Simple E-learning at Insubria

Obiettivo del progetto seatIn (Simple E-learning at Insubria) è la realizzazione di una semplice piattaforma di elearning.

L'applicazione dovrà essere utilizzata da utenti con ruoli distinti: 1) il docente, che gestisce la pubblicazione di materiale didattico e informazioni riguardante i corsi di cui è titolare, comunica con gli studenti dei suoi corsi attraverso newsletter, e analizza l'accesso alle pagine del corso; 2) lo studente, che una volta iscritto ad un corso può visualizzare tutte le informazioni pubblicate, scaricare il materiale didattico, ed inviare email al docente; 3) l'amministratore, che abilita un profilo utente (studente, docente o amministratore), crea un'istanza di un corso, assegna docenti ad un corso, e analizza l'uso della piattaforma.

La piattaforma seatIn consiste di: 1) un modulo denominato *seatInServer*, che interfacciandosi con un DBMS relazionale (PostgreSQL), fornisce servizi di back-end; 2) un modulo denominato *seatInUser*, che fornisce servizi designati per docenti e studenti; e 3) un modulo *seatInAdmin*, che fornisce servizi di gestione della piattaforma per utenti amministratori. La realizzazione dei servizi offerti richiede una opportuna suddivisione di funzionalità tra le componenti, e la definizione di opportuni protocolli di interazione tra queste parti. *seatIn* dovrà essere realizzata in modo tale da supportare l'interazione in parallelo con più utenti connessi alla piattaforma da postazioni differenti.

seatIn dovrà essere realizzata in Java utilizzando tecnologie che permettano un'implementazione efficiente dei servizi.

Requisiti funzionali

Avvio del server

All'invocazione di *seatInServer* deve essere richiesto di specificare: i) le credenziali per accedere a *dbSeatIn*, un database di supporto all'esecuzione dei servizi della piattaforma *seatIn*, e ii) l'host del database. Se *dbSeatIn* non contiene un profilo di utente amministratore, *seatInServer* deve avviare un processo di registrazione, richiedendo all'utente amministratore che ha invocato l'esecuzione: la matricola, il cognome, il nome, l'email istituzionale, la struttura di appartenenza e una password. Le credenziali registrate potranno essere utilizzate per effettuare l'autenticazione dell'utente amministratore tramite *seatInAdmin* e *seatInServer*. Al termine della registrazione, o nel caso in cui in *dbSeatIn* sia già presente un profilo di amministratore, *seatInServer* rimane in attesa di richieste di connessione da parte di client *seatInAdmin* e *seatInServer*.

Autenticazione

L'accesso ad ogni funzionalità offerta dalla piattaforma tramite *seatInUser* e *seatInAdmin* (ad eccezione del servizio di ripristino password dimenticata) richiede l'autenticazione dell'utente.
Un utente studente / docente / amministratore effettua il login mediante *seatInUser* / *seatInAdmin*, specificando come identificativo l'indirizzo di posta e la password associata al proprio profilo.
Se l'autenticazione fallisce l'utente non può accedere a nessun servizio di *seatIn* (l'unica eccezione è il servizio *Reset password* presentato in seguito). In seguito a 10 tentativi di autenticazione

consecutivi falliti, il profilo utente deve essere bloccato. Una volta bloccato, il profilo potrà essere riattivato unicamente da un utente amministratore.

Registrazione di corsi ed utenti

Un utente amministratore (autenticato) su indicazione delle segreterie didattiche, e tramite *seatInAdmin*, predefinisce una collezione di corsi gestiti dalla piattaforma. Un corso è caratterizzato da un codice identificativo, un nome, un anno accademico di attivazione, un corso di laurea di riferimento, e una breve descrizione dei contenuti.

Un utente amministratore tramite *seatInAdmin* effettua anche la registrazione degli utenti abilitati all'uso della piattaforma, predefinendone i profili utente.
In particolare, su indicazione della segreteria studenti, l'amministratore registra i dati degli studenti regolarmente iscritti ad un corso di laurea e i corsi che in base ai piani di studi risultano essere di competenza di ogni studente. L'amministratore può anche registrare docenti ed altri amministratori, e, sempre su indicazione della segreteria didattica dei dipartimenti che organizzano i corsi, l'amministratore attribuisce uno o più docenti (precedentemente registrati) ad un corso.

Quando un profilo utente viene aggiunto al sistema, indipendentemente dal tipo di utente considerato (studente, docente o amministratore), *seatIn* deve associare al profilo una password temporanea generata casualmente ed un codice di attivazione del profilo. Un profilo registrato deve risultare non attivo fino al primo utilizzo della piattaforma da parte dell'utente corrispondente che dovrà attivarlo. Per ogni profilo registrato, *seatIn* deve inoltrare all'indirizzo di posta elettronica istituzionale associato al profilo, la password e il codice di attivazione generati. L'utente sarà tenuto a resettare la password al primo utilizzo di *seatInUser*.

Per fini di praticità è bene prevedere che *seatInAdmin* possa caricare le informazioni di corsi ed utenti da file testuali in formato *csv*.

Attivazione del profilo utente

Se al momento dell'autenticazione il profilo dell'utente risultasse non attivo, il servizio di login dovrà effettuare l'attivazione del profilo registrato dall'amministratore. Tale funzionalità dovrà essere offerta sia da *seatInUser* che da *seatInAdmin* per i rispettivi utenti.

Al momento dell'autenticazione, l'utente deve specificare come identificativo l'indirizzo di posta associato al proprio profilo e la password temporanea inviata per email in fase di registrazione. Una volta accertata la validità delle credenziali, e identificato che si tratta del primo tentativo di accesso, il servizio di autenticazione deve richiedere all'utente di impostare una nuova password e di specificare il codice di attivazione ricevuto per email. Se il codice specificato è corretto, la password specificata dovrà essere associata al profilo utente e il profilo sarà infine attivato.

Reset di password

I client *seatInUser* e *seatInAdmin* devono fornire un servizio per reimpostare una password dimenticata. Il servizio dovrà richiedere all'utente di specificare l'email istituzionale associata al profilo, dovrà rimpiazzare la pwd del profilo (generata automaticamente o specificata dall'utente) con una nuova generata automaticamente e dovrà inoltrare la nuova password e un nuovo codice di attivazione all'indirizzo di posta elettronica associato al profilo.

Gestione degli utenti

Un amministratore, utilizzando *seatInAdmin*, può modificare i dati degli utenti precedentemente registrati (ad eccezione della password a cui non può accedere neppure in lettura), mentre uno studente o un docente possono visualizzare le informazioni riguardanti il proprio profilo personale, ma non hanno modo di modificarle.

Accesso ai contenuti

SeatInUser e *seatInAdmin* devono mostrare l'elenco dei corsi accessibili dall'utente autenticato, che nel caso di uno studente corrispondono ai corsi contenuti nel piano di studi, mentre, nel caso di un docente/amministratore, ai corsi da loro gestiti.

L'utente seleziona il corso di cui intende accedere ai contenuti. Se l'utente è uno studente/docente e non è iscritto al corso *seatInUser / seatInAdmin* può solamente visualizzare: il titolo del corso, i docenti assegnati e una breve descrizione del corso. Contestualmente a tale visualizzazione *seatIn* deve informare l'utente in questione dell'iscrizione necessaria per accedere ai contenuti e richiedere all'utente il consenso all'iscrizione. L'iscrizione effettuata dovrà essere notificata per email all'utente richiedente e ai docenti titolari del corso.

seatInUser e *seatInAdmin* devono permettere a tutti gli utenti regolarmente iscritti e agli amministratori di accedere ai servizi di visualizzazione dei contenuti di un corso selezionato, che permettano all'utente di navigare all'interno della gerarchia di sezioni e risorse che caratterizzano il corso e selezionare i contenuti di interesse. Per ogni sezione selezionata l'applicazione deve visualizzare titolo e descrizione, e deve permettere all'utente di effettuare un eventuale download delle risorse contenute. *(Opzionale) se la risorsa selezionata per il download è una cartella o se l'utente seleziona più di una risorsa, l'applicazione deve generare un archivio zip temporaneo contenente le risorse corrispondenti e avviare il download dell'archivio.*
NB. Non è richiesto che venga gestita la visualizzazione di file.

Pubblicazione dei contenuti

I privilegi di scrittura sui contenuti di un corso sono concessi unicamente ai docenti titolari e agli amministratori.

Un corso è caratterizzato da un codice identificativo, un nome, un anno accademico di attivazione, un corso di laurea di riferimento, una breve descrizione dei contenuti, ed è strutturato in sezioni. Una sezione è caratterizzata da un titolo, una descrizione (opzionale), da file (ad es. txt, ppt, pdf e jpg) eventualmente organizzati in cartelle, che rappresentano le risorse didattiche del corso e da un numero variabile di sotto-sezioni con la medesima strutturazione. Per semplicità assumiamo che le sotto-sezioni non possano contenere sotto-sotto sezioni e le risorse di tipo cartella possano contenere solo risorse di tipo file.

Ad esempio la sezione *Esercizi* potrebbe specificare come descrizione "Collezioni di esercizi assegnati", e mostrare una cartella contenente un numero variabile di file ognuno contenente uno o più esercizi assegnati.

Ogni sezione/risorsa ha associato un livello di visibilità che può essere *privato*, cioè visibile unicamente dal docente titolare, o *pubblico,* cioè accessibile da ogni utente (opzione di default), e *(richiesta opzionale) un intervallo temporale di visibilità, che specifica una restrizione di visibilità*

di un contenuto pubblico. Ad esempio, il docente potrebbe aggiungere la sezione pubblica *Soluzione* come sottosezione di *Esercizi*, caratterizzata da un numero variabile di file che forniscono le soluzioni degli esercizi proposti. Il docente potrebbe specificare l'intervallo temporale 2018-05-01T00:00:00 ..* per indicare che le soluzioni sono accessibili solo a partire dalla data specificata. Se per una sezione/risorsa non vengono specificati intervalli temporali di visibilità, si intende che localmente non è definita alcuna restrizione alla visibilità dell'elemento considerato.

La piattaforma *seatIn* specifica i seguenti criteri di composizione della visibilità dei contenuti: 1) una sezione/risorsa pubblica può contenere una sezione/risorsa privata, mentre una sezione/risorsa privata può solo contenere sezioni/risorse private, e *(richiesta opzionale) 2) una sezione/risorsa pubblica che specifica un intervallo di visibilità inv può solo contenere sezioni/risorse il cui intervallo di visibilità inv' non sia disgiunto da inv*.

Le informazioni sui corsi specificate dall'amministratore in fase di registrazione non possono essere modificate dal docente. Il docente può però aggiungere ed editare i contenuti del corso. In particolare il docente può:

1. aggiungere una sezione al corso
2. aggiungere una sottosezione a una sezione precedentemente definita
3. aggiungere una risorsa di tipo file / cartella ad una sezione/sottosezione precedentemente definita
4. aggiungere una risorsa di tipo file ad una risorsa di tipo cartella relativa ad una (sotto)sezione
5. modificare titolo o descrizione di una sezione/sottosezione
6. modificare la visibilità pubblica/privata di una sezione/sottosezione/risorsa e *(richiesta opzionale) l'intervallo di visibilità*. Se l'esecuzione della modifica richiesta viola i vincoli di visibilità, *seatIn* deve bloccare la richiesta, *(richiesta opzionale) oppure, se il conflitto può essere risolto per propagazione del criterio, su conferma dell'utente, seatIn propaga la modifica alle sezioni/risorse più interne che violano il vincolo, sovrascrivendo i criteri locali che generano il conflitto*.
   Ad esempio ipotizziamo che l'utente voglia attribuire visibilità privata alla sezione *Esercizi*. Verificato il conflitto, il criterio di propagazione porta a modificare la visibilità di tutte le risorse e sottosezioni contenute in *Esercizi*, sovrascrivendo i criteri di visibilità impostati localmente. Di conseguenza, sia la sottosezione *Soluzione* che le relative risorse saranno impostate con visibilità privata, rimuovendo anche il vincolo temporale impostato sulle risorse.
7. cancellare una sezione/risorsa. Se una sezione/risorsa contiene sottosezioni/risorse, la cancellazione viene propagata.
8. visualizzare i contenuti come se fosse uno studente iscritto

Comunicazione

seatInUser e *seatInAdmin* devono inoltre offrire servizi di newsletter. Un docente titolare / amministratore può decidere di notificare avvisi per email a tutti gli studenti iscritti al corso, o a una

selezione di studenti iscritti. Il servizio può anche essere usato contestualmente alla pubblicazione di nuovi contenuti del corso, per notificare a tutti i partecipanti un aggiornamento rilevante.

Deve essere offerta la possibilità all'utente di selezionare come oggetto del messaggio il titolo di una sezione del corso.

Monitoraggio

seatInUser/*seatInAdmin* deve offrire servizi di analisi per docenti titolari di un corso e amministratori. In particolare deve:

1. Mostrare il numero complessivo di utenti connessi che stanno visualizzando/interagendo con i contenuti del corso
2. Mostrare il numero complessivo di utenti che hanno effettuato il download di una o più risorse in intervalli temporali dati
3. Derivare il tempo medio di connessione degli studenti alle pagine del corso

seatInAdmin deve permettere agli amministratori di effettuare analisi di utilizzo. In particolare deve:

1. Mostrare il numero complessivo di utenti contemporaneamente connessi a *seatIn*
2. Mostrare il numero complessivo accessi per corso in una fascia temporale
3. Derivare il tempo medio di connessione degli studenti per ogni corso
4. Derivare il numero complessivo di download per ogni corso

Requisiti dati

seatInServer fornisce servizi di supporto alla memorizzazione ed analisi dei dati dell'intera piattaforma. E' richiesto di progettare la base di dati che *seatInServer* utilizzerà per gestire informazioni relative: agli studenti, ai docenti e agli amministratori, e ai corsi erogati. In particolare, sono di interesse le seguenti informazioni.

1. Il profilo degli studenti regolarmente iscritti ad un corso di laurea, caratterizzato dal numero di matricola, il cognome, il nome, l'email istituzionale, il corso di laurea, l'anno di immatricolazione, lo stato della carriera dello studente (codifica l'iscrizione al I / II / III anno in corso / fuori corso), il codice di attivazione e la password.
2. L'elenco degli esami di profitto che lo studente è tenuto a sostenere in base al piano di studi presentato. Lo studente può iscriversi solamente a corsi presenti nel piano di studi.
3. Il profilo di docenti e amministratori, caratterizzato dalla matricola, il cognome, il nome, l'email istituzionale, il dipartimento o la struttura di riferimento dell'utente, il codice di attivazione e la password.
4. Il catalogo dei corsi, dove ogni corso è caratterizzato da un codice identificativo, un nome, un anno accademico di attivazione, un corso di laurea di riferimento, una breve descrizione, e da contenuti strutturati in sezioni. Una sezione è caratterizzata da un titolo, una descrizione (opzionale), un numero variabile (0..*) di risorse, e un numero variabile(0..*) di sottosezioni ognuna con possibili risorse.

Una risorsa può modellare un file o una cartella, ed è caratterizzata da un nome e da una descrizione (opzionale). Se una risorsa modella una cartella, può contenere a sua volta un numero variabile (0..*) di file.

Svolgimento del progetto

L'obiettivo del progetto consiste nella 1) progettazione e 2) sviluppo di un sistema software e della relativa base di dati che soddisfino i requisiti proposti, 3) utilizzando un processo di sviluppo strutturato, e 4) nella produzione di prodotti intermedi, quali, i diagrammi UML e gli schemi ER, conformi alle metodologie seguite a lezione. Le attività di analisi e progettazione devono essere adeguatamente documentate facendo estesamente uso del linguaggio UML per l'applicazione client/server e del modello Entity-Relationship (ER) per il database.

È richiesto di progettare l'applicazione avvalendosi dove possibile dell'uso di design patterns, e di realizzare l'applicazione con un'opportuna interfaccia grafica, usando il linguaggio di programmazione Java, e progettare e realizzare un database utilizzando PostgreSQL per la sua implementazione (http://www.postgresql.org) e JDBC per l'accesso alla base di dati da programma Java (http://jdbc.postgresql.org/download.html)

È opportuno evidenziare che i servizi dell'applicazione vengono potenzialmente erogati in parallelo a più utenti che interagiscono con *seatInServer* per mezzo di *seatInUser* e *seatInAdmin*. In questo scenario, possono verificarsi accessi concorrenti a risorse condivise. L'applicazione deve quindi essere definita in modo tale da gestire opportunamente l'accesso concorrente alle risorse accedute.

Nota implementativa

Come descritto in vari casi d'uso, l'interazione tra *seatIn* e l'utente finale avviene anche per mezzo di email composte e inoltrate automaticamente. La realizzazione di tale funzionalità richiede l'utilizzo di servizi di inoltro di messaggi di posta elettronica forniti da un server SMTP. JavaMail (https://java.net/projects/javamail), è una libreria che supporta l'interfacciamento di applicazioni Java con vari servizi di gestione della posta elettronica, tra cui quelli forniti da server SMTP. E' pertanto richiesto di utilizzare JavaMail (https://maven.java.net/content/repositories/releases/com/sun/mail/javax.mail/1.5.6/javax.mail-1.5.6.jar) per supportare l'interazione tra seatIn e il server SMTP usato in ateneo (smtp.office365.com). In allegato alla presente documentazione viene fornito il codice sorgente di una classe che effettua la spedizione di una mail usando il server SMTP usato in ateneo. Per implementare la spedizione di email è suggerito di ispirarsi a tale codice, riutilizzandolo, ed eventualmente modificandolo. Il server SMTP considerato consente l'invio di email esclusivamente ad utenti autenticati. Le credenziali di autenticazione sono le stesse usate per il login al servizio di web mail dell'università (ad es. username: nome.cognome@studenti.uninsubria.it pwd: mypassword). La spedizione è autorizzata solo se le credenziali sono valide e il mittente del messaggio corrisponde allo username.

Realizzazione del Database

La realizzazione del database prevede svariate fasi. In seguito si riportano le attività ed i relativi artefatti che dovranno essere prodotti nelle varie fasi.

1. Si ristrutturino, se necessario, secondo le metodologie di progettazione i requisiti descritti. Si scelgano le metodologie per la costruzione dello schema ER, motivando le scelte fatte.
2. Si definisca lo schema concettuale ER per il database, evidenziando le entità e le associazioni di interesse, nonché i vincoli di cardinalità e di identificazione, motivando le scelte effettuate. Altri eventuali vincoli devono essere espressi in linguaggio naturale.
3. Si effettui la ristrutturazione dello schema ER motivando le scelte effettuate.

E' richiesto di produrre un documento di analisi dei requisiti ristrutturato e la documentazione associata allo schema ER (ristrutturato e non), con eventuale specifica di vincoli in linguaggio naturale.

Si effettui la traduzione dello schema ER ristrutturato in un equivalente schema relazionale.

E' richiesto di produrre la documentazione associata allo schema relazionale derivato dallo schema concettuale.

Si realizzi il database utilizzando PostgreSQL, e SQL per la definizione dei dati, l'implementazione dei vincoli identificati, e la manipolazione dei dati, secondo le operazioni previste dall'applicazione.

Documentare gli script SQL necessari alla creazione della base di dati e dei vincoli definiti sui dati e le query SQL a supporto dei servizi erogati da seatIn.

Codifica

Il codice sorgente dell'applicazione dovrà essere definito in modo da facilitarne la comprensione e la manutenibilità. Per favorire questo obiettivo vengono in seguito riportate alcune linee guida:

1. Utilizzare la lingua inglese per etichettare classi, metodi, attributi e variabili nel codice sorgente java. I commenti possono essere espressi sia in inglese che in italiano.
2. Ogni volta che si identificano due estratti di codice sufficientemente simili, il codice deve essere reingegnerizzato, e il codice duplicato eliminato portando a fattore comune le funzionalità condivise. Si consiglia l'utilizzo di strumenti di refactoring forniti da ambienti di sviluppo quali Eclipse / NetBeans.
3. E' opportuno notare che esistono delle "regole di stile" per ogni linguaggio di programmazione, quindi per indentazione, nomi, parentesi, commenti, etc. Esistono strumenti automatici quali Checkstyle che aiutano a verificare che il codice sorgente soddisfi standard di codifica (http://checkstyle.sourceforge.net/).

Documentazione del codice

Generare documentazione javadoc documentando i metodi e gli attributi con visibilità *public* o *protected* di ogni classe con visibilità *public, e* (*opzionale) per le classi con visibilità package e per i metodi e gli attributi con visibilità package o private*. La documentazione generata deve spiegare come usare una determinata classe o interfaccia, non come questa sia stata effettivamente implementata, se questo può essere tenuto nascosto.

Prima di commentare il codice perché altrimenti incomprensibile, tentare le seguenti strade:

1. Se si sono usati nomi poco (o del tutto non) chiari, sostituirli con nomi che spieghino lo scopo dell'entità (variabile locale, parametro, attributo, metodo, classe, package...)
2. Se il corpo di un metodo è troppo grande, scomporre il metodo in più metodi (eventualmente con visibilità private)
3. Se i metodi o gli attributi di una classe sono troppo numerosi, scomporre la classe in classi con meno responsabilità (eventualmente con visibilità package)
4. Se la classe svolge uno specifico ruolo di un pattern, specificare il ruolo, il pattern e gli altri partecipanti (quali altri classi e che ruolo hanno); non descrivere il funzionamento del pattern.

Se il codice rimane comunque di difficile comprensione, inserire commenti in linguaggio naturale.

Suddivisione del lavoro

Si richiede di:

1. Indicare come è stato suddiviso il prodotto software. Ad es. il programma è stato diviso in "moduli": GUI, gestione/elaborazione dati, etc.
2. Indicare come è stato organizzato il processo di sviluppo (ad es. in quali fasi è stato suddiviso): analisi del problema, modellazione UML, implementazione, progettazione DB, realizzazione DB, e stesura della documentazione.
    - Per ogni attività specificare l'elenco degli strumenti utilizzati.
    - A chi sono state allocate le diverse attività rispetto alle varie parti del prodotto. Ad es. l'analisi è stata fatta da X, i modelli UML da X e Y in coppia, l'implementazione della GUI da X, l'implementazione della gestione dati da Y, etc. Queste informazioni sono richieste a tutti i gruppi, anche quelli con un solo partecipante. Naturalmente in questo caso si può omettere l'ultimo punto.

Valutazione del progetto

È auspicabile che il progetto possa essere un'occasione formativa di lavoro in team. Gli studenti sono pertanto invitati a svolgere il progetto in gruppi di lavoro composti da 3 / 4 studenti.

La valutazione del progetto avviene singolarmente per ciascuno studente. Il giorno dell'appello verrà comunicato tramite avviso sulla pagina del corso nella piattaforma di elearning. Durante la valutazione è richiesto allo studente di saper argomentare in modo opportuno le scelte progettuali, algoritmiche, e implementative adottate. In fase di discussione orale verrà verificata l'effettiva padronanza delle tecniche utilizzate attraverso una serie di domande.

La valutazione terrà conto dei seguenti fattori: l'aderenza del sistema realizzato ai requisiti proposti, i documenti di analisi e progettazione prodotti sia per la realizzazione del software che per il database (correttezza sintattica, semantica, completezza e leggibilità, minimalità dello schema logico), le scelte algoritmiche e di progettazione effettuate (design pattern), la qualità del codice sorgente prodotto (funzionalità, correttezza, facilità d'uso).

Consegna

Il progetto deve essere consegnato entro il giorno stabilito per ogni appello d'esame, utilizzando l'apposita form disponibile sulle pagine del Laboratorio Interdisciplinare B della piattaforma di e-learning.
Alla voce commenti, specificare il nome e le matricole degli studenti del gruppo coinvolto. Il progetto deve essere allegato in formato compresso ZIP.

I progetti devono essere corredati da:

1. documentazione
2. codice sorgente · file di build per Apache Ant (http://ant.apache.org/) / Maven (https://maven.apache.org/) per compilare il progetto, lanciare il server e i client, creare il database, creare la documentazione javadoc, etc.
3. eventuali librerie necessarie alla compilazione e/o all'esecuzione
4. file README con indicazioni sull'installazione e sulla compilazione, specificando i comandi Ant/Maven da utilizzare, ed indicazioni di particolari librerie, usate in modo non standard, specificarlo nel file di README.

Il progetto deve compilare correttamente (tramite il file di build), una volta espanso in una directory.

I progetti con errori di compilazione e/o di esecuzione non verranno valutati.

Una volta effettuata la consegna, agli studenti verrà comunicata una data in cui potranno sostenere la discussione orale del progetto e la valutazione del lavoro svolto.

Domande sul Progetto Per qualsiasi dubbio durante lo svolgimento del progetto:
pietro.colombo@uninsubria.it

L'oggetto della mail deve essere: "Laboratorio B: Domande"

Progetto valido a partire dall'appello di Giugno 2018