla ricerca di specifiche keywords o **espressioni regolari** (da file di conf/properties).
Per ogni match, ID macchina, timestamp e riga sono pubblicati sul P/S Queue.

MSG QUEUE: riceve i dati (asynch) dagli SCANNER e li invia ai vari MSG Handler

- MSG Handler Content-based: salva sul db usando i servizi del data manager (synch).
-

DB: Mysql/MongoDB

IL **DATA MANAGER**, sviluppato a componenti EJB, rappresenta l'interfaccia Read (per Analyser) / Write (MSG Handler) per i database.
Il Data Manager espone i propri servizi statetless mediante interfaccia REST
5 repliche

- **Replicazione quorum-based e fault detector con timeout**
- **A ciascun replica corrisponde un ejb istanziato sul DataManager che funge da connettore verso il database remoto**

ANALYSER "acts as CONTROLLER".
Mette a disposizione all'utente finale query relative ai dati memorizzati.
Esempio

- Eventi ordinati nel tempo per tutte le macchine
- Eventi specifici ordinati nel tempo per tutte le macchine
- Eventi ordinati nel tempo per una singola macchina
- Eventi specifici ordinati nel tempo per una singola macchina
- (Tempo, evento, macchina come parametri principali)
- Dati su info per macchina
- Dati su info per l'intero sistema
- Dati su specifiche finestre temporali

Home Work 9

SCANNER#2: programma Java che esegue uno script per il recupero di informazioni relative alle performance (Cpu/Mem, disk I/O, network I/O)

MSG QUEUE: riceve i dati (asynch) dagli SCANNER e li invia ai vari MSG Handler

- MSG Handler Topic-based: salva sul db usando i servizi del data manager (synch).
- MSG Handler Content-based: salva sul db usando i servizi del data manager (synch).

DB: Mysql/MongoDB

IL **DATA MANAGER**, sviluppato a componenti EJB, rappresenta l'interfaccia Read (per Analyser) / Write (MSG Handler) per i database.
Il Data Manager espone i propri servizi stateless tramite interfaccia REST

5 repliche

- Replicazione passiva (**primary-Backup** "classico" con garanzia di linearizzabilità - con lettura/scrittura da primary e scrittura committed) con **Paxos** come algoritmo di leader election e **fault detector** basato time-out
- **A ciascun replica corrisponde un ejb istanziato sul DataManager che funge da connettore verso il database remoto**

ANALYSER "acts as CONTROLLER".
Mette a disposizione all'utente finale query relative ai dati memorizzati.
Esempio

- Eventi ordinati nel tempo per tutte le macchine
- Eventi specifici ordinati nel tempo per tutte le macchine
- Eventi ordinati nel tempo per una singola macchina
- Eventi specifici ordinati nel tempo per una singola macchina
- (Tempo, evento, macchina come parametri principali)
- Dati su info per macchina
- Dati su info per l'intero sistema
- Dati su specifiche finestre temporali

Home Work 3

SCANNER#1: programma Java che analizza ogni nuova riga di uno specifico log file alla ricerca di specifiche keywords o **espressioni regolari** (da file di conf/properties).
Per ogni match, ID macchina, timestamp e riga sono pubblicati sul P/S Queue.

MSG QUEUE: riceve i dati (asynch) dagli SCANNER e li invia ai vari MSG Handler

- MSG Handler **Topic-based**: salva sul db usando i servizi del data manager (synch).

IL **DATA MANAGER**, sviluppato a componenti EJB, rappresenta l'interfaccia Read (per Analyser) / Write (MSG Handler) per i database.

- Il Data Manager espone i propri servizi stateless tramite interfaccia WebSocket

DB: Mysql/MongoDB

5 repliche

- Replicazione passiva (**primary-Backup con garanzia di sequential consistency**: Scrittura commited da primary, Lettura da Backup) con **Paxos** come algoritmo di leader election e **fault detector** basato time-out
- **A ciascun replica corrisponde un ejb istanziato sul DataManager che funge da connettore verso il database remoto**

ANALYSER "acts as CONTROLLER".
Mette a disposizione all'utente finale query relative ai dati memorizzati.
Esempio

- Eventi ordinati nel tempo per tutte le macchine
- Eventi specifici ordinati nel tempo per tutte le macchine
- Eventi ordinati nel tempo per una singola macchina
- Eventi specifici ordinati nel tempo per una singola macchina
- (Tempo, evento, macchina come parametri principali)
- Dati su info per macchina
- Dati su info per l'intero sistema
- Dati su specifiche finestre temporali

Home Work 10

SCANNER#2: programma Java che esegue uno script per il recupero di informazioni relative alle performance (Cpu/Mem, disk I/O, network I/O)

MSG QUEUE: riceve i dati (asynch) dagli SCANNER e li invia ai vari MSG Handler

- MSG Handler Topic-based: salva sul db usando i servizi del data manager (synch).
- MSG Handler Content-based: salva sul db usando i servizi del data manager (synch).

DB: Mysql/MongoDB

IL **DATA MANAGER**, sviluppato a componenti EJB, rappresenta l'interfaccia Read (per Analyser) / Write (MSG Handler) per i database.

5 repliche

- Replicazione passiva (**primary-Backup Eventually Consistent:** con Scrittura non-commited da primary) con **Paxos** come algoritmo di leader election e **fault detector** basato time-out
- **A ciascun replica corrisponde un ejb istanziato sul DataManager che funge da connettore verso il database remoto**

ANALYSER "acts as CONTROLLER".
Mette a disposizione all'utente finale query relative ai dati memorizzati.
Esempio

- Eventi ordinati nel tempo per tutte le macchine
- Eventi specifici ordinati nel tempo per tutte le macchine
- Eventi ordinati nel tempo per una singola macchina
- Eventi specifici ordinati nel tempo per una singola macchina
- (Tempo, evento, macchina come parametri principali)
- Dati su info per macchina
- Dati su info per l'intero sistema
- Dati su specifiche finestre temporali

Home Work 4

SCANNER#1: programma Java che analizza ogni nuova riga di uno specifico log file alla ricerca di specifiche keywords o **espressioni regolari** (da file di conf/properties).
Per ogni match, ID macchina, timestamp e riga sono pubblicati sul P/S Queue.

MSG QUEUE: riceve i dati (asynch) dagli SCANNER e li invia ai vari MSG Handler

- MSG Handler **Topic-based**: salva sul db usando i servizi del data manager (synch).

IL **DATA MANAGER**, sviluppato a componenti EJB, rappresenta l'interfaccia Read (per Analyser) / Write (MSG Handler) per i database.
Il DataManager espone i propri servizi stateless tramite WebSocket

5 repliche

- Replicazione passiva (**primary-Backup** "classico" con garanzia di linearizzabilità - con lettura/scrittura da primary e scrittura committed) con algoritmo di **leader election** a piacere, **fault detector gossip-based** e un **protocollo di agreement sulle operazioni eseguite**
- **Ciascuna replica ha un ejb posto nel nodo remoto sul quale risiede il DB il cui compito è quello di eseguire tutte le query e gli update richiesti dal DataManager**

ANALYSER "acts as CONTROLLER".
Mette a disposizione all'utente finale query relative ai dati memorizzati.
Esempio

- Eventi ordinati nel tempo per tutte le macchine
- Eventi specifici ordinati nel tempo per tutte le macchine
- Eventi ordinati nel tempo per una singola macchina
- Eventi specifici ordinati nel tempo per una singola macchina
- (Tempo, evento, macchina come parametri principali)
- Dati su info per macchina
- Dati su info per l'intero sistema
- Dati su specifiche finestre temporali

Home Work 11

SCANNER#2: programma Java che esegue uno script per il recupero di informazioni relative alle performance (Cpu/Mem, disk I/O, network I/O)

MSG QUEUE: riceve i dati (asynch) dagli SCANNER e li invia ai vari MSG Handler

- MSG Handler **Content-based**: salva sul db usando i servizi del data manager (synch).

DB: Mysql/MongoDB

IL **DATA MANAGER**, sviluppato a componenti EJB, rappresenta l'interfaccia Read (per Analyser) / Write (MSG Handler) per i database.
Il Data Manager espone i propri metodi stateless tramite interfaccia REST

5 repliche

- Replicazione passiva (**primary-Backup con garanzia di sequential consistency**: **Scrittura committed da primary, Lettura da un Backup**) con algoritmo di **leader election a piacere**, **fault detector gossip-based,** e un **protocollo di agreement sulle operazioni eseguite**
- **Ciascuna replica ha un ejb posto nel nodo remoto sul quale risiede il DB il cui compito è quello di eseguire tutte le query e gli update richiesti dal DataManager**

ANALYSER "acts as CONTROLLER".
Mette a disposizione all'utente finale query relative ai dati memorizzati.
Esempio

- Eventi ordinati nel tempo per tutte le macchine
- Eventi specifici ordinati nel tempo per tutte le macchine
- Eventi ordinati nel tempo per una singola macchina
- Eventi specifici ordinati nel tempo per una singola macchina
- (Tempo, evento, macchina come parametri principali)
- Dati su info per macchina
- Dati su info per l'intero sistema
- Dati su specifiche finestre temporali

Home Work 5

SCANNER#1: programma Java che analizza ogni nuova riga di uno specifico log file alla ricerca di specifiche keywords o espressioni regolari (da file di conf/properties).
Per ogni match, ID macchina, timestamp e riga sono pubblicati sul P/S Queue.

MSG QUEUE: riceve i dati (asynch) dagli SCANNER e li invia ai vari MSG Handler

- MSG Handler Topic-based: salva sul db usando i servizi del data manager (synch).
- MSG Handler Content-based: salva sul db usando i servizi del data manager (synch).

IL **DATA MANAGER**, sviluppato a componenti EJB, rappresenta l'interfaccia Read (per Analyser) / Write (MSG Handler) per i database.

5 repliche

- Replicazione passiva (**primary-Backup Eventually Consistent:** con Scrittura non-commited da primary) con algoritmo di **leader election token based** e f**ault detector gossip-based**
- **Ciascuna replica ha un ejb posto nel nodo sul quale risiede il DB il cui compito è quello di eseguire tutte le query e gli update richiesti dal DataManager**

ANALYSER "acts as CONTROLLER".
Mette a disposizione all'utente finale query relative ai dati memorizzati.
Esempio

- Eventi ordinati nel tempo per tutte le macchine
- Eventi specifici ordinati nel tempo per tutte le macchine
- Eventi ordinati nel tempo per una singola macchina
- Eventi specifici ordinati nel tempo per una singola macchina
- (Tempo, evento, macchina come parametri principali)
- Dati su info per macchina
- Dati su info per l'intero sistema
- Dati su specifiche finestre temporali

Home Work 12

SCANNER#2: programma Java che esegue uno script per il recupero di informazioni relative alle performance (Cpu/Mem, disk I/O, network I/O)

MSG QUEUE: riceve i dati (asynch) dagli SCANNER e li invia ai vari MSG Handler

- MSG Handler Topic-based: salva sul db usando i servizi del data manager (synch).
- MSG Handler Content-based: salva sul db usando i servizi del data manager (synch).

DB: Mysql/MongoDB

Il **DATA MANAGER**, sviluppato a componenti EJB, rappresenta l'interfaccia Read (per Analyser) / Write (MSG Handler) per i database.

5 repliche

- Replicazione passiva (**primary-Backup Eventually Consistent:** con Scrittura non-commited da primary) con **leader election basati su Bully algorthm** e **fault detector** basato su time-out
- **Ciascuna replica ha un ejb posto nel nodo sul quale risiede il DB il cui compito è quello di eseguire tutte le query e gli update richiesti dal DataManager**

ANALYSER "acts as CONTROLLER".
Mette a disposizione all'utente finale query relative ai dati memorizzati.
Esempio

- Eventi ordinati nel tempo per tutte le macchine
- Eventi specifici ordinati nel tempo per tutte le macchine
- Eventi ordinati nel tempo per una singola macchina
- Eventi specifici ordinati nel tempo per una singola macchina
- (Tempo, evento, macchina come parametri principali)
- Dati su info per macchina
- Dati su info per l'intero sistema
- Dati su specifiche finestre temporali

Home Work 6

SCANNER#2: programma Java che esegue uno script per il recupero di informazioni relative alle performance (Cpu/Mem, disk I/O, network I/O)

MSG QUEUE: riceve i dati (asynch) dagli SCANNER e li invia ai vari MSG Handler

- MSG Handler Topic-based: salva sul db usando i servizi del data manager (synch).

IL **DATA MANAGER**, sviluppato a componenti EJB, rappresenta l'interfaccia Read (per Analyser) / Write (MSG Handler) per i database.

DB distribuito su una DHT. prevedere join, fault e leave e strategia di fault tolerance

ANALYSER "acts as CONTROLLER".
Mette a disposizione all'utente finale query relative ai dati memorizzati.
es.

- Eventi ordinati nel tempo per tutte le macchine
- Eventi specifici ordinati nel tempo per tutte le macchine
- Eventi ordinati nel tempo per una singola macchina
- Eventi specifici ordinati nel tempo per una singola macchina
- (Tempo, evento, macchina come parametri principali)
- Dati su info per macchina
- Dati su info per l'intero sistema
- Dati su specifiche finestre temporali

Home Work 13

SCANNER#1: programma Java che analizza ogni nuova riga di uno specifico log file alla ricerca di specifiche keywords o espressioni regolari (da file di conf/properties).
Per ogni match, ID macchina, timestamp e riga sono pubblicati sul P/S Queue.

MSG QUEUE: riceve i dati (asynch) dagli SCANNER e li invia ai vari MSG Handler

- MSG Handler Content-based: salva sul db usando i servizi del data manager (synch).

DB: Mysql/MongoDB

IL **DATA MANAGER**, sviluppato a componenti EJB, rappresenta l'interfaccia Read (per Analyser) / Write (MSG Handler) per i database.

5 repliche

- Replicazione passiva (**primary-Backup con garanzia di sequential consistency**: **Scrittura commited da primary, Lettura da un Backup in modo bilanciato**) con **Paxos** come algoritmo di leader election e **fault detector** basato time-out
- **A ciascun replica corrisponde un ejb istanziato sul DataManager che funge da connettore verso il database remoto**

ANALYSER "acts as CONTROLLER".
Mette a disposizione all'utente finale query relative ai dati memorizzati.
Esempio

- Eventi ordinati nel tempo per tutte le macchine
- Eventi specifici ordinati nel tempo per tutte le macchine
- Eventi ordinati nel tempo per una singola macchina
- Eventi specifici ordinati nel tempo per una singola macchina
- (Tempo, evento, macchina come parametri principali)
- Dati su info per macchina
- Dati su info per l'intero sistema
- Dati su specifiche finestre temporali

Home Work 7

SCANNER#1: programma Java che analizza ogni nuova riga di uno specifico log file alla ricerca di specifiche keywords o espressioni regolari (da file di conf/properties).
Per ogni match, ID macchina, timestamp e riga sono pubblicati sul P/S Queue.

MSG QUEUE: riceve i dati (asynch) dagli SCANNER e li invia ai vari MSG Handler

- MSG Handler Topic-based: salva sul db usando i servizi del data manager (synch).
- MSG Handler Content-based: salva sul db usando i servizi del data manager (synch).

IL **DATA MANAGER**, sviluppato a componenti EJB, rappresenta l'interfaccia Read (per Analyser) / Write (MSG Handler) per i database.

5 repliche

- **Replicazione quorum-based e fault detector gossip-based**
- Replicazione passiva (**primary-Backup con garanzia di sequential consistency**: **Scrittura commited da primary, Lettura da un Backup in modo bilanciato**) con algoritmo di **leader election a piacere** e **fault detector gossip-based** (7)
- **A ciascun replica corrisponde un ejb istanziato sul DataManager che funge da connettore verso il database remoto**

ANALYSER "acts as CONTROLLER".
Mette a disposizione all'utente finale query relative ai dati memorizzati.
Esempio

- Eventi ordinati nel tempo per tutte le macchine
- Eventi specifici ordinati nel tempo per tutte le macchine
- Eventi ordinati nel tempo per una singola macchina
- Eventi specifici ordinati nel tempo per una singola macchina
- (Tempo, evento, macchina come parametri principali)
- Dati su info per macchina
- Dati su info per l'intero sistema
- Dati su specifiche finestre temporali

Home Work 14

SCANNER#2: programma Java che esegue uno script per il recupero di informazioni relative alle performance (Cpu/Mem, disk I/O, network I/O)

MSG QUEUE: riceve i dati (asynch) dagli SCANNER e li invia ai vari MSG Handler

- MSG Handler Topic-based: salva sul db usando i servizi del data manager (synch).

DB: Mysql/MongoDB

IL **DATA MANAGER**, sviluppato a componenti EJB, rappresenta l'interfaccia Read (per Analyser) / Write (MSG Handler) per i database.
Il DataManager espone i propri servizi stateless anche tramite un'interfaccia REST

5 repliche

- **Replicazione quorum-based e fault detector gossip-based**
- **A ciascun replica corrisponde un ejb istanziato sul DataManager che funge da connettore verso il database remoto**

ANALYSER "acts as CONTROLLER".
Mette a disposizione all'utente finale query relative ai dati memorizzati.
Esempio

- Eventi ordinati nel tempo per tutte le macchine
- Eventi specifici ordinati nel tempo per tutte le macchine
- Eventi ordinati nel tempo per una singola macchina
- Eventi specifici ordinati nel tempo per una singola macchina
- (Tempo, evento, macchina come parametri principali)
- Dati su info per macchina
- Dati su info per l'intero sistema
- Dati su specifiche finestre temporali

Home Work per studente singolo

Andrea Claudio Sebastian Susinna

SCANNER#2: programma Java che esegue uno script per il recupero di informazioni relative alle performance (Cpu/Mem, disk I/O, network I/O)

MSG QUEUE: riceve i dati (asynch) dagli SCANNER e li invia ai vari MSG Handler

- MSG Handler Topic-based: salva sul db usando i servizi del data manager (synch).

DB: Mysql/MongoDB

IL **DATA MANAGER**, sviluppato a componenti EJB, rappresenta l'interfaccia Read (per Analyser) / Write (MSG Handler) per i database.

3 repliche

- **Replicazione primary-backup classico**

ANALYSER "acts as CONTROLLER".
Mette a disposizione all'utente finale query relative ai dati memorizzati.
Esempio

- Eventi ordinati nel tempo per tutte le macchine
- Eventi specifici ordinati nel tempo per tutte le macchine
- Eventi ordinati nel tempo per una singola macchina
- Eventi specifici ordinati nel tempo per una singola macchina
- (Tempo, evento, macchina come parametri principali)
- Dati su info per macchina
- Dati su info per l'intero sistema
- Dati su specifiche finestre temporali