Exam.Net

Progetto Advanced Programming Languages

Sviluppatori: Francesco Cerruto 1000005927 - Giovanni Traina 1000053629

Linguaggi utilizzati

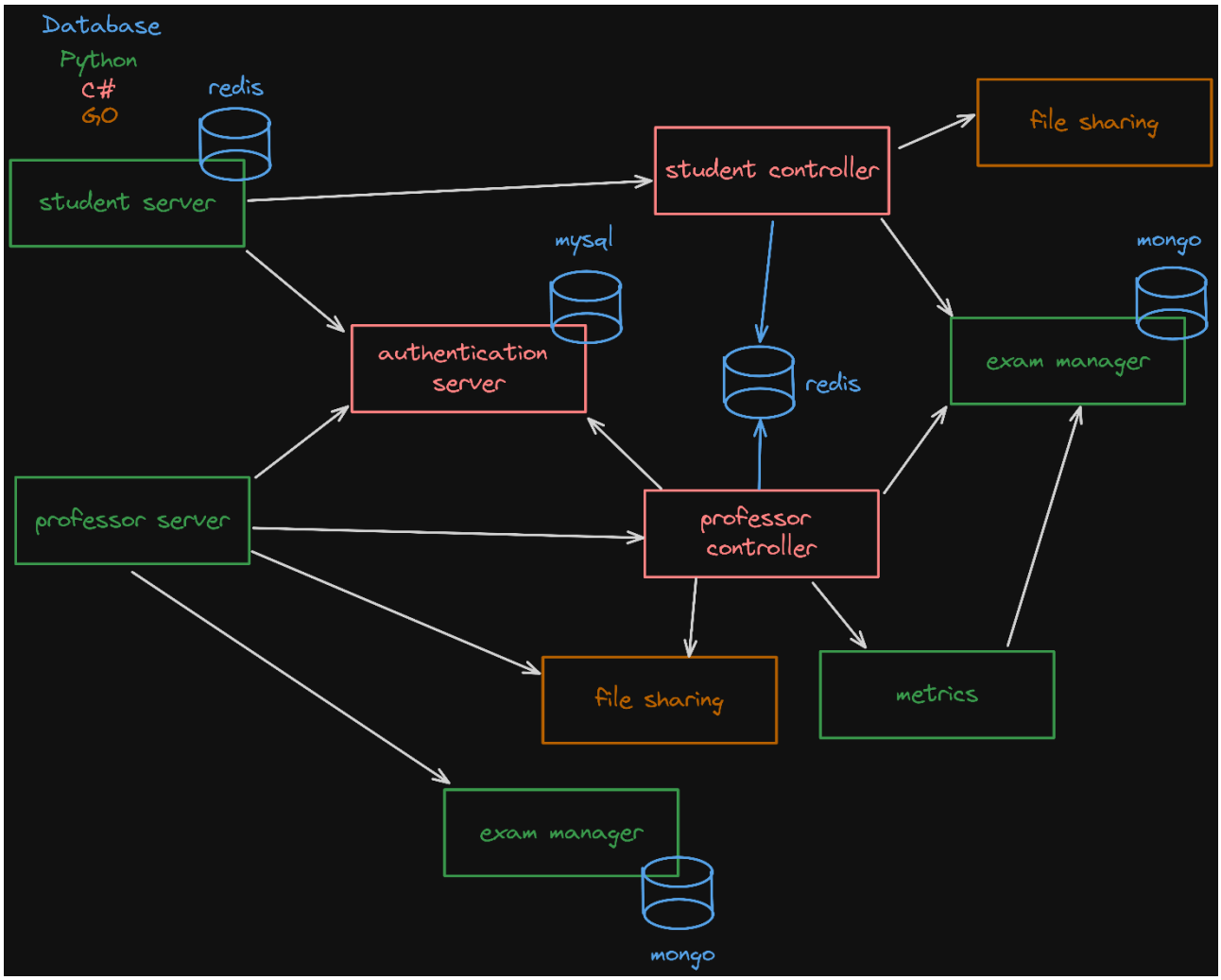
- Python
    - Student server – interfaccia client per lo studente
    - Professor server – interfaccia client per il professore
    - Exam manager – servizio backend
    - Metrics controller – servizio backend
- Go
    - File sharing – servizio backend
- C#
    - Student controller – servizio backend
    - Professor controller – servizio backend
    - Authentication server – servizio backend

Database utilizzati

- MySQL
    - Autenticazione utente
- MongoDb
    - Gestione compiti
- Redis
    - Memorizzazione temporanea compito ed informazioni correlate

NOTA BENE

- Si è scelto di far viaggiare tutti i dati utente all'interno di un request body (e quindi in un request method POST (indicato più alla modifica che al recupero di informazioni) anche nel caso in cui potesse essere più intuitivo utilizzare un request method GET per non farli viaggiare in chiaro all'interno dell'url
- Lo sviluppo del progetto è stato fatto contemporaneamente da entrambi



Moduli Python

Tutti i moduli sono stati scritti con il supporto del framework web Flask. Flask è progettato per semplificare lo sviluppo di applicazioni web in modo rapido e con un codice “boilerplate” minimo.

L’entrypoint dei vari servizi è il metodo main che istanzia l’applicativo Flask (oggetto della classe Flask) opportunamente configurato (viene richiesta in particolare una chiave di cifratura per i cookie scambiati con i browser e l’eventuale configurazione di ulteriori servizi esterni (nel caso dell’**exam manager viene configurato il collegamento per il database MongoDB**). Una volta istanziato l’applicativo questo viene mandato in esecuzione e messo in ascolto di tutto il traffico http proveniente da qualsiasi indirizzo ip e diretto verso la porta 5000.

Il routing delle richieste http avviene tramite decorator. Il decorator invocato in particolare è il metodo route dell’oggetto Flask che richiede in input l’URL della route servita ed il relativo tipo di richiesta (in particolare si è utilizzato GET e POST).

Professor server

Tramite questo client è possibile inizializzare il sistema. In particolare, previa autenticazione (**/register** e **/login**) (**ogni successiva route visitata richiede l’autenticazione, implementata tramite dati di sessione recuperati tramite apposito decorator user defined**) (**interazione con authentication server**), l’utente può abilitare un set di studenti ad utilizzare la piattaforma. Tale

indicazione viene fornita tramite l'upload ed il successivo parsing di un file **.csv** (**/add_student**) (**interazione con authentication server**). Si è scelto un file csv così da rendere più agevole la lettura del file (si procede con la lettura per righe e per ogni riga ogni campo viene riconosciuto tramite splitting sul carattere del '; '). Poiché la piattaforma può essere utilizzata da più professori, all'interno del file vengono riportate le informazioni di login del professore (**codice** e **materia**). Per accertarsi della correttezza delle informazioni il sistema elabora solo le righe del file che corrispondono all'utente che sta utilizzando il sistema (quindi stessa materia e stesso codice) e che siano formattate bene (si richiede la presenza di tre campi: codice studente, codice professore e materia). Sono stati anche previsti dei controlli per evitare l'inserimento di informazioni duplicate. Inoltre il professore, tramite lo stesso meccanismo, può caricare il set delle domande da utilizzare per la generazione dei compiti (**/add_question**) (**interazione con exam manager**). Controlli analoghi vengono effettuati su questo file su campi diversi, in particolare 8: (codice professore, materia, testo della domanda, 4 risposte separate, lista delle risposte corrette). Infine, il professore può stabilire il numero di domande per compito ed i punteggi da utilizzare in fase di correzione (si è previsto un punteggio per risposta corretta, sbagliata e incompleta) (**/exam_parameters**) (**interazione con exam manager**). Nota di rilievo va messa sul numero di domande presenti all'interno del compito. Nel caso in cui il professore dovesse richiedere più domande di quelle che ha a disposizione viene scelto il minimo tra i due.

In assenza delle informazioni sullo studente, questo non potrà accedere alla piattaforma. In assenza delle informazioni sul compito (domande da utilizzare e parametri esecutivi), lo studente sarà in grado di accedere alla piattaforma ma non verrà generato alcun compito.

Oltre l'inizializzazione del sistema è possibile recuperare le informazioni di utilizzo della piattaforma stessa. In particolare il professore è in grado di vedere delle statistiche generate sul momento dei compiti attualmente gestiti dalla propria piattaforma (**/metrics**) (**interazione con professor controller**). Le statistiche mostrate, nel dettaglio, sono il

- numero di compiti creati dalla piattaforma
- numero di compiti effettivamente consegnati dallo studente
- numero di compiti non consegnati
- numero di compiti superati
- numero di compiti non superati
- tempo di consegna puntuale e medio
- risultato ottenuto puntuale e medio

Le statistiche vengono generate a runtime tramite plot.

Oltre le metriche il professore può recuperare il file del compito sottoposto allo studente (**/downloads**) (**interazione con file sharing)** dove sono riportate le domande sottoposte (con rispettive risposte fornite e corrette) e le risposte indicate dallo studente. Alla fine del file (sempre in formato csv) è riportata la valutazione generata dal sistema

Infine il professore sarà in grado di resettare il sistema (**/restore**) (**interazione con professor controller**). In particolare vengono cancellati i file dei compiti dal server che li ospita ed i record degli studenti, delle domande, dei compiti e dei parametri dai rispettivi database.

Student server

Tramite questo client è possibile utilizzare il sistema come studente. In particolare, previa autenticazione (**/login**) (**interazione con authentication server**), l'utente sarà in grado di effettuare un esame. A differenza del client del professore, lo studente non ha la possibilità di poter navigare all'interno del server ma le pagine mostrate sono strettamente legate a ciò che in quel momento lo studente può fare.

In particolare, non appena effettua il login, viene direttamente generato il compito ed il sistema si mette in attesa di una sua azione (**/start_exam**) (**interazione con student controller**). In ogni caso il compito viene temporaneamente salvato su di un database Redis così da consentirne un eventuale successivo recupero. Nel caso in cui lo studente decida di cominciare, il sistema sottopone allo studente le domande del compito una per volta e memorizza temporaneamente sullo stesso database le risposte date ed il numero di domande già sottoposte (**/execute**) (così facendo si fa fronte ad un ipotetico guasto del server e l'utente riprende l'esecuzione del compito nel punto dove aveva precedentemente interrotto). Una volta risposto a tutte le domande, il compito viene salvato permanentemente su di un altro database, su cui lo student server non ha accesso, e viene salvato in un server per il successivo recupero da parte del docente (**interazione con student controller**).

Una volta consegnato il compito l'utente non sarà più in grado di utilizzare la piattaforma se non per il riepilogo del risultato (**/end_exams**).

Così come per il professore, anche per lo studente le informazioni di login vengono memorizzate sui dati di sessione e recuperati tramite apposito decorator user defined.

Modulo Exam manager

Tramite questo server è possibile la gestione dei compiti attraverso un database MongoDb. In particolare questo server consente la specifica delle domande da utilizzare per la generazione del compito (**/add_question**), la specifica dei parametri di esecuzione (**/add_parameters**), la creazione (**/create_exam**) e correzione (**/end_exam**) del compito.

Come politica di correzione si è scelto di assegnare in positivo il punteggio indicato dal professore per la domanda corretta se lo studente indica tutte le risposte correte alla domanda, in negativo il punteggio indicato dal professore per la domanda incompleta nel caso in cui lo studente dovesse inviare una domanda senza alcuna risposta ed il punteggio indicato dal professore per la domanda errata nel caso in cui lo studente indichi almeno una risposta errata.

Il meccanismo di correzione è reso possibile grazie al confronto degli indici di riposta selezionati dall'utente tramite l'interfaccia web e gli indici delle risposte corrette memorizzati nel database (le risposte vengono mostrate allo studente nello stesso ordine con cui sono memorizzate sul database)

Il server inoltre offre delle route di supporto per la generazione delle metriche (**/return_exams** che ritorna la lista di tutti gli esami memorizzati (completati e non) sulla piattaforma e **/return_parameters** che ritorna la lista dei vari punteggi indicati dal docente per consentire il calcolo delle metriche sull'esito dell'esame).

Altre route previste sono **/delete_exams** che cancella tutte le informazioni ospitate sul database e **/retrieve_questions** che ritorna la lista delle domande sottoposte allo studente (in particolare, oltre alle informazioni già ritornate allo studente precedentemente in fase di creazione, vengono restituite anche le risposte corrette).

A differenza dello student server e del professor server in questo caso non ho bisogno di dati di sessione in quanto questi vengono continuamente inviati all'interno del body delle request.

Modulo metrics controller

Tramite questo server è possibile la generazione a run time e su richiesta delle metriche. In particolare sono state predisposte delle route, ciascuna che calcola e restituisce, una sola metrica.

Come nel caso dell'exam manager non ho bisogno di dati di sessione in quanto questi vengono continuamente inviati all'interno del body delle request.

Modulo file sharing

Tramite questo server è possibile l'upload ed il download dei file dei compiti. In particolare, tramite la route **/upload** l'utente ha la possibilità di memorizzare permanentemente il file del compito mentre con la route **/file/path_to_required_file** è possibile scaricare il file. Piuttosto che inviare il file nel request body si è preferito inviare tutte le informazioni per creare il file (quindi domande e risposte). Infine è stata prevista la route **/delete_exams** per la cancellazione dei file dallo storage interno del server.

Poiché la piattaforma può essere utilizzata da più professori, i file dei compiti vengono memorizzati su subdirectory corrispondenti al codice e alla materia del professore così da consentire il recupero e la cancellazione dei soli file associati a quel professore.

I dati necessari per l'indexing del file (codice e materia del professore), così come tutti gli altri dati necessari per la creazione del file (nome dello studente, domande e risposte), vengono forniti nel body della request.

In fase di upload del file vengono scritte solamente le domande sottoposte allo studente. Il risultato viene successivamente scritto sullo stesso file dalla funzione associata alla route **/update**.

Nota di rilievo va messa sulle route separate di scrittura del file. Poiché la consegna del compito (che prevede memorizzazione e correzione e upload del file) viene fatta in parallelo tramite **delegates,** il task che si occupa dell'upload del compito non è in grado di sapere il risultato del compito. Per questo motivo si è scelto di procedere in parallelo con correzione e upload e successivamente, quando entrambe le operazioni sono finite, procedere con il recupero e la scrittura del risultato sul file.

Moduli in C#

I 3 moduli scritti in C# consistono in API Rest Controller con la versione .NET 8. L'entry point dei controller è il metodo Main (definito dentro la classe Program) che ha il compito di creare un host web e di eseguirlo. La creazione di un web host comporta la sua configurazione. Tra i parametri di configurazione abbiamo l'indirizzo ip e la porta di ascolto per il traffico http in arrivo, ed i vari servizi web che lo compongono (in particolare si inseriscono i controller definiti dall'applicativo).

L’inserimento di un controller comporta la mappatura di una route con una funzione definita all’interno dell’oggetto controller user defined

Nota di merito va messa sui professor e student controller. Mentre l’authentication server ospita solo le route di autenticazione degli utenti (gestite internamente con un database MySQL), le route di professor e student controller, tramite l’utilizzo dei delegates, consentono l’esecuzione in parallelo di più operazioni. In particolare per lo student controller sono stati predisposti due delegates: uno per la correzione del compito (**interazione con exam manager**) e uno per l’upload del compito (**interazione con file sharing**) (entrambi eseguiti sotto la route **/end_exam**. Per quanto riguarda il professor controller sono stati definiti 10 delegates: tre di questi si occupano del restore del sistema (uno è dedicato all’**interazione con l’authentication server**, uno per l’**exam manager** e l’altro per il **file sharing**), i restanti sono tutti dedicati per il recupero delle metriche (una metrica per ciascun delegate) (**interazione con metrics**). I due set di delegate vengono eseguiti sotto le route **/restore_system** e **/metrics** rispettivamente.

Sui controller è presente il meccanismo di lock e rilascio dello studente per riutilizzare la piattaforma. Nel dettaglio, quando lo studente richiede la generazione del compito, il suo nome viene memorizzato temporaneamente (quindi su Redis) su di un database condiviso tra il professor controller e lo student controller. Lo student controller, dal suo lato, se trova sul database il nome dello studente lo disabilita dall’utilizzo. Il professor controller, invece, controlla per ogni chiave memorizzata (non sa a priori chi ha usato il suo sistema) se l’utente ha lasciato la piattaforma (viene scritto in corrispondenza del nome la stringa “Non completato” se l’utente sta ancora eseguendo il compito, mentre il risultato della correzione se ha finito di usare la piattaforma). Nel caso in cui trovi che la piattaforma è stata correttamente rilasciata da tutti gli studenti procede con la restore.