C. SOLUZIONE

Partendo dai paragrafi precedenti, riusciamo ora a delineare quelle che sono le possibili componenti software da costruire per i nostri scopi. La prima e indispensabile ÃÍ un indice di risorse storiche, da consultare per ottenere le entitÃă piÃź pertinenti alla vita di un individuo; proprio per decidere cosa ÃÍ attinente e cosa no, un algoritmo deve essere pensato e implementato; affinchÃÍ il risultato dellâĂŹimplementazione sia scalabile e flessibile, deve essere scelto un meccanismo di accesso aperto e utilizzabile in piÃź frangenti; ultima, ma non meno importante, la visualizzazione dei dati raccolti e filtrati, che devono essere presentati in una forma user-friendly, con attenzione particolare agli utenti di una certa etÃă, che, rappresentando una fetta consistente dei possibili utilizzatori, richiedono una progettazione piÃź attenta.

Nella realizzazione di un sistema capace di rispondere allâĂŹutente con dati relativi al tempo e al luogo in cui egli ha vissuto alcuni momenti della sua vita, la prima difficoltÃă ÃÍ riscontrabile nella scelta riguardo quale meccanismo utilizzare per recuperare le entitÃă con cui costruire il contesto; per ottenere questo risultato le possibilitÃă si possono ricondurre a tre approcci principali:

Sfruttare i linked data per recuperare a runtime le informazioni necessarie: questo ÃÍ una scelta ambiziosa, che pone molta fiducia nel fatto di riuscire a scavare nel modo giusto tra i datasets per trovare abbastanza elementi da restituire, elementi che siano anche inerenti alla richiesta fatta. Utilizzare i linked data, collegandoci un altro dataset, formato da informazioni raccolte da noi: il problema che vogliamo risolvere riguarda la storia delle persone, e chi meglio delle altre persone puÃš aiutarci a raggiungere questo obiettivo? per questo un grosso aiuto nella ricerca di immagini, eventi e altro potrebbe venire da un opera di crowdsourcing che, immagazzinata sotto forma di entitÃă con un significato e, collegata ad altre sorgenti di dati, andrebbe a completare i nostri bisogni e forse quelli di altri utilizzatori del semantic web.

QuestâĂŹultima ÃÍ la proposta di soluzione piÃź interessante, che si tuffa nel futuro di una rete nella quale informazioni da sorgenti completamente diverse tra di loro vanno a completarsi e impreziosirsi a vicenda, ma che per la natura sperimentale del nostro lavoro va purtroppo considerata solo un possibile sviluppo futuro . Per poter essere meno dipendenti dalle specifiche RDF e SPARQL, certamente ancora in evoluzione, lâĂŹapproccio che abbiamo scelto ÃÍ di per sÃl' il piÃź semplice: perchÃÍ non costruire un archivio storico in forma di database, indicizzarlo in maniera corretta e renderlo disponibile tramite un algoritmo di creazione di un contesto intorno a delle coordinate spazio-temporali? CosÃň nasce Reminiscens, il cui funzionamento e struttura sono spiegati qui sotto.

C.1 Architettura generale

Reminiscens ÃÍ un sistema strutturato di varie parti, che lavorano insieme per fornire numerosi servizi. Le funzionalitÃă di raccolta e memorizzazione dati sono svolte da due database, dei moduli software e unâĂŹinterfaccia grafica: Il primo database ÃÍ la componente fondamentale di Reminiscens, contenente tutto il materiale utilizzato dal sistema per restituire un contesto (di cui si parlerÃă piÃź avanti) allâĂŹutente, mentre il secondo serve a memorizzare eventuali contesti personali (FUTURE WORK); perchÃÍ la Knowledge Base contenga dei dati, questi devono essere raccolti da qualche parte, nel nostro caso il lavoro ÃÍ fatto da dei moduli ETL, ognuno dei quali si occupa di interrogare una determinata risorsa sul web (e.g. DBpedia); la stessa funzione ÃÍ svolta da CrowdMemories, una UI sviluppata da Francesco Maturi che utilizza il crowdsourcing per reperire materiale storico direttamente dagli utenti. LâĂŹaccesso ai dati invece, ÃÍ implementato tramite delle API con architettura REST, che svolgono quindi la funzione di intermediari nella comunicazione tra i client e la Knowledge Base. La parte di visualizzazione consiste attualmente in una semplice interfaccia in forma di libro sfogliabile, a cui nel futuro si affiancheranno altri servizi; questi ulteriori sviluppi verranno trattati nella SEZIONE CONCLUSIONI

C.2 Il mio lavoro

Durante il mio lavoro, mi sono trovato a seguire i dati per tutto il percorso, dalla loro raccolta alla loro visualizzazione, passando per tutta la fase di elaborazione. Di seguito ecco le parti che ho sviluppato:

Moduli ETL (Extract, Transform, Load): LâĂŹidea ÃÍ quella di un insieme di moduli standalone che raccolgono ed elaborano dati, dipendenti quasi solo da un modulo che si occupa della comunicazione diretta con la Knowledge Base. La loro importanza ÃÍ vitale perchÃÍ, oltre allâĂŹovvia funzione di recuperare gli elementi che andranno poi restituiti allâĂŹutente in forma di contesto, il hanno anche il compito di ricostruire dove possibile le coordinate spazio-temporali dei dati estratti (e.g. tramite il Geocoding di una locazione scritta in maniera testuale). Essendo la lista delle risorse consultabili sul web sempre in crescendo, questa scelta ÃÍ stata fatta per rendere semplice lâĂŹaggiunta e la sostituzione dei moduli, che attualmente supportano la consultazione di Flickr, DBpedia, Catinabib (catalogo di cartoline e incisioni a tema regionale di proprietÃă della biblioteca comunale di Trento) e di dataset del progetto OpenData Trentino. Proprio grazie a questâĂŹultima risorsa, si spera che con il passare del tempo, grazie allâĂŹaggiunta di nuovi datasets, le informazioni contenute nella Knowledge Base e di conseguenza fornite allâĂŹutente possano crescere di quantitÃă e di qualitÃă.

Indice: Per permettere una lettura veloce dei dati raccolti, il risultato delle estrazioni dal web viene periodicamente indicizzato, cosÃň che non sia necessario leggere tutto il db per ottenere un risultato da spedire ai client visuali.

Api REST: Per permettere la consultazione della KB senza dover interrogarla direttamente, ho sviluppato queste api che, tramite messaggi HTTP, permettono di fare ricerche a partire da parametri spazio-temporali. Queste si dividono in due tipi: Nel caso sia necessario semplicemente leggere una lista di elementi dal KB, un insieme di chiamate si assicura che al richiedente torni la giusta lista di tutte le immagini o gli eventi o altro combacianti i parametri di input. Se invece si tratta della comunicazione con i client, altre chiamate vanno a leggere lâĂŹindice per restituire un contesto. Questo argomento ÃÍ descritto formalmente a [4.3] Per ques-

tioni piÃź tecniche, a [nonloso] ÃÍ disponibile la documentazione delle api.

CRUD: Affiancato ai moduli ETL e alle api ÃÍ presente un piccolo e semplice sistema con interfaccia minima che effettua CRUD, per aggiungere e modificare manualmente il materiale del db.

Booklet: UnâĂŹultima parte riguarda il lavoro per comporre unâĂŹinterfaccia che potesse essere utile a visualizzare i dati restituiti dalle API, indipendente dai client di Reminiscens: chiedendo allâĂŹutente una decade e un luogo, quello che viene mostrato ÃÍ un libricino sfogliabile tramite browser, e contenente un contesto completo in una forma che possa essere familiare a chi lo guarda, soprattutto nel caso dei meno giovani, che mancano di dimestichezza con i mezzi piÃź tecnologici. Il libro restituito ÃÍ perÃš un insieme di dati impersonali, in quanto per avere un risultato personalizzato la scelta giusta ÃÍ il vero e proprio client di Reminiscens.

C.3 La costruzione di un contesto

Come definito sopra, il contesto ÃÍ un insieme di immagini, eventi, personaggi e musica che hanno a che fare con il tempo e il luogo di cui lâĂŹutente potrebbe voler scoprire qualcosa, e che potrebbe aiutarlo a raccontare della sua vita. La nostra scelta ÃÍ stata quella di comporre questo contesto utilizzando cinque entitÃă per tipo, selezionate secondo alcuni criteri allâĂŹinterno dellâĂŹindice. Partendo dalla knowledge base, la Lifeincontext API crea un contesto intorno ad una lista di eventi. LâĂŹelaborazione si divide due casi principali, allâĂŹinterno dei quali si snodano diversi sottocasi: se le coppie lat-lon possono essere racchiuse allâĂŹinterno di un cerchio virtuale di raggio 50 km: se la maggior parte delle decadi ÃÍ uguale, viene creato un unico contesto a partire da una delle coppie lat-lon e da quella decade. questo si puÃš ottenere con una singola query che, ordinando i risultati per pertinenza, seleziona direttamente i primi cinque. se il numero degli eventi di input ÃÍ minore di tre, viene creato un contesto componendo dei âĂIJsotto-contestiâĂİ, uno per ogni coppia formata dalle decadi di input e da una delle coppie lat-lon. lâĂŹultimo caso ÃÍ unâĂŹestensione del secondo, infatti, a partire da una lunga lista di coppie, viene creato un contesto componendo tre sotto-contesti tra quelli possibili; questo viene fatto per non inquinare troppo il risultato, che senza il filtraggio appena indicato sarebbe una semplice accozzaglia di elementi che in ultima analisi sarebbero poco correlati tra di loro. se le coppie non soddisfano la condizione, vengono riproposti il secondo e il terzo sotto-caso indicati sopra, con la differenza che ogni contesto viene computato a partire dalle coppie cosÃň come indicate in input, senza elaborazioni addizionali. Oltre al calcolo a partire dalle sole indicazioni spazio-temporali, un ulteriore versione dellâĂŹalgoritmo ÃÍ stata pensata per cercare, filtrare e personalizzare i risultati ottenuti anche in base a delle parole chiave, siano esse argomenti, generi musicali, interessi dellâĂŹutente e oggetto della risorsa: ad esempio, una richiesta riguardante Trento nel decennio 1980, con keyword âĂIJcalcioâĂİ, dovrebbe restituire tra le altre una voce sul campionato di calcio serie C2 1984-85, quando la squadra locale venne promossa in Serie C1 dopo uno spareggio vinto contro l'Ospitaletto. Grazie a queste poche informazioni, fornite al sistema dallâĂŹutente in fase di configurazione del profilo, potrebbero consegnargli del materiale interessante ed evocativo di momenti del suo passato, stimolando il racconto di episodi della sua vita.

C.4 Framework e librerie

Per realizzare queste componenti, mi sono avvalso di diverse tecnologie, elencate qui sotto per sezione. Moduli ETL e Indice: Tutti i moduli sono realizzati in Java utilizzando Apache HttpComponents per gli scambi di messaggi HTTP, Hibernate, Apache Jena per poter interrogare tramite SPARQL, le API di Google Maps per risolvere tutti i problemi relativi alla risoluzione sia di coordinate in luoghi che il contrario, Google Gson per il parsing di messaggi JSON e le API di Youtube per associare un video musicale a ogni canzone nella KB. Api REST: la scrittura di queste librerie mi ÃÍ stata facilitata dallâĂŹutilizzo di Ruby unito al framework Sinatra, con lâĂŹausilio di ActiveRecord e del linguaggio di templating RABL. Al fine di eseguire alcuni calcoli complessi, ho eseguito il porting di uno script per la conversione di coordinate geografiche in UTM (http://home.hiwaay.net/ taylorc/toolbox/geography/geoutm.html) e ho adattato un implementazione Ruby dellâĂŹalgoritmo lineare di Megiddo per il calcolo del Minimum Enclosing Circle di un set di punti (http://www.dseifert.net/code/me
CRUD: Questa ÃÍ unâĂŹapplicazione Ruby on Rails che si avvale di una semplice UI per la modifica manuale dei dati, ma che permette eseguire le stesse operazioni programmaticamente, inviando e ricevendo JSON. Booklet: LâĂŹinterfaccia del booklet si basa su âĂIJ20 things I learned about browsers and the webâĂİ (http://www.20thingsilearned.com/), applicazione web HTML5 resa open source e riadattata da me per le nostre necessitÃă.