SOI 2020- DataByter

DataByter è una piattaforma per la gestione di dataset per Machine Learning, sviluppata per il progetto di Sistemi Orientati ad Internet del A.A. 2020/2021. Di seguito la discussione e relativa implementazione dei requisiti funzionali del progetto.

Requisiti funzionali

Accesso

I punti 1 e 2 riguardano l'accesso al sito e la possibilità di registrarsi. Per accedere al sito è richiesto un nome utente e una password. È possibile anche registrarsi, inserendo una mail, uno username e una password. Non è possibile registrarsi utilizzando una mail o uno username già registrato. E però possibile aggiornare la propria password. Quest'ultima operazione richiede però che l'utente sia a conoscenza sia del proprio username e e-mail inseriti durante la procedura di registrazione. In caso contrario non sarà possibile aggiornare la propria password. Sia nella fase di registrazione che di aggiornamento password è necessario che l'utente inserisca due volte la stessa password per evitare possibili errori di battitura. Tutti i controlli vengono effettuati lato server. Di fatto il login è solo simbolico: è possibile infatti accedere al sito puntando direttamente alla pagina principale. I dati di accesso degli utenti vengono salvati in chiaro all'interno del database. Questo rappresenta un problema per quanto riguarda la sicurezza, che potrebbe essere risolto tramite l'utilizzo di algoritmi di criptazione. Un altro punto critico non gestito è il salvataggio dello username all'interno del sessionStorage. Un utente malevolo infatti potrebbe modificare il valore salvato e agire come un altro utente

Progetto

I punti dal 3 al 8 riguardano la struttura dei progetti. L'utente può creare un nuovo progetto inserendo un nome, una descrizione e un "obiettivo" di dimensione del dataset. Quest'ultimo indica quale dovrebbe essere la dimensione minima del dataset per essere ritenuto adatto all'addestramento di un modello. In ogni caso questo è solo un valore indicativo e non produce nessuna limitazione al progetto. Si possono selezionare due tipi di progetto: Immagine o Testo. I progetti Immagine prevedono l'inserimento di due o più campi (fields) di cui uno dovrà essere selezionato come il campo dei labels, cioè il campo su cui i modelli dovranno addestrarsi. Tutti i campi a parte quello dei labels sono campi Immagine, potranno quindi contenere solo immagini. I progetti Testo invece prevedono anche la possibilità da parte dell'utente di definire un tipo per il campo: Testo, Numerico, Binario, Data. In questo modo gli utenti che vorranno aggiungere o modificare delle entry del progetto dovranno attenersi a una struttura più rigida, permettendo una più facile manutenibilità del dataset e un processo di Data Cleaning più rapido. In entrambi i tipi di progetto occorre inoltre inserire tutti i possibili valori del campo Label. In questo modo è il

creatore del progetto a gestirne la struttura e a decidere se il dataset potrà essere multi-label o meno. Una volta creato il progetto non può essere modificato ma solo eliminato.

Entry

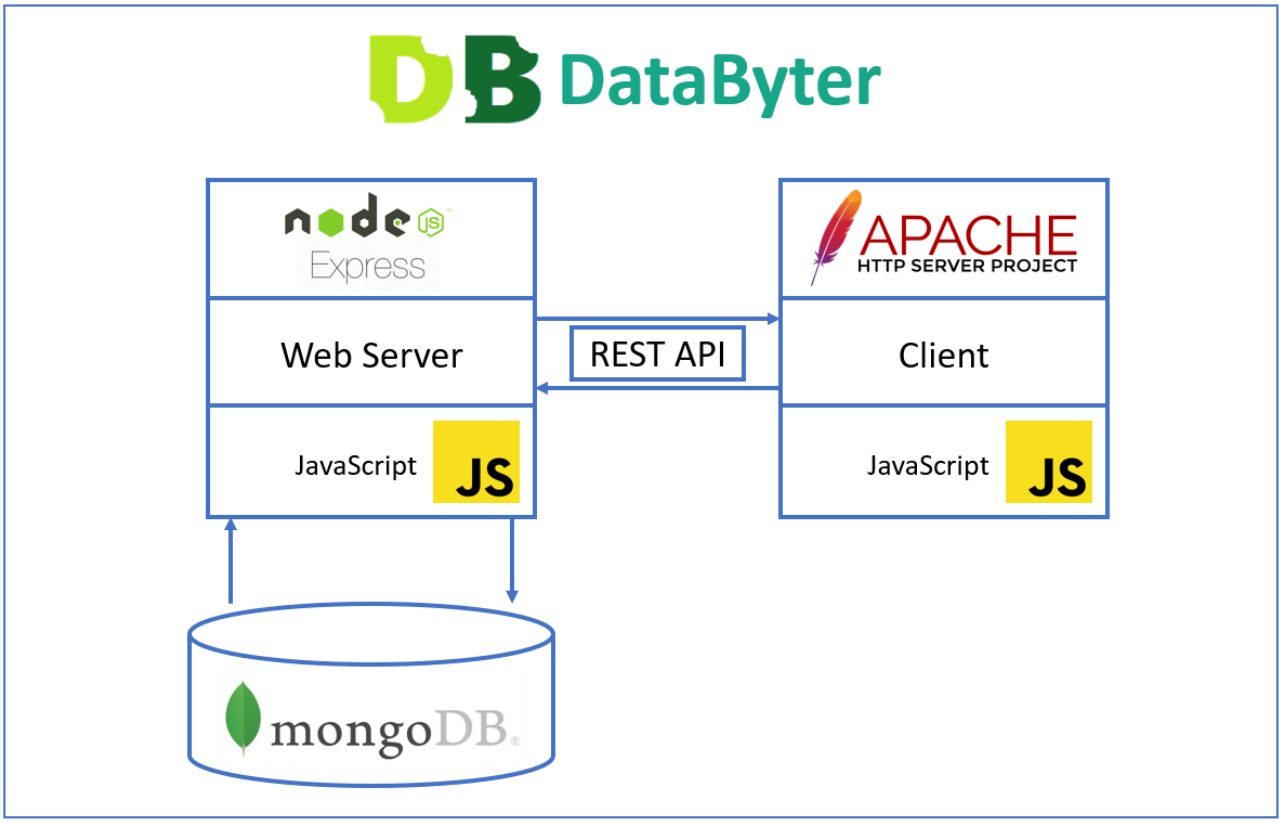
I punti 9, 10 e 11 riguardano la gestione delle entry degli utenti nei vari progetti. Ogni utente ha la possibilità di visualizzare o eliminare qualsiasi progetto, anche non creato da lui. La pagina del progetto contiene una tabella ordinata con i campi e i valori aggiunti dagli utenti, oltre che a dei grafici che mostrano lo stato del progetto rispetto alla propria soglia minima e a bilanciamento (balance) tra i label delle entry. È spesso preferibile infatti avere dei dataset il più possibile bilanciati per evitare episodi di overfitting durante l'addestramento. Ogni utente può anche aggiungere, modificare o eliminare un qualsiasi campo. La pagina di aggiunta di un nuovo campo contiene una form strutturata in base ai tipi di campi aggiunti durante la creazione del progetto. Nel caso di modifica di una entry la pagina è simile alla precedente con la differenza che i campi di input vengono popolati di default con il valore salvato nella entry. L'utente può visualizzare anche lo storico di una specifica riga, vedendo ad esempio le modifiche che sono state fatte nel tempo da ciascun utente. Infine è possibile anche scaricare il dataset sotto forma di file CSV: nel caso di progetti Testo i dati sono quelli effettivi, mentre per i progetti Immagine in ogni cella viene salvata una stringa Base64 che rappresenta l'immagine salvata.

Requisiti tecnici

Database

Si è scelto di utilizzare MongoDB, uno dei più noti database non relazionali, per il salvataggio dei progetti e delle relative entry. La notevole libertà offerta da questo tipo di database ben si sposa con la natura dinamica di questi progetti: risulta infatti molto semplice la creazione di progetti con strutture diverse. Un altro aspetto molto utile è il fatto che le entry vengono salvate come document BSON, i quali presentano la stessa struttura di un oggetto JS. È quindi possibile eseguire delle query, ottenere il risultato e inviarlo direttamente al client senza aver bisogno di particolari modifiche alla struttura. Inoltre MongoDB, come molti altri database, offre la possibilità di eseguire query asincrone, permettendo una gestione più efficiente delle chiamate tramite l'utilizzo dei cursori.

Website architecture



Il sito web DataByter è formato da tre componenti:

- L' interfaccia Client, gestita tramite server Apache, tramite la quale l'utente è in grado di visualizzare, creare o eliminare i vari progetti. Per ogni progetto è inoltre possibile visualizzare, creare, modificare, eliminare o scaricare le relative entry. Viene anche data la possibilità di visualizzare lo storico di ogni entry del progetto.
- Il Web Server, sviluppato tramite Node.js ed Express, che gestisce la parte di dialogo con il Database e con il Client. Vengono riportate e spiegate le chiamate REST più importanti:
    - *(GET) projects,* recupera tutti i progetti esistenti in ordine di creazione.
    - *(POST) saveProject,* per il salvataggio di un nuovo progetto. Oltre ai dati inseriti dall'utente viene aggiunto il *projectId*, recuperato dalla collection *id-manager (*il counter viene subito incrementato nella stessa chiamata di recupero così da impedire la creazione di due progetti con lo stesso ID*)* e un counter *entryId*= 0 a cui faranno riferimento le future entry del progetto.
    - *(DELETE) project,* per l'eliminazione di un progetto. Richiede l'ID del progetto da eliminare. Viene eliminata la voce all'interno di *projects* e tutte le entry, presenti e passate, relative al progetto in *entries.*
    - *(GET) entries* recupera tutte le entry per un dato progetto. Richiede l'ID del progetto come parametro. Vengono recuperate da *entries* tutte le entry con il flag *isActive* = true, cioè tutte le ultime versioni di ciascuna entry.
    - *(POST) addEntry*, per il salvataggio di una nuova entry. Richiede l'ID del progetto. Oltre ai dati inseriti dall'utente viene aggiunto il *projectId* del progetto, l'*entryId*

recuperato dal counter relativo in *projects* (il quale verrà contemporaneamente incrementato). Infine viene aggiunto il flag *isActive* impostato a true.

- *(PUT) entry,* per la modifica di una entry esistente. Richiede l'ID del progetto e della entry da modificare. Il servizio di fatto crea una nuova voce in *entries* con i nuovi dati della entry e con gli ID passati come parametri. Inoltre viene aggiornata l'ultima versione dell'entry impostando il flag *isActive* a false.
- *(DELETE) entry,* per l'eliminazione di una entry. Richiede l'ID del progetto e della entry da eliminare. Vengono eliminate tutte le versioni dell'entry, anche quelle passate.
- *(GET) entryHistory,* per visualizzare lo storico di una entry. Richiede l'ID del progetto e della entry. Vengono restituite tutte le versioni dell'entry, ordinate in base alla versione più recente.
- *(GET) piechartData,* per recuperare le informazioni con cui popolare i grafici del progetto. Richiede l'ID del progetto. Le informazioni vengono calcolate dinamicamente tramite una pipeline di aggregazione.

- Il Database MongoDB, in cui sono salvati tutti i dati, nello specifico:
  - *users* contiene le utenze del sito.

```
{
    "_id" : ObjectId("5ff6c9356226c4122a441667"),
    "email" : "michael.petrolini@studenti.unipr.it",
    "username" : "MikaelRT",
    "password" : "admin"
}
```

1. Esempio di utenza

  - *id-manager* contiene un unico document con un ID al quale si fa riferimento durante la creazione di un nuovo progetto. In questo modo ogni progetto avrà un ID univoco.

```
{
    "_id" : ObjectId("5ff31521da339dce04f5cd22"),
    "type" : "project",
    "id" : 51.0
}
```

2. Counter relativo ai progetti

  - *projects* contiene tutti i dati relativi ai progetti. In particolare ogni document contiene, oltre alle caratteristiche particolari fornite dall'utente durante la fase di creazione, un *projectId* che lo identifica unicamente come progetto e un *entryId*, che viene incrementato ogni volta che viene creata una nuova entry per il progetto.

3. Esempio di progetto

  - *entries* contiene tutte le entry dei progetti. Una specifica entry viene identificata univocamente tramite il proprio *projectId* (ID del progetto) e *entryId* (ID dell'entry).

Il flag *isActive* specifica se quel document rappresenta la versione più recente dell'entry. Per una data entry in ogni momento può esistere un'unica versione attiva, mentre tutte le altre rappresentano lo storico delle modifiche per la specifica entry.

4. Esempio di entry

Sia la parte di front-end che back-end sono sviluppate, come da specifiche, in JavaScript. Per la parte di front-end non è stato utilizzato alcun framework JS o tema CSS. Le comunicazioni tra client e server avvengono tramite chiamate RESTful, mentre per la gestione del database viene utilizzato il driver ufficiale MongoDB Driver.

Di seguito viene riportata la struttura delle pagine del sito: