ALMA MATER STUDIORUM

REPORT LABORATORIO DI ALGORITMI E STRUTTRE DATI

PROGETTO: ALGAT

AlgaT - Applicazione Interattiva per la Visualizzazione di Algoritmi su Alberi Binari di Ricerca e Alberi Red-Black

Studenti
GIOVANNI FAZI
MICHELE DI STEFANO

Anno Accademico
2018/2019

May 8, 2019

AlgaT
ABR

Contents

Chapter 1

Analisi

In questo capitolo vengono analizzati i requisiti che l'applicazione deve soddisfare e descritti i casi d'uso che riguardano l'interazione dell'utente con l'applicazione.

1.1 Requisiti

1.1.1 Spiegazione interattiva

- all'apertura AlgaT mostra informazioni di contesto – ad esempio spiegazione testuale dell'algoritmo o credits – e permette di far partire il tutorial
- AlgaT non è un tutorial video ma un ambiente interattivo
- le lezioni dipendono dall'argomento scelto e sono decise dal gruppo ma devono essere almeno 2
- in ogni lezione, il tutorial mostra le strutture dati rilevanti e come queste cambiano durante l'esecuzione
- la lezione è organizzata in passi e l'utente controlla l'esecuzione e può scegliere ad esempio quando avanzare, attraverso appositi bottoni dell'interfaccia (granularità, organizzazione e numero di passi sono arbitrari)

1.1.2 Autoapprendimento

- AlgaT prevede anche un'area in cui l'utente può rispondere a domande associate alla lezione
- le domande dipendono dall'argomento scelto e sono decise dal gruppo ma devono essere almeno 5 per ogni lezione
- ogni gruppo decide il tipo di domande e la difficoltà

- l'applicazione mostra le domande una per volta e verifica la risposta dell'utente, seguita da eventuali spiegazioni
- non è possibile visualizzare la domanda successiva se l'utente non ha risposto correttamente alla precedente
- dopo aver risposto alle domande la lezione è conclusa e si torna al menù principale

1.2 Casi d'uso

Di seguito sono riportati i casi d'uso individuati in seguito all'analisi dei requisiti.
Le principali azioni che producono un comportamento completo e significativo per l'utente sono le seguenti:

- leggere la scheda introduttiva
- interagire con l'esecuzione dell'algoritmo
- rispondere alle domande
- navigare fra le lezioni
- visualizzare il codice eseguito dall' algoritmo

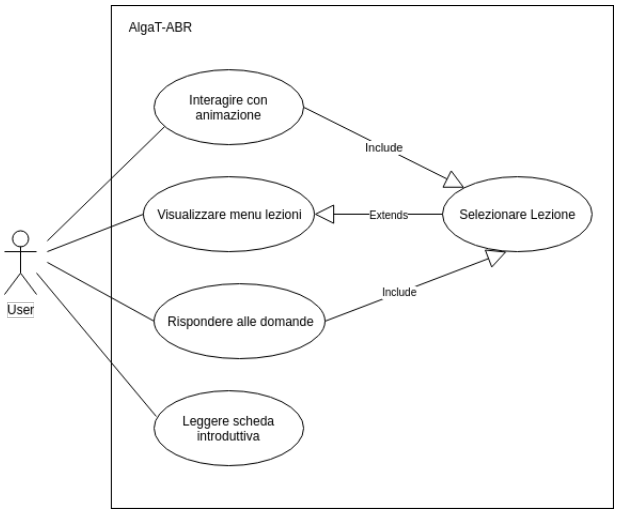

Figure 1.1: Casi d'uso

Chapter 2

Progettazione

2.1 Design Pattern

Nella fase di progettazione è stato seguito un approccio Object Oriented. Si è scelto di usare il design pattern Model View Controller perché ritenuto ottimale nella suddivisione delle responsabilità tra le varie componenti del sistema.

2.2 CRC card

Nel processo di identificazione delle classi principali sono state utilizzate le *Class-Responsibility-Collaboration* (CRC) card, strumento impiegato nella progettazione Object Oriented che permette di astrarsi dai dettagli implementativi e di focalizzarsi sugli elementi essenziali della classe, evitando così un livello di complessità che, in questa fase, potrebbe essere controproducente.

Controller Main	
Crea la finestra iniziale Carica lezioni Carica menu	Controller Menu Controller Lezione

Figure 2.1: Controller Main

Controller Menu	
Conosce titoli lezioni Mostra informazioni di contesto Modifica la rispettiva View	Controller Main

Figure 2.2: Controller Menu

Controller Lezione	
Inizializza il controller dello Pseudocodice Inizializza il controller delle Domande Inizializza il controller delle Animazioni Gestisce la comunicazione tra controller Modifica la rispettiva View Mostra il messaggio di benvenuto	Controller Main Controller Pseudocodice Controller Domande Controller Animazioni

Figure 2.3: Controller Lezione

Controller Pseudocodice	
Conosce i metodi da caricare Modifica la rispettiva View Mostra il codice eseguito nella animazione	Controller Lezione

Figure 2.4: Controller Pseudocodice

Controller Domande	
Conosce testo domande Conosce risposte domande Conosce spiegazioni domande Modifica la rispettiva View Gestisce la fine della lezione	Controller Main Controller Lezione

Figure 2.5: Controller Domande

Controller Animazioni	
Permette di eseguire animazioni Conosce lo stato della struttura dati che rappresenta Modifica la rispettiva View Comunica lo step di codice eseguito	Model Albero Binario Model Albero Red Black Controller Lezione

Figure 2.6: Controller Animazioni

Model Albero Binario	
Crea la struttura dati Gestisce la struttura dati Fornisce informazioni sui passi di esecuzione	Controller Animazioni

Figure 2.7: Modello ABR

Model Albero Red Black	
Crea la struttura dati Gestisce la struttura dati Fornisce informazioni sui passi di esecuzione	Controller Animazioni

Figure 2.8: Modello RB

2.3 Altre scelte progettuali

Al fine di creare una struttura flessibile è stato scelto di memorizzare lezioni ed altre informazioni rilevanti su un file esterno riducendo il numero di elementi "hard-coded" nel sistema. Tale file esterno contiene:

- informazioni descrittive delle lezioni (titolo, scopo lezione, etc)
- domande, risposte ed eventuali spiegazioni di una lezione
- funzioni in pseudocodice rilevanti per la lezione
- tipo di struttura dati usata nella lezione

Chapter 3

Implementazione

3.1 Java package

Per motivi di maggior leggibilità, di facilitazione della manutenzione e per raggruppare classi che contribuiscono ad espletare una determinata funzionalità, il codice è stato suddiviso in 4 *package*: app, controller, model, view (vedi Figure 3.1).

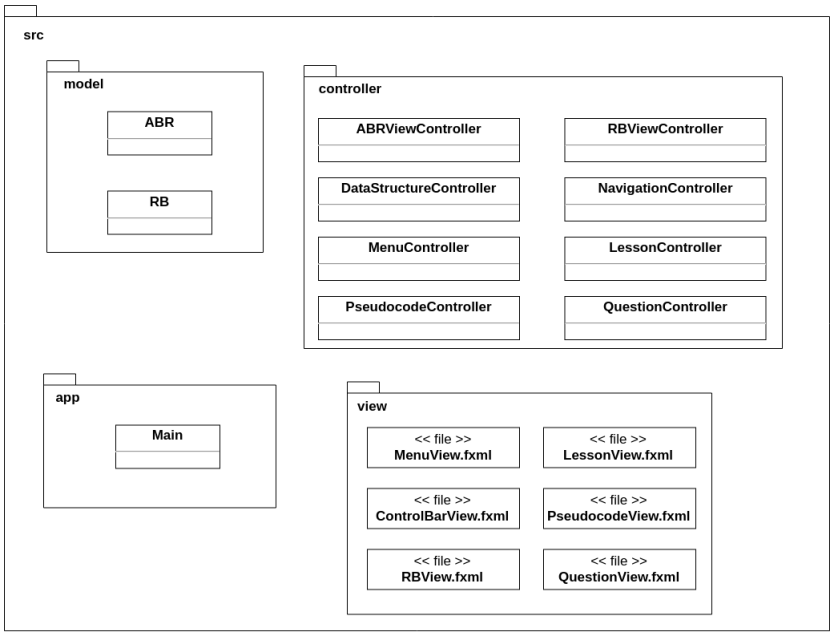

Figure 3.1: java package

app

In questo package è contenuta la classe `Main` che si occupa dell'avvio dell'applicazione

model

In questo package sono contenute le classi `ABR` e `RB` che rappresentano i modelli, rispettivamente, di alberi binari di ricerca e alberi red-black

view

In questo package sono contenuti i file `*.fxml` corrispondenti alle diverse view.

controller

In questo package sono contenuti i controller che si occupano di modificare i modelli e le view.

3.2 Javadoc

Per ulteriori dettagli sull'implementazione delle classi si rimanda al Javadoc allegato.