Relazione di progetto
“iUniversity”

Kelvin Oluwada Milare Obuneme Olaiya
Emanuele Orlietti

25 aprile 2021

Indice

Capitolo 1

Analisi

Il software è un gestionale per università che permette di tenere traccia degli aspetti principali nel contesto universitario, quali studenti iscritti, docenti e appelli d'esame.

1.1 Requisiti

Requisiti funzionali

- L'applicazione deve permettere all'utente di autenticarsi. Nello specifico un utente può essere uno studente, docente o un 'amministratore del sistema (admin)
- Un docente deve poter creare o annullare appelli d'esame.
- Uno studente deve potersi iscrivere o ritirare da un appello d'esame.
- Un admin deve poter inserire nel sistema le informazioni relative ai docenti, studenti, Insegnamenti e Corsi di laurea.
- Uno studente deve poter visualizzare gli esiti di un'appello d'esame

Requisiti non funzionali

- Le password degli utenti devono essere cifrate prima di essere salvate.

1.2 Analisi e modello del dominio

Il sistema è rivolto a tre tipologie di utenti: Amministratore (admin), studente e docente. L'utente amministratore è colui che si occupa di inserire i

dati sugli insegnamenti erogati dall'università, i corsi di laurea, gli studenti iscritti e i docenti. Un insegnamento è caratterizzato da un numero di crediti formativi (CFU) ed insieme ad altri insegnamenti andrà a comporre l'offerta formativa di un corso di laurea a cui uno studente puo essere iscritto. Un docente può essere titolare di uno o più insegnamenti. Ogni docente avrà la possibilità di pubblicare degli appelli di esame ai quali gli studenti possono iscriversi o ritirarsi e che si svolgono in una particolare data e fanno riferimento ad un particolare insegnamento. Per ogni appello di esame viene mantenuta una lista con gli studenti che intendono sostenere la prova d'esame. A seguito di una prova d'esame il docente può registrare un'esito che può essere positivo o negativo (studente che si ritira, viene respinto o rifiuta la valutazione proposta).

Gli elementi costitutivi il problema sono riassunti in Figura 1.1.

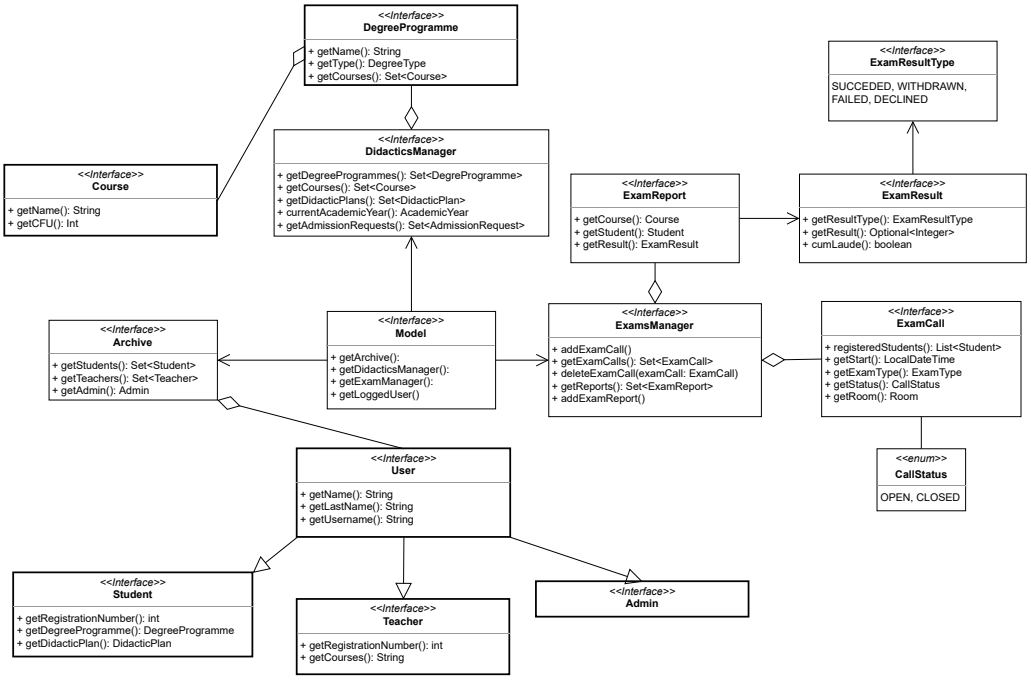

Figura 1.1: Schema UML dell'analisi del problema, con rappresentate le entità principali ed i rapporti fra loro

Capitolo 2

Design

2.1 Architettura

L'architettura del software segue il pattern MVC ed è organizzata in componenti. Ogni vista ha il suo rispettivo controller che si occupa dell'interazione con il dominio. All'avvio sarà il LoginController ad avere il controllo del domino e della LoginView. Il controller passera poi il controllo ad altri controller a seconda della tipologia di utente che si autentichera al sistema. Con questa architettura eventuali modifiche alla view, come ad esempio cambiare il framework per l'interfaccia grafica, non comportano modifiche al controller. Stessa cosa si puo dire per le interazioni tra controller e il model. Non vi sono interazioni tra view e model. Tutti i controller saranno classi passive le quali reagiscono in seguito a richieste provenienti dalle view. Il model modella il dominio e ne mantiene i dati. In Figura 2.1 è esemplificato il diagramma UML architetturale con i principali componenti.

2.2 Design dettagliato

Kelvin Olaiya

Mi sono occupato della modellazione delle attività che riguardano gli esami.

Ho utilizzato il pattern Builder per la creazione degli appelli d'esame. Questo mi ha consentito di separare la logica e la verifica della creazione dell'istanza dalla classe che rappresenta l'entità. Il pattern avrà beneficio eventualmente anche in futuro, in particolare se avrò necessita di variare i parametri di creazione poiché questa variazione non avrà impatti significativi nei costruttori della classe ma si tradurrà in modifiche del builder. Inoltre

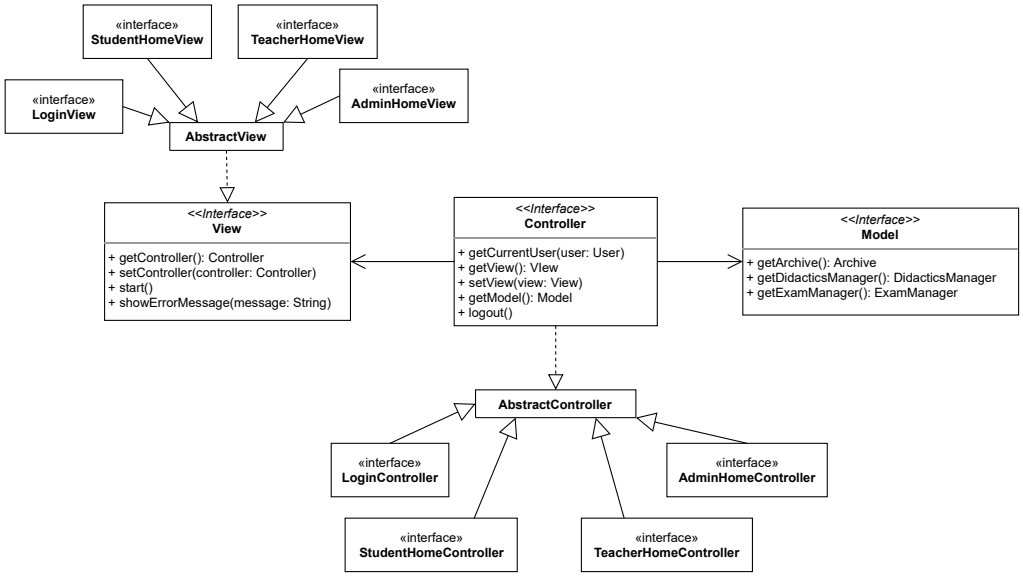

Figura 2.1: Schema UML architetturale di iUniversity.

per garantire praticità di utilizzo ho adottato lo stile fluente. Si veda la Figura 2.2

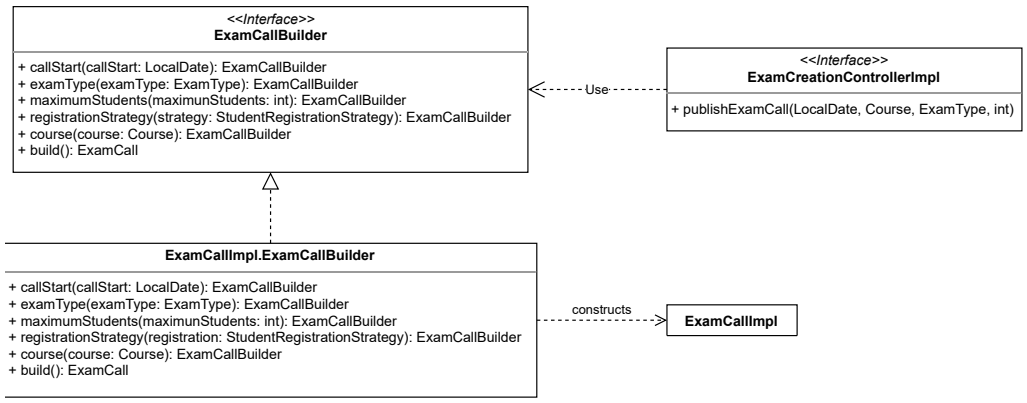

Figura 2.2: Builder per la creazione degli appelli d'esame

Un appello d'esame prevede il mantenimento di una lista ordinata degli studenti iscritti. Per incapsulare la logica di inserimento di uno studente in lista ho adottato il pattern Strategy. In questo modo le implementazioni di `ExamCall` non dovranno preoccuparsi di definire i diversi algorit-

mi di inserimento in lista e delegheranno il compito alla strategy. Sono state individuate tre strategie di inserimento: in testa alla lista, in coda alla lista ed in ordine alfabetico. Le strategy implementano l'interfaccia `StudentRegistrationStrategy`. Si veda la Figura 2.3

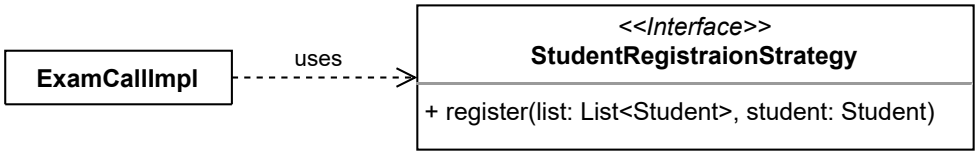

Figura 2.3: Strategy per l'inserimento in lista degli studenti

Nulla vieta che in futuro si possano definire altre strategie più particolari, per questo ho deciso di introdurre una factory che innanzitutto raccoglie le tre strategie sopra elencate e mi permetterà qualora ne avrò bisogno di aggiungerne delle altre con facilità. In figura Figura 2.4 viene mostrato l'utilizzo della factory.

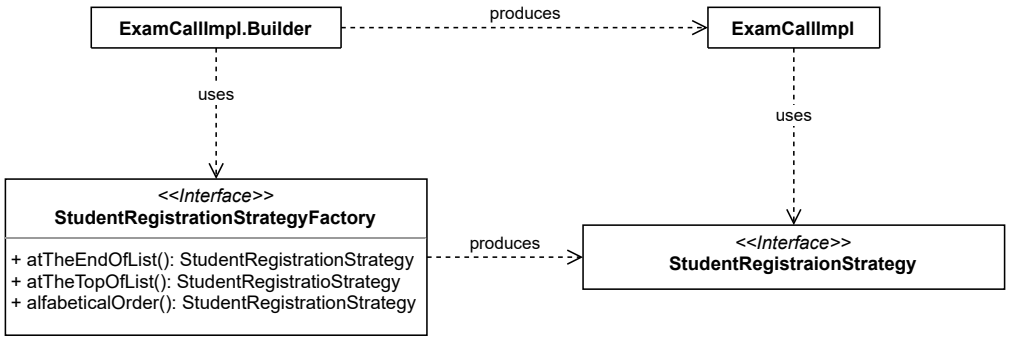

Figura 2.4: Factory per la creazione delle strategie di registrazione degli studenti agli appelli d'esame

Ho utilizzato il pattern Builder anche per la creazione dei verbali d'esame. Questi infatti hanno bisogno o meno di alcuni argomenti a seconda della tipologia di verbale che si vuole instanziare (ad es. non è richiesta una valutazione numerica se lo studente si ritira da un esame). Anche qui il pattern dunque evita l'introduzione di costruttori diversi per ricoprire le varie casistiche e si occupa esso stesso di verificare la correttezza della creazione dell'oggetto. Si veda la Figura 2.5

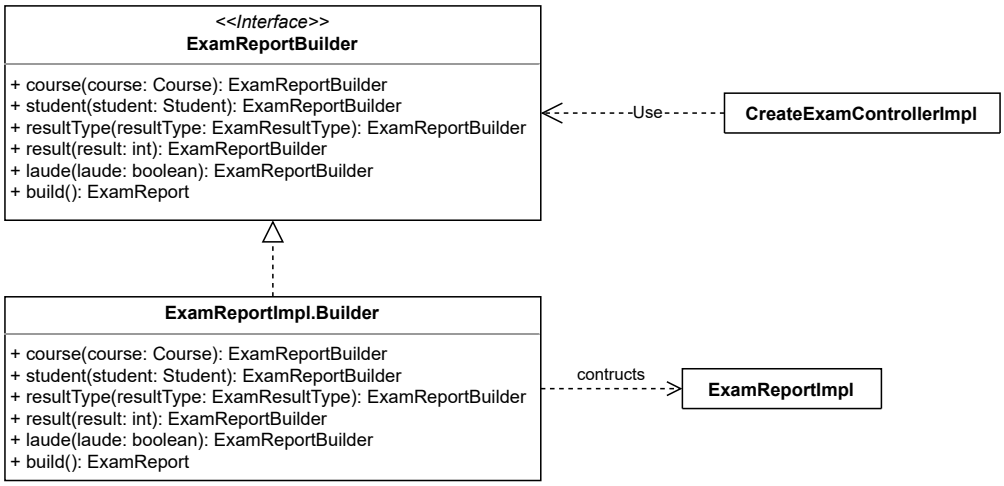

Figura 2.5: Builder per la creazione dei verbali d'esame

Per la creazione delle valutazioni che andranno a comporre i verbali d'esame è stato utilizzato il pattern Simple Factory. L'interfaccia `ExamResultFactory` offre i metodi per la istanziazione delle varie tipologie di esiti di un esame. Si veda la Figura 2.6.

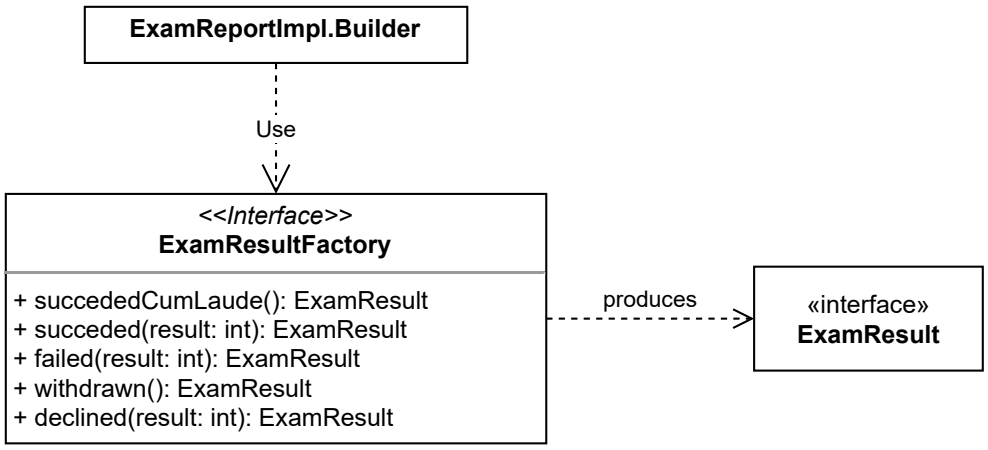

Figura 2.6: factory per la creazione delle diverse tipologie di esiti d'esame

Per il salvataggio dei dati ho utilizzato il pattern strategy. In questo modo chi utilizza la strategia non si deve preoccupare di come i dati vengono

effettivamente salvati. Attualmente esiste una sola strategia che gestisce la persistenza dei dati tramite scritture e letture su file. In futuro sarà possibile implementare nuove strategie che possono interfacciarsi con un database oppure con delle api in rete, il tutto senza cambiare la logica di chiamata negli utilizzatori. L'interfaccia che realizza la strategy è `DataStore`. Si veda la Figura 2.7.

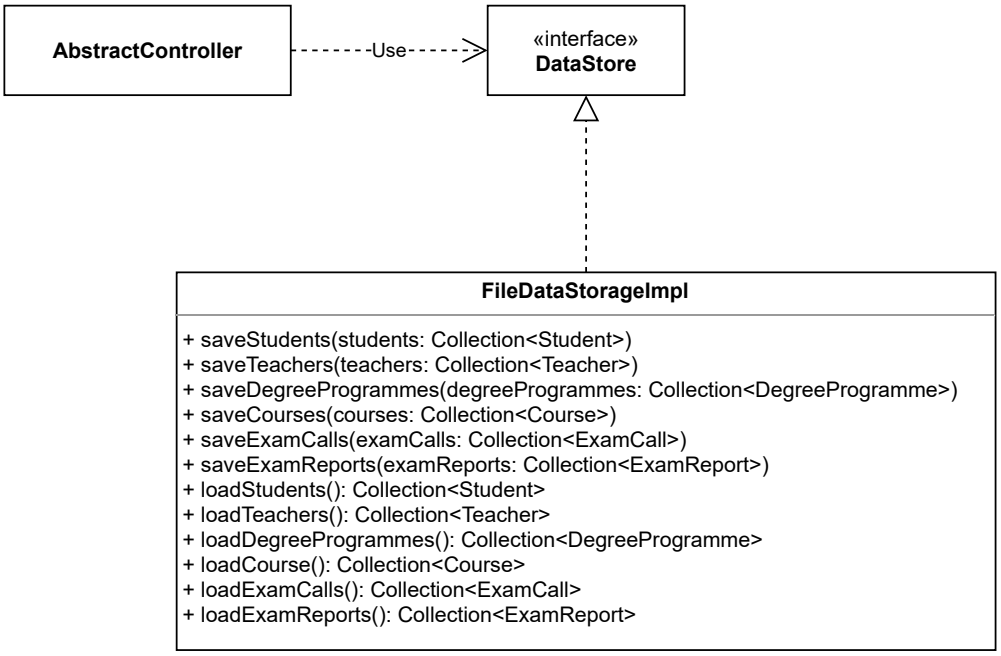

Figura 2.7: strategia per il salvataggio dei dati

Emanuele Orlietti

Riguardo la creazione del docente e dello studente è stato utilizzato il pattern Builder. Grazie al pattern sopraccitato si è evitata la proliferazione dei costruttori e inoltre entrambe le classi nel momento della creazione non si devono preoccupare della logica di creazione. Il motivo principale è che sia lo studente e sia il docente, hanno tanti parametri da inserire al momento della registrazione, con il Builder risulta tutto più facilitato grazie anche allo stile adottato, quello fluente. Si vedano le figure 2.6, 2.7.

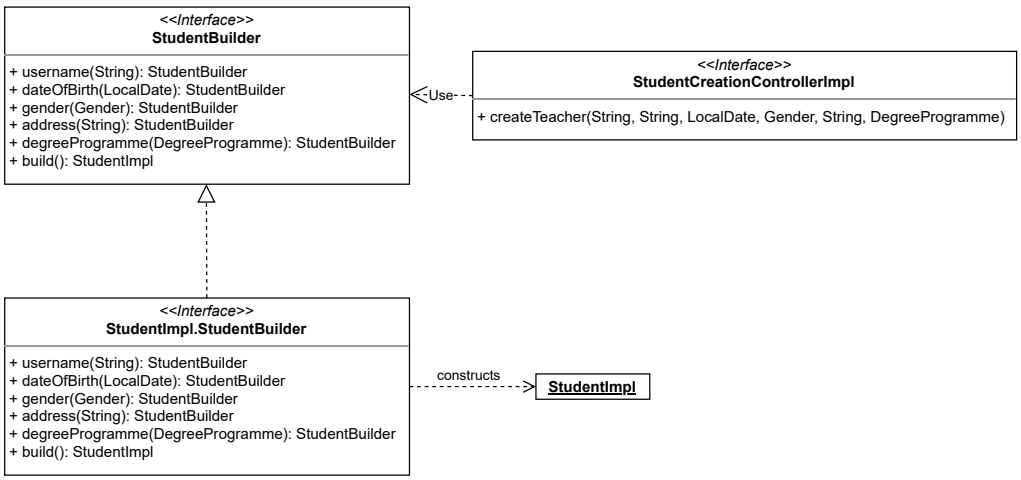

Figura 2.8: Builder per la creazione dello studente

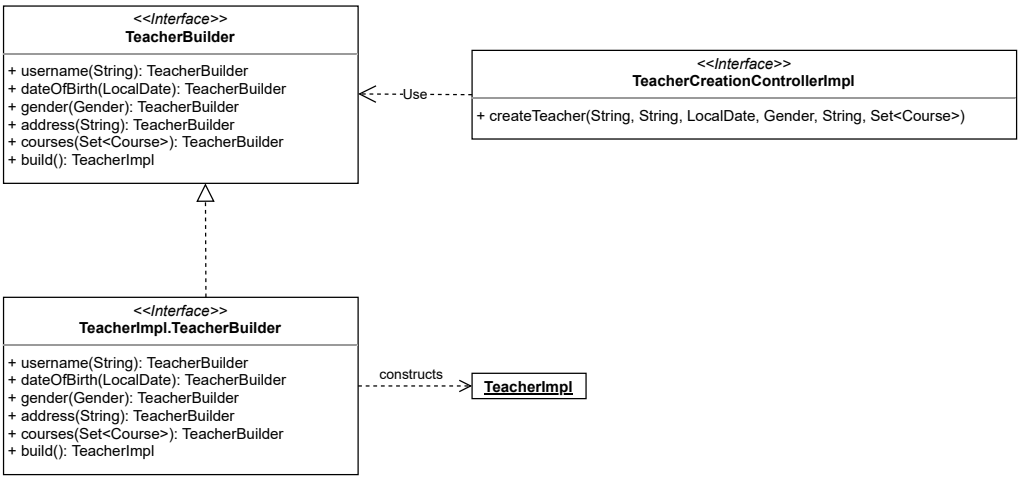

Figura 2.9: Builder per la creazione del professore

Capitolo 3

Sviluppo

3.1 Testing automatizzato

Per il testing automatizzato è stata utilizzata la suite JUnit. Sono state sottoposte a testing le classi apparteneti al Model poiché non richiedono una diretta interazione con l'utente.

- **ExamReportBuilderTest**: verifica che il builder crei correttamente i verbali d'esami aspettandosi il lancio di eccezioni nel caso in cui questo venga utilizzato nel modo scorretto.
- **ExamResultFactoryTest**: richiama i metodi della factory passando valori corretti e non corretti verificando il lancio di eccezioni in quest'ultimo caso.
- **ExamManagerTest**: controlla che i dati sugli appelli e verbali d'esame vengano memorizzati correttamente.
- **StudentRegistrationStrategyFactoryTest**: testa la correttezza dell'implementazione delle strategie di registrazione.
- **DidacticsManagerTest**: controlla la creazione dei corsi di laurea e degli insegnamenti.
- **ArchiveTest**: controlla il builder dello studente e del docente, la corretta matricola associata all'utente che intende registrarsi e la ricerca utilizzando la stessa.

Ulteriori test sono stati condotti manualmente come la verifica di funzionamento su sistema operativo linux e il funzionamento dell'interfaccia grafica.

3.2 Metodologia di lavoro

Il lavoro sull'analisi e l'architettura è stato svolto insieme. Particolare importanza è stata dedicata alla definizione delle interfacce poichè, ci ha consentito in un primo momento di poter lavorare nonostante l'assenza di alcune implementazioni. La divisione dei compiti è stata pressoché netta, tuttavia poichè Olaiya ha gestito il problema del salvataggio dei dati utilizzando la java serialization, ci sono stati dei suoi piccoli interventi nelle classi di competenza di Orlietti. Il DVCS è stato utilizzato lavorando ciascuno in feature-branch in locale ed effettuando merge e pull nel branch develop per condividere ciascuno il proprio lavoro.

Kelvin Olaiya

Mi sono occupato della gestione della fase di autenticazione. Inoltre mi sono occupato della gestione degli esami ovvero la creazione di appelli di esame da parte di un docente, la possibilità per gli studenti di prenotarsi e/o ritirarsi. Verbalizzazione degli esiti di un esame, visualizzazione dei dati statistici di studenti e docenti. Infine ho gestito il salvataggio dei dati. Di seguito l'elenco dei package a cui ho lavorato:

- iuniversity
- iuniversity.model
- iuniversity.model.exams
- iuniversity.storage
- iuniversity.storage.io
- iuniversity.controller
- iuniversity.controller.exams
- iuniversity.controller.home
- iuniversity.controller.login
- iuniversity.view
- iuniversity.view.exams
- iuniversity.view.home
- iuniversity.view.login

Emanuele Orlietti

Ho svolto la parte della gestione dell'inserimento, modifica e cancellazione dei dati. Sia per quanto riguarda gli utenti(studenti e docenti) e sia per la didattica(corsi di laurea, insegnamenti). Ecco l'elenco dei package a cui ho lavorato:

- iuniversity.controller.didactics
- iuniversity.controller.users
- iuniversity.model.didactics
- iuniversity.model.user
- iuniversity.view.didactics
- iuniversity.view.users

3.3 Note di sviluppo

Kelvin Olaiya

Ho fatto uso delle seguenti feature avanzate:

- Lambda expressions
- Stream
- Optional

utilizzato le seguenti librerie di terze parti

- `JavaFX`: come framework per le interfacce grafiche.
- `Apache commons-io`: per gestire l'I/O su file.
- `Apache commons-lang`: per l'utilizzo di strutture dati non presenti nel package java.lang.
- `Google guava`: per l'utilizzo di optional serializzabili.
- `Password4j`: per l'hashing delle password degli utenti.

Inoltre ho fatto uso del build system Gradle come gestore delle dipendenze. Ho riadattato l'uso del pattern Builder per la creazione di una configurazione per la gestione degli account di accesso, dando un'occhiata alle soluzioni del `laboratorio 08`[1] del corso.

Emanuele Orlietti

- `JavaFX`: per gestire la parte relativa alla GUI del progetto.

[1]https://github.com/APICe-at-DISI/OOP-Lab08 il builder in questione è quello dell'esercizio 6. advanced

Capitolo 4

Commenti finali

4.1 Autovalutazione e lavori futuri

Kelvin Olaiya

Tutto sommato ritengo di aver fatto un buon lavoro. Tuttavia a mio avviso sono molti i punti migliorabili. Per quanto riguarda l'architettura del software col senno di poi avrei cercato alternative ad MVC che forse si prestano meglio al dominio da noi scelto. Altro aspetto migliorabile è la grafica e l'usabilità del software, aspetti che credo che con un po' più di esperienza sarò in grado di assimilare. Per andare in positivo, il mio obiettivo principale è stato quello di tenere a mente il più possibile le good practice presentate durante il corso e far uso di alcuni aspetti avanzati quali stream e lambda expressions. Ritengo di aver ricoperto un ruolo centrale nel gruppo ma questo non ha fatto sì che non fossi aperto al confronto e a nuove proposte in fase di sviluppo. L'interazione con il mio collega, per lo svolgimento del progetto, non è stata semplice per via del diverso livello di preparazione, ma per vedere il bicchiere mezzo pieno sono situazioni che potrebbero capitare in ambito lavorativo ed averle vissute in questo progetto potrà essermi utile in futuro. Non è nelle mie intenzioni proseguire con lo sviluppo di questo specifico progetto. Quello che farò sarà raccogliere i commenti dei docenti, magari spunti di miglioramento che utilizzerò per sperimentare altre metodologie e approcci di sviluppo. Un altro ambito, abbastanza cruciale, su cui devo lavorare è il problem solving così che poi avrò modo di mettere in campo il massimo delle mie capacità.

Emanuele Orlietti

Non è stato assolutamente facile per me svolgere questo progetto anche perchè inizialmente eravamo in 4 ma alla fine ci siamo ritrovati in 2. Sicuramente, nonostante le difficoltà iniziali siamo riusciti a portarlo a termine con un buon risultato a mio parere. Ci si può sempre migliorare ovviamente,magari aggiungendo la bibilioteca, migliorare la GUI, ecc. Nonostante ciò mi reputo soddisfatto del lavoro svolto e di tutto ciò che ho imparato in queste 80 ore circa di lavoro, tornerà sicuramente utile in ambito lavorativo. Ho avuto alcuni problemi con l'utilizzo del pattern Builder ma fortunatamente, come sempre, ci ha pensato Kelvin ad aiutarmi.

4.2 Difficoltà incontrate e commenti per i docenti

Kelvin Olaiya

Questo per me è stato il corso più interessante e motivante fra tutti quelli che ho seguito fin'ora, merito della capacità che il professor Viroli ha nel trasmettere la materia. Il laboratorio mi ha permesso di avere uno sguardo più da vicino di quello che è un mestiere all'apparenza alla portata di molti ma che non tutti svolgono con **qualità**, parola quest'ultima che risuona nella testa e risuonerà in futuro ogni qualvolta dovrò mettere mano su un progetto. Le difficoltà che ho incontrato si sono presentate durante lo sviluppo del progetto. Purtroppo la motivazione iniziale è calata per via del ritiro di 2 nostri colleghi. Speravo in un clima di scambio di idee e confronti costruttivi. L'impatto più forte, soprattuto a livello emotivo, lo ha avuto la pandemia che stiamo tutti vivendo.

I commenti che vorrei dare riguardano le lezioni di laboratorio. Sono state una immensa opportunità per mettersi in gioco e fare personalmente un salto di qualità e per questo bisogna dare il merito ai docenti ed ai tutor. Secondo me alcuni temi trattati, sono stati illustrati forse troppo presto per far si che se ne apprezzasse il beneficio. A volte sono stati introdotti degli argomenti che non hanno avuto molto spazio per gli approfondimenti come ad esempio la libreria JavaFX, la cui trattazione potrebbe essere arricchita con più scenari d'uso di varia complessità e che si avvicinano di più all'uso che se ne fa in un realistico progetto software.

Appendice A

Guida utente

Credenziali di accesso

All'avvio dell'applicazione è necessario inserire le credenziali di accesso. Per il profilo admin l'username è **admin** e la password **admin**.

Sono presenti i seguenti profili docenti:

- **doc.viroli.mirko** (Corso Programmazione ad oggetti)
- **doc.caselli.fabrizio** (Corsi di Algebra e Matematica Discreta)

e i seguenti profili studente:

- **stu.rossi.mario**
- **stu.verdi.luciano**
- **stu.bianchi.luca**

La password è per tutti gli utenti **1234**

Inserimento dati

Nel profilo admin è possibile inserire i dati principali. Una volta inseriti gli insegnamenti è possibile inserire i corsi di laurea. Quando anche i corsi di laurea sono stati inseriti è possibile procedere con l'inserimento degli studenti e dei docenti.

Appelli d'esame

Il docente può creare un appello d'esame con almeno un giorno di anticipo. La lista rimane aperta per le iscrizioni/ritiri degli studenti fino ad un giorno prima della data dell'appello.

Il docente può eliminare un appello d'esame selezionandolo dalla schermata home e premento il pulsante `Elimina`. Lo Studente può ritirarsi da un appello d'esmae selezionandolo dalla schermata home e premendo il pulsante `Ritirati`.

Verbalizzazione dei voti

Il docente può verbalizzare i voti agli studenti iscritti ad un appello una volta che la lista si è chiusa.

Appendice B

Esercitazioni di laboratorio

B.0.1 Kelvin Olaiya

- Laboratorio 04: https://virtuale.unibo.it/mod/forum/discuss.php?d=62685#p101090
- Laboratorio 05: https://virtuale.unibo.it/mod/forum/discuss.php?d=62684#p101086
- Laboratorio 06: https://virtuale.unibo.it/mod/forum/discuss.php?d=62579#p100886
- Laboratorio 07: https://virtuale.unibo.it/mod/forum/discuss.php?d=62582#p100888
- Laboratorio 08: https://virtuale.unibo.it/mod/forum/discuss.php?d=63865#p102753
- Laboratorio 09: https://virtuale.unibo.it/mod/forum/discuss.php?d=64639#p103769
- Laboratorio 10: https://virtuale.unibo.it/mod/forum/discuss.php?d=66753#p106612
- Laboratorio 11: https://virtuale.unibo.it/mod/forum/discuss.php?d=66463#p106390