Esame di Stato di istuzione secondaria superiore

a.s. 2019-2020

Elaborato

Discipline di indirizzo

Planeet

Autore
Pietro Cappelletti

Indirizzo: ITIA - INFORMATICA E TELECOMUNICAZIONI
ARTICOLAZIONE INFORMATICA

Contents

Listings

1 Descrizione

Abstract

Descrizione generale del progetto, delle sue funzionalità e del suo impiego, reale o ipotetico, includendo eventuali ipotesi aggiuntive ritenute necessarie al funzionamento e svolgimento dell'elaborato.

1.1 Generale

Planeet è un progetto nato durante il periodo di stage 2019-2020 grazie alle prime conoscenze pratiche acquisite rispetto alla creazione di **applicazioni server**.

1.2 Concept

L'idea di partenza è stata quella di realizzare una suite completa di servizi che potesse offrire all'utente la possibilità di organizzare la propria agenda, sia lavorativa che privata, in modo facile, intuitivo e riservato agevolando la comunicazione, lo scambio di dati e la condivisione di eventi tra conoscenti, colleghi e amici.

Le funzionalità attualmente disponibili sono un servizio di messaggistica e di condivisione post tra followers.

La funzione calendario è in sé pronta in un altro progetto iniziato anch'esso a stage su commissione dell'azienda, manca solo la sua implementazione, oltre a queste il piano di sviluppo prevede anche l'introduzione di un sistema per lo sharing di documenti basato su un algoritmo di version control.

1.3 Design

I servizi offerti dalla piattaforma, come ad esempio la messaggistica, condivisione post e quelli che verranno aggiunti nel seguito sono rappresentati da un pianeta, che insieme vanno a comporre l'universo di **Planeet**. Il nome della piattaforma stessa è stato ottenuto unendo le parole planet e plan it, mettendo in evidenza come questa suite simboleggi un sistema organizzativo a se stante.

2 Tecnologie

Abstract

Descrizione delle tecnologie utilizzate, analizzandone gli aspetti teorici di base e la loro ricaduta nel contesto del progetto.

2.1 Node.js

`Node.js` viene utilizzato come framework per la creazione di servizi back-end, chiamati API, che rendono possibile l'utilizzo e il salvataggio dei dati all'interno del database. Questo servizio utilizza il linguaggio `javascript`, che lo rende fruibile ad un grande bacino d'utenza, per gestire le chiamate rendendo l'intero sistema più consistente e pulito dato l'utilizzo sia a livello server che a lato client. Questo runtime permette di lavorare in modalità asincrona, aumentando le prestazioni rispetto ad una modalità sincrona potendo gestire contemporaneamente più richieste allo stesso momento.Questo servizio è stato utilizzato per sostituire una vecchia applicazione web dell'azienda PayPal scritta inizialmente mediante tecnologia `spring`, basa su linguaggio `java`, ottenendo degli evidenti miglioramenti come:Diminuzione tempo di programmazione, della durata di circa due mesi, passando da un totale di cinque mesi nella prima applicazione ad un totale di tre mesi, pur utilizzando un numero minore di programmatori,passando da un numero di cinque ad un totale di due ingegneri. Diminuzione del 60% del numero di linee di codice utilizzate. Diminuzione del 40% del numero di file utilizzati. Velocizzazione dei tempi di risposta della stessa pagina fino al 35%.

2.2 Express

`Express` rappresenta un framework minimale e flessibile di `node.js` che da a disposizione l'utilizzo di robuste features per applicazioni mobile e web. Permette la creazione in modo facile di API tramite l'utilizzo di una miriade di metodi HTTP e middleware di sua disposizione. Le principali procedure utilizzate nel presente progetto derivano da HTTP e sono rispettivamente GET e POST.

Nel seguito è riportato un confronto tra le due funzioni.

	Get	**Post**
Cronologia	Parametri rimangono all'interno della cronologia essendo una parte dell'URL	Parametri non vengono salvati nella cronologia
Segnalibri	Possono essere aggiunti ai segnalibri	Non possono essere aggiunti ai segnalibri
Comportamento al reaload	Le richiesto sono eseguite nuovamente ma potrebbero non essere inviate al server nel caso in cui la pagina sia già stata salvata nella memoria cache	Il browser allerta l'utente che i dati verranno inoltrati nuovamente al server
Tipo di cifratura	*application/x-wwww-form-urlencoded*	*multipart/form-data* o *application/x-www-form-urlencoded* oltre all'uso della criptazione multipart per i dati binari
Parametri	I parametri possono essere passati nella stringa URL che presenta tuttavia un limite di lunghezza. Metodo più sicuro è quello di utilizzare meno di 2K parametri anche se alcuni server possono gestirne fino a 64K.	Invio di parametri illimitati al server incluso file
Violazione	Facile da violare	Più difficoltosa la violazione
Restrizioni tipo dati	Le richieste GET accettano soltanto caratteri di tipo ASCII	Non sono presenti restrizioni. Sono abilitati anche dati binari.
Sicurezza	GET meno sicura di POST a causa dell'invio di dati nella stringa URL e del loro salvataggio nella cronologia e nel log del server	POST leggermente più sicura rispetto a GET a causa del non salvataggio della trasmissione dei parametri nella cronologia e nei log del sistema
Restrizioni lunghezza dati	Esiste una restrizione sulla lunghezza della stringa URL variabile da browser a browser che influenza la lunghezza del passaggio di dati. è ritenuto sicuro utilizzare meno di 2048 caratteri	Non sono presenti restrizioni
Uso	Solitamente utilizzata per il recupero di documenti, come ad esempio la banare richiesta get '\' per ottenere la pagina iniziale di un sito web	Usato per l'inoltro e il salvataggio di dati da lato client a lato server
Visibilità	Richieste GET visibile a causa del loro salvataggoi nella stringa URL	Non vengono visualizzate nella stringa URL rendendo più sicura la reamissione di dati
Cache	Richiesta salvate nella memoria cache	Richieste non salvate nella memoria cache

2.3 Hash

Le funzioni hash permetto di creare, partendo da una stringa di lunghezza arbitraria, una stringa di bit con alcune proprietà.

Le proprietà che contraddistinguono un hash sono le seguenti:

- Facile calcolare l'hash dato un valore;
- È praticamente impossibile ritrovare la stringa iniziale dal suo valore di hash. (può diventare possibile ritrovare la stringa iniziale tramite "dizionari" online che salvano il valore di hash accanto alla stringa iniziale);
- Mappatura dei dati di lunghezza arbitraria in una stringa binaria di dimensione fissa chiamata valore di hash (message digest);

- La modifica, anche di un solo bit, della stringa iniziale comporta un drastico cambiamento del proprio hash;

Nell'ambiente di Planeet viene utilizzata una funzione hash chiamata SHA-256 per il salvataggio delle password all'interno del database favorendo in questo modo la privacy dell'utente.

2.4 MongoDB

`MongoDB` rappresenta un DBMS per la creazione di database orientati a documenti, quindi composti da oggetti salvati in formato di tipo `JSON`, rendendone più semplice la comprensione all'utente rispetto ad un modello tradizionale a righe e colonne.

```
{
"_id": "5cf0029caff5056591b0ce7d",
"firstname": "Jane",
"lastname": "Wu",
"address": {
    "street": "1 Circle Rd",
    "city": "Los Angeles",
    "state": "CA",
    "zip": "90404"
    },
"hobbies": ["surfing", "coding"]
}

```

Listing 1: Rappresentazione dati MongoDB in formato JSON

Si permette di comporre query ricche e comprensibili che favoriscono il monitoraggio e l'ordinamento di ogni attributo. Le stesse query sono composte in linguaggio `JSON` lasciandosi così alle spalle il concatenamento di stringhe prodotto dalla normale query SQL.

2.4.1 Normalizzazione

Con normalizzazione viene identificato il procedimento utilizzato dai progettisti di database per diminuire la ridondanza di dati all'interno del database, garantendo di conseguenza una riduzione di spazio utilizzato.

Esistono tre diverse forme di normalizzazione:

- **1° Forma normale** in cui ogni attributo presente nel modello viene trasformato, se necessario, in un tipo semplice, la forma in cui si devono trovare tutte le relazioni secondo il modello relazionale classico;
- **2° Forma normale** è invece la forma in cui alla relazione, già portata precedentemente in prima forma normale, vengono rimosse tutte le dipendenze parziali della chiave;
- **3° Forma normale** è rappresentata dalla relazione già in seconda forma normale a cui sono state tolte tutte le dipendenze transitive dalla chiave;

2.4.2 Denormalizzazione

La denormalizzazione invece è il processo che permette di ottimizzare le performance in lettura di un database aggiungendo ridondanza nei dati che permette di poter ottenere le informazioni necessarie senza l'utilizzo di un numero elevato di join, il che provocherebbe un accesso intenso al disco. `MongoDB` permette di utilizzare questa funzionalità nei suoi servizi.

3 Schemi progettuali

Abstract

Rappresentazioni grafiche dell'infrastruttura tecnologica ed informatica nel suo complesso (architettura di rete, progetto della base di dati, progettazione delle pagine web).

3.1 Progetto della base di dati

Durante l'elaborazione dell'idea del progetto ho deciso di dividere il carico di dati su database diversi per migliorarne l'ordine ed aumentare la sicurezza. Ora sono presenti due database separati incaricati rispettivamente del servizio di condivisione post e di messaggistica.

Il database riservato a dati rappresentanti il servizio social contiene informazioni su utenti, posts e followers nelle rispettive collezioni:

`USERS(_id, username, mail, password, securityQuestion, securityAnswere, validation, token, endToken)`

`USERINFO(_id, id, name, surname, birthday, sentimental, musics, films, bio)`

`POST(_id, idUser, text,url, date)`

`FOLLOWERS(_id, username, account)`

Il database riservato invece alla funzionalità di messaggistica contiene di default una collezione chiamata allchat, strutturata in questo modo:

`ALLCHATS(_id, user0, user1, date, read:%idUser%, read;%idUser%)`

contenente la lista della totalità delle conversazioni utilizzata per la rilevazione di notifiche, e l'insieme di tutte le singole collezioni rappresentanti ogni singola chat, denominate come `%idUser%` | `%idUser%` nel quale la stringa `%idUser%` viene sostituita con il vero identificatore dell'utente. Questi identificatori vengono ordinati alfabeticamente.

`%idUser%` | `%idUser%(_id, form, to, text, url, readOther, date)`

Alcuni attributi non sono temporaneamente utilizzati nella loro completezza ma sono inseriti per usi futuri, come ad esempio il campo url che andrà ad indicare un array di immagini che un utente potrà inviare e il campo `_id` che, descrivendo univocamente ogni messaggio, verrà utilizzato per permettere all'utente di contrassegnare qualche messaggio come messaggio importante o simile.

4 Codice

Abstract

Segmenti significativi del software prodotto.

Per sintetizzare nel modo migliore il codice ho scelto di inserire alcune delle API più significative all'interno del progetto.

4.1 API

4.1.1 Cambio nome utente

/db/checkUpdateUsername permette di cambiare il proprio username all'interno del social e può fornire come risposta:

- `okChange` nel caso in cui la modifica del nome sia andata a buon fine;
- `error errFindNew` nel caso in cui il nuovo nome utente sia già in uso e quindi non disponibile;
- `error Update` nel caso in cui si fosse verificato un malfunzionamento od un errore durante il cambiamento del nome successivamente alla verifica della sua disponibilità;
- `error fsRename` nel caso in cui si fosse verificato un malfunzionamento del sistema dovuto dalla modifica dei file in cui sono state salvate tutte le immagini caricate dall'utente.

```
1  app.post('/db/checkUpdateUsername',function(req, res) {
2      MongoClient.connect(url,function(errConnection, db) {
3          if(errConnection) res.send('error Connection');//throw err;
4          vardbo = db.db(databaseSocial);
5          o_id =newObjectId(req.body.idUser);
6
7          varoldValue = { _id: o_id };
8          varquerySearch = { username: req.body.newUsername };
9          varnewvalues = { $set: { username: req.body.newUsername } };
10
11         console.log(req.body.idUser +'-->'+ req.body.newUsername);
```

```
        dbo.collection(collectionUtenti).findOne(querySearch,function(errFindNew, result) {
            if(errFindNew) res.send('error errFindNew');//throw errFindNew;
            if(result !=null) {
                console.log('NoChange');
                res.send('NoChange');
                db.close();
            } else {
                dbo.collection(collectionUtenti).updateOne(oldValue, newvalues,function(errUpdate, res2) {
                    if(errUpdate) res.send('error Update');//throw errUpdate;
                    console.log('OkChange');
                    res.send('OkChange');
                    fs.rename(
                        'assets/uploadedFiles/'+ req.body.username,
                        'assets/uploadedFiles/'+ req.body.newUsername,
                        function(errChangeName) {
                            if(errChangeName) res.send('error fsRename');
                            }
                        );
                        db.close();
                    });
                }
            });
        });
    });

```

Listing 2: Cambio nome utente

4.1.2 Caricamento immagini

/db/uploadImage permette di caricare file in una cartella raggiungibile dal server rendendoli successivamente visibili all'utenza che ne richiede l'accesso. Successivamente a questa chiamata verrà utilizzata l'API **/db/saveImage** permettendo così di spostare il file nella cartella specifica dell'utente.

```
app.post('/db/uploadImage', multipartMiddleware,function(req, res) {
    filename = req.files.myFile.path.split('\\');
    filename = filename[filename.length - 1];
    console.log('uploaded:'+ filename);
    fs.copyFile(req.files.myFile.path,'./assets/uploadedFiles/Smistare/' + filename, (errCopy)=>{
        if(errCopy) res.send('error copy');// throw errCopy;
        res.send('./assets/uploadedFiles/Smistare/'+ filename);
    });
});

```

Listing 3: Caricamento immaigini

4.1.3 Post persone seguite

/db/FollowerPostspermette di ricevere, passando come attributi un array contenente la lista degli id degli account seguiti, il numero di post massimi che si possono caricare nella pagina ed un numero progressivo definito dalla posizione di scroll dell'utente, una lista di loro post.

```
app.post('/db/FollowerPosts',function(req, res) {
    MongoClient.connect(url,function(errConnection, db) {
        if(errConnection) res.send('error Connection'); //throw errConnection;
        var dbo = db.db(databaseSocial);
        var offset;
        var limite;
```

```
            var initialPost = Math.round(req.body.numberPagePost / 2 * 3);

            if(req.body.number == 0) {
                offset = 0;
                limite = parseInt(initialPost);
            } else {
                offset = initialPost + req.body.numberPagePost * (req.body.number - 1);
                limite = parseInt(req.body.numberPagePost);
            }

            dbo
                .collection(posts)
                .find({ idUser: { $in: req.body.array } })
                .sort({ date: -1 })
                .limit(limite)
                .skip(offset)
                .toArray(function(err, resultFind) {
                    res.send(resultFind);
                });
        });
    });

```

Listing 4: Post persone seguite

5 Riflessioni

Abstract

Riflessioni personali sul percorso svolto, sullo stato di avanzamento del progetto e sugli eventuali sviluppi futuri, sulle difficoltà incontrate e sulla loro risoluzione o meno.

Il progetto ha stimolato molto il mio interesse in quanto mi ha permesso, da un lato, di mettere in praticala teoria appresa a lezione mentre, dall'altro, di approfondire e/o apprendere nuove nozioni che durante lo svolgimento dell'anno scolastico non sono state affrontate.

L'attività svolta durante lo stage si è rivelata molto utile in quanto mi ha dato uno sguardo nuovo su come aziende reali impostano lavori di sviluppo (e programmazione) di progetti simili a quello presentato in questo documento, permettendomi di approcciare l'elaborato in modo più sistematico e sicuro.

L'intero processo di sviluppo mi ha portato a comprendere nuovi meccanismi per la creazione di API e per la gestione di dati all'interno di un database. La maggiore difficoltà incontrata riguarda il metodo di programmazione asincrona implementato, in quanto tutta la preparazione precedente riguardava solamente sistemi sincroni. Successivamente alla scoperta della soluzione di callback per superare questa problematica non ho riscontrato numerosi problemi nella creazione dei servizi, se non quelli dovuti alla mancanza di idee da implementare.

Alcune funzioni presentate in precedenza, come ad esempio l'integrazione di un calendario all'interno di Planeet, sono già in uno stadio avanzato di sviluppo ma non ancora integrate completamente all'interno dell'universo di Planeet. Il focus principale nel prossimo futuro sarà quindi quello di terminare tale implementazione al fine di raggiungere l'obiettivo prefissato all'inizio dello sviluppo del progetto.

Per quanto riguarda l'ideazione della funzione di recovery, questa è nata durante il mio primo approccio al caricamento file dovuto alla possibilità di utilizzare foto all'interno di post e come l'immagine profilo.