Note TrackME Class Diagram

Non ci sono problemi di sincronizzazione dal momento che non c’è modifica concorrente degli stessi oggetti sui due tier.

DataReader crea degli oggetti di classe Data e li salva localmente: periodicamente li invia al server. Dentro il server il router li inserisce nell’oggetto di classe User corrispondente, il data collector svuota il buffer della lista nello User e riempie il database ( pattern producer/consumer)- tra i metodi sendAllData and StoreInDB)

Singleton pattern nel client per Thresholds

Il fatto di inserire I dati nello User e dare la possibilità di metterli nel database solo quando vuole il dataCollector ci permette di semplificare la gestione del database usando un buffer intermedio ( problemi di lock e sincronizzazione al dataabase, accessi concorrenti)

Abbiamo pensato ad un’architettura fat client

Il compito di DataReadingService è quello di creare gli oggetti di classeData

CreateData va in loop: ogni volta che riceve dati crea un oggetto come sopra detto, se non lo riceve dopo un timer segnala il malfunzionamento

Abbiamo deciso di stare vaghi in preferences e nel database perché vogliamo lasciare all’implementazione fianle la scelta di quali possibilità di preference possano essere fornite alla third party, in dipendenza dalla facilità di fornirle nel linguaggio di implementazione scelto, da come il mercato risponde alle nostre release, etc.

NewDataSubscriber è observer di User, quando ci sono modifiche e nuovi dati vengono generati, lui fa richieste utilizzando il RequestsService

Bottom up Strategy, partendo dai componenti più essenziali (Emergency, DataCollector, Model)

Unit Testing dei componenti più importanti,

Le parti erano scollegate le une con le altre???

No chiaro match tra integration, implementation and testing: le parti non erano collegate tra loro e non era chiara la strategia effettiva

(che cosa non andava bene nel DD degli altri nel plan for the implementation???)