

POLITECNICO DI TORINO

Corso di Laurea Triennale in
Ingegneria Gestionale

Anno Accademico 2021/2022

Tesi di Laurea Triennale

Software per la simulazione di un anno scolastico



**Politecnico
di Torino**

Relatore:

Prof. Fulvio Corno

Candidato:

Lorenzo Canini

Ottobre 2022

Sommario

Capitolo 1:	5
Proposta di progetto	5
1.1 Titolo della proposta	5
1.2 Descrizione del problema proposto	5
1.3 Descrizione della rilevanza gestionale del problema	5
1.4 Descrizione dei data-set per la valutazione	5
1.5 Descrizione preliminare degli algoritmi coinvolti	6
1.6 Descrizione preliminare delle funzionalità previste per l'applicazione del software	6
Capitolo 2:	7
Descrizione dettagliata del problema	7
Capitolo 3:	8
Descrizione del data-set utilizzato per l'analisi	8
3.1 Struttura data-set	8
3.2 Tabelle	9
3.2.1 Voti	9
3.2.2 Materie	10
Capitolo 4:	11
Descrizione delle strutture dati e degli algoritmi utilizzati	11
4.1 Strutture dati	11
4.2 Algoritmi principali	12
4.2.1 creaScuola()	12
4.2.2 creaClasse()	13
4.2.3 simulaAnno()	14
4.2.4 calcolaVoto()	14
4.2.5 calcolaPercentuali()	15
4.2.6 simulaVoto()	16
4.2.7 calcolaRisparmi()	16
Capitolo 5:	17
Diagramma delle classi principali	17
Capitolo 6:	18
Videate dell'applicazione e collegamento al video	18
6.1 Videate dell'applicazione	18
6.2 Video dimostrativo	19

Capitolo 7:	20
Risultati sperimentali	20
Capitolo 8:	22
Conclusione e valutazione dei risultati ottenuti	22

Capitolo 1:

Proposta di progetto

1.1 Titolo della proposta

Software per la simulazione di un anno scolastico

1.2 Descrizione del problema proposto

Il software ha come obiettivo quello di simulare l'andamento scolastico di tutti gli alunni all'interno di un'istituzione scolastica.

I risultati ottenuti in seguito a questa ricostruzione saranno utilizzati dal programma per indicare all'utente la divisione del budget scolastico. Questo perché la scuola divide i fondi che ha a disposizione in base al numero di alunni iscritti per offrire più servizi possibili come, per esempio, l'acquisto dei libri di testo nuovi, l'organizzazione di corsi di recupero per gli studenti bocciati o rimandati e la realizzazione dei vari progetti scolastici (gite d'istruzione, incontri con esperti, ...)

1.3 Descrizione della rilevanza gestionale del problema

Il software risulta utile dal punto di vista gestionale perché può essere utilizzato da qualsiasi istituzione scolastica a inizio anno per prevedere quale sarà il numero di alunni promossi, bocciati o rimandati. Questo permetterà alla scuola di scegliere in maniera più precisa quanto budget dedicare alle attività previste: un numero maggiore di rimandati porterà a stanziare più budget per i corsi di recupero, come un maggior numero di bocciati in quarta porterà a stanziare meno budget per il viaggio di istruzione delle quinte.

1.4 Descrizione dei data-set per la valutazione

Il software utilizzerà i data-set che l'istituzione scolastica presso qui ho insegnato mi ha fornito. Questi file contengono gli andamenti scolastici di tutti gli alunni degli ultimi 5 anni scolastici. All'interno di questi file è possibile trovare le medie di ogni alunno per tutte le materie, le ore di assenza, la media generale dei voti e l'esito degli scrutini (ammesso/non ammesso). Tutti i dati sensibili (come nome e cognome) sono stati censurati.

1.5 Descrizione preliminare degli algoritmi coinvolti

Il programma conterrà algoritmi di simulazione per riprodurre il comportamento degli alunni lungo l'anno scolastico, questi permetteranno di vedere i voti che ogni alunno si ritroverà in pagella alla fine dell'anno e che decreteranno se lo studente è stato promosso, bocciato o rimandato.

Il programma utilizzerà anche algoritmi di ottimizzazione sia per la creazione di ogni classe, questo perché in base ai dati inseriti dall'utente il software cercherà di distribuire in maniera uniforme gli alunni all'interno delle classi, sia per quanto riguarda la divisione del budget, adattando i costi in base al numero di studenti iscritti.

1.6 Descrizione preliminare delle funzionalità previste per l'applicazione del software

Il software chiederà all'utente di inserire il numero di alunni iscritti ad ogni anno (questi numeri cambiano ogni anno e potranno essere inseriti dal dirigente scolastico a inizio anno una volta ricevute le iscrizioni). Dopodiché partirà la simulazione. Una volta terminata sarà possibile visualizzare gli esiti di tutte le classi presenti nell'istituto e per ogni alunno saranno mostrate le medie di tutte le materie.

La seconda parte dell'applicazione permetterà al dirigente di inserire l'importo ottenuto dai finanziamenti e il software calcolerà automaticamente tutte le spese previste, basandosi sui dati delle simulazioni, e fornirà in output il budget rimanente alla scuola per realizzare i vari progetti scolastici.

Capitolo 2:

Descrizione dettagliata del problema

L'applicazione si pone come obiettivo quello di aiutare un dirigente e i suoi collaboratori a programmare al meglio l'anno scolastico che sta per cominciare.

Il problema principale che il software mira a risolvere è quello della gestione dei finanziamenti che la scuola ottiene ogni anno. Infatti, ogni scuola riceve delle sovvenzioni annuali per finanziare nuovi progetti e per coprire le spese dell'istituto. Il compito dell'applicazione è quello di calcolare automaticamente la parte di spese e indicare la parte restante di budget da poter dedicare al finanziamento di nuovi progetti. La difficoltà principale nel calcolo di queste spese sta nel fatto che alcune di esse non sono fisse, come per esempio le spese per il personale, ma variano a seconda di vari fattori. Alcuni elementi sono facili da calcolare, come ad esempio le spese amministrative, che corrispondono sempre alla stessa percentuale del budget, altri sono più complicati, come per esempio le spese per i corsi di recupero. Infatti, questi ultimi dipendono dal numero di studenti che saranno bocciati o rimandati durante l'anno e non potendo prevedere il futuro, può essere complicato stimare questi costi.

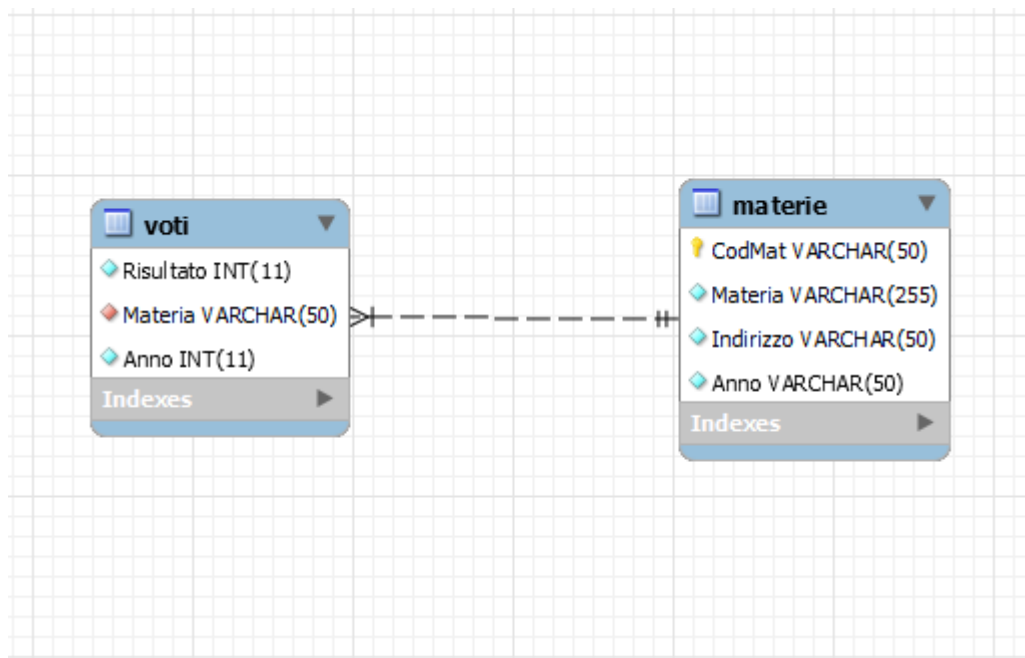
L'applicazione prova a risolvere questo problema andando a simulare completamente un anno scolastico, restituendo in output i risultati ottenuti dai singoli studenti. Grazie a questi dati il software verifica il numero di studenti che sono stati promossi, rimandati o bocciati e aggiusta le spese di conseguenza.

Capitolo 3:

Descrizione del data-set utilizzato per l'analisi

Il database utilizzato è stato creato utilizzando i file che mi sono stati forniti dall'istituzione scolastica in cui ho insegnato durante l'anno. Ogni file conteneva i tabelloni degli scrutini finali di tutte le classi dell'istituto per un anno scolastico. Sono stati presi in considerazione i tabelloni degli anni scolastici 2016/2017, 2017/2018, 2018/2019. Sono stati esclusi i tabelloni degli anni scolastici 2019/2020 e 2020/2021 perché ho ipotizzato che i risultati ottenuti dagli alunni in questi due anni siano stati influenzati, in positivo e in negativo, dall'alternanza tra lezioni in presenza e didattica a distanza.

3.1 Struttura data-set



Come si può vedere dal diagramma entità relazione proposto, il database è stato diviso in due tabelle. La tabella "voti", come dice il nome, contiene tutti i voti presi negli anni dagli alunni di tutta l'istituzione. Mentre nella tabella "materie" sono presenti tutte le materie che vengono insegnate nei vari anni all'interno dell'istituzione.

3.2 Tabelle

3.2.1 Voti

Risultato	Materia	Anno
3	DEE/IT	1
4	DEE/IT	1
4	DEE/IT	1
4	DEE/IT	1

I dati all'interno della tabella sono descritti da tre attributi: Risultato, Materie e Anno.

- Risultato: è il voto presente sui tabelloni degli scrutini di fine anno.
- Materie: indica il codice della materia a cui fa riferimento il voto e si comporta da chiave esterna con la tabella materie.
- Anno: indica la classe in cui è stato preso il voto. Il dominio va da 1, per le prime, a 5, per le quinte.

3.2.2 Materie

CodMat	 Materia	Indirizzo	Anno
DEE/AFM	Diritto ed economia	AFM	B
DEE/CAT	Diritto ed economia	CAT	B
DEE/IT	Diritto ed economia	IT	B
DIR/AFM	Diritto	AFM	T
EA/AFM	Economia Aziendale	AFM	S
EP/AFM	Economia politica	AFM	T
FRA/AFM	Lingua e letteratura francese	AFM	S

I dati all'interno della tabella sono descritti da quattro attributi: Codice Materia, Materia, Indirizzo e Anno.

- Codice Materia: ha la funzione di chiave primaria e viene utilizzato per rendere le materie univoche.
- Materia: indica il nome completo della materia.
- Indirizzo: l'istituzione scolastica che mi ha fornito i dati ha tre diversi indirizzi di studi. Quindi per poter analizzare separatamente i risultati tra i singoli indirizzi ho usato questo attributo.
- Anno: indica l'anno in cui viene insegnata la materia e può assumere diversi valori:
 - S: la materia è insegnata per tutti e cinque gli anni.
 - B: la materia è insegnata solo al biennio.
 - T: la materia è insegnata solo al triennio.
 - 1,2,3,4,5: indica che la materia è insegnata solo in uno di questi anni o che è insegnata fino a quell'anno (es. informatica, nell'indirizzo AFM, è insegnata dalla prima alla quarta e quindi nella colonna anno presenta il valore 4).
 - 34: indica che la materia è insegnata solo in terza e quarta.

Capitolo 4:

Descrizione delle strutture dati e degli algoritmi utilizzati

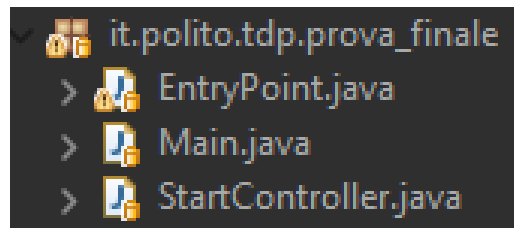
4.1 Strutture dati

L'applicazione è stata creata usando il linguaggio di programmazione Java e utilizzando il pattern MVC (Model View Controller) e il pattern DAO (Data Access Object). Questi due pattern permettono di tenere separata la logica applicativa, dalla struttura dati e dall'interfaccia grafica.

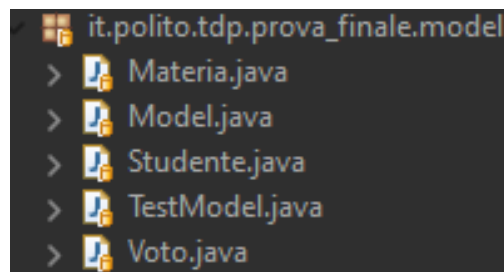
Per la parte grafica dell'applicazione è stato usato JavaFX e il software SceneBuilder.

L'applicazione è divisa quindi in tre package:

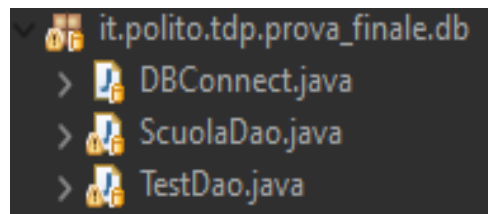
- Il package *it.polito.tdp.prova_finale* si occupa di gestire la parte grafica, permettendo all'utente di vedere la scena e di interagire con l'applicazione inserendo i dati e azionando i bottoni



- Il package *it.polito.tdp.prova_finale.model* si occupa di gestire la logica applicativa del software. Al suo interno sono presenti tutte le classi di oggetti che vengono utilizzati dal programma ed è presente la classe *Model*, dentro la quale si trovano tutti gli algoritmi che permettono all'applicazione di funzionare.



- Il package *it.polito.prova_finale.db* si occupa di gestire tutte le interazioni col database. Infatti, all'interno della classe DBConnect è presente la funzione che permette di connettere il software con il database e nella classe ScuolaDAO sono presenti tutti i metodi che permettono di interrogare il database.



4.2 Algoritmi principali

4.2.1 creaScuola()

Il primo algoritmo che attiviamo facendo partire la nostra simulazione è formato da due metodi: *creaScuola()* e *creaClasse()*. Il primo è un metodo molto semplice che riceve in input tutti i dati inseriti dall'utente (numero di alunni per ogni anno e numero di classi per ogni anno) e richiama 5 volte il metodo *creaClasse()*, per andare a creare le varie classi che compongono l'istituto.

```
public void creaScuola(int nStud1, int nStud2, int nStud3, int nStud4, int nStud5, int nClassi1, int nClassi2, int nClassi3, int nClassi4, int nClassi5, String indirizzo) {  
    creaClasse(nStud1, nClassi1, 1, indirizzo);  
    creaClasse(nStud2, nClassi2, 2, indirizzo);  
    creaClasse(nStud3, nClassi3, 3, indirizzo);  
    creaClasse(nStud4, nClassi4, 4, indirizzo);  
    creaClasse(nStud5, nClassi5, 5, indirizzo);  
}
```

4.2.2 creaClasse()

Il metodo creaClasse(), invece, si occupa di andare a dividere gli alunni all'interno delle varie classi. La funzione riceve in input il numero di studenti, il numero di classi, l'anno e l'indirizzo, calcola il numero di studenti da assegnare ad ogni classe e poi crea un numero di oggetti Studente pari al numero fornito in input. Ad ogni Studente viene assegnata una matricola con numero progressivo, l'anno di appartenenza, la sezione di appartenenza e l'indirizzo a cui è iscritto.

Inoltre, se il numero di studenti non è divisibile perfettamente per il numero di classi, il programma si occupa di dividere il resto di studenti tra le varie classi, in modo da creare classi uniformi.

```
private void creaClasse(int NStud, int NClassi, int anno, String indirizzo) {
    int classe = 0;
    int resto = 0;
    if (NStud%NClassi == 0) {
        classe = NStud/NClassi;
    }else {
        classe = NStud/NClassi;
        resto = NStud%NClassi;
    }
    for (int x = 0; x < NClassi; x++) {
        for(int i = 0; i < classe; i++) {
            String idStud = "STU" + String.valueOf(scuola.size() + 1);
            Studente stud = new Studente(idStud, anno, sezioni[x], indirizzo);
            scuola.add(stud);
        }
    }
    for (int y = 0; y < resto; y++) {
        String idStud = "STU" + String.valueOf(scuola.size() + 1);
        Studente stud = new Studente(idStud, anno, sezioni[y], indirizzo);
        scuola.add(stud);
    }
}
```

4.2.3 simulaAnno()

Questo è il secondo algoritmo che viene lanciato per avviare la simulazione e si occupa di stimare quali saranno i voti di ogni studente. Il metodo non riceve dati in input, ma semplicemente scorre la Lista scuola, che viene riempita da Studenti dal metodo precedente, e per ognuno di essi calcola un voto per ogni materia grazie al metodo calcolaVoto(). Il risultato che viene ottenuto viene aggiunto alla pagella di ogni studente. Questa pagella non è altro che una Lista di Voti che ogni Studente possiede.

```
public void simulaAnno() {
    for (Studente s : scuola) {
        List<Materia> materie = dao.getMaterie(s.getClasse(), s.getIndirizzo());
        for (Materia mat : materie) {
            Voto v = new Voto(calcolaVoto(s.getIndirizzo(), s.getClasse(), mat.getCodMat()), s.getClasse(), mat.getMateria());
            s.aggiungiVoto(v);
        }
    }
}
```

4.2.4 calcolaVoto()

Il metodo calcolaVoto() è uno dei tanti piccoli algoritmi che permettono di arrivare a calcolare i voti assegnati a tutti gli studenti nelle varie materie. Per calcolare il voto che uno studente prenderà mi sono basato sulla probabilità che hanno i singoli eventi (voti) di presentarsi, utilizzando come campione tutti i voti ottenuti dagli studenti nelle singole materie e separati per indirizzo negli anni analizzati. Quindi, per esempio, per simulare i voti di italiano in prima, indirizzo IT, ho utilizzato come campione tutti i voti presi in prima IT negli anni analizzati e ho calcolato la probabilità di estrarre ciascun voto (1, 2, 3, ..., 9, 10) come casi favorevoli/ casi totali.

Il metodo otterrà i voti di un determinato anno e di una determinata materia grazie alla funzione getAllVoti() e calcolerà in percentuale le volte che un singolo voto si è presentato grazie alla funzione calcolaPercentuali(). In seguito, l'algoritmo utilizzerà questo insieme di percentuali per creare un vettore di fasce. Il vettore che viene creato rappresenta le probabilità di estrazione di un singolo voto. Infatti, il software per simulare il voto estrarrà a caso un numero compreso tra 1 e 100 e in base ai valori delle fasce questo numero genererà un voto. Facendo un esempio concreto, ipotizziamo che le fasce siano 1, 5, 9, 15, 40, 75, 87, 95, 99, 100 (ogni fascia corrisponde a un voto quindi la prima corrisponde al 1, la seconda al 2 e così fino alla decima che corrisponde al 10) e supponiamo di aver estratto casualmente il numero 65. Il programma andrà a controllare a quale fascia appartiene questo numero e genererà il voto corrispondente. Nel nostro esempio 65 è

maggiore di 40 e minore di 75 e quindi il voto corrispondente sarà 6. Tutto questo viene fatto dalla funzione creando il vettore probabilità[] e utilizzando la funzione simulaVoto().

```
private int calcolaVoto(String indirizzo, int classe, String materia) {
    List<Voto> voti = new ArrayList<Voto>();
    voti = dao.getAllVoti(indirizzo, classe, materia);
    int[] percentuali = calcolaPercentuali(voti);
    int[] probabilità = new int[10];
    int somma = 0;
    for (int i = 0; i < 10; i++) {
        somma = somma + percentuali[i];
        probabilità[i] = somma;
    }
    int voto = simulaVoto(probabilità);
    return voto;
}
```

4.2.5 calcolaPercentuali()

Questo metodo, come già spiegato sopra, calcola in percentuale la probabilità che ogni voto ha di essere ottenuto. Utilizza il metodo contaVoti(), che semplicemente conta quante volte è presente un determinato voto all'interno della lista passata come parametro, per ottenere i casi favorevoli e li divide per la dimensione della lista (i casi totali).

Tutte le percentuali vengono arrotondate per difetto per due motivi: il primo è quello di ottenere delle fasce senza decimali, il secondo è legato alla verosimiglianza degli output. Infatti, alcuni dei 6 che comparivano nei tabelloni finali erano dovuti al superamento degli esami di recupero di settembre. Quindi, alcuni di questi voti sufficienti a settembre, non lo erano a giugno. Per sopperire a questo inconveniente ho deciso di aumentare leggermente le probabilità di uscita del 5, andando a sommare alla sua percentuale (ottenuta con casi favorevoli/casi contrari), la somma di tutti i decimali andati persi a causa degli arrotondamenti.

```
private int[] calcolaPercentuali(List<Voto> v) {
    int[] percentuali = new int[10];
    int somma = 0;
    for (int i = 0; i < 10; i++) {
        double percentuale = Math.floor(((double)contaVoti(i+1, v)/((double)v.size()))*100);
        percentuali[i] = (int)percentuale;
        somma = somma + (int)percentuale;
    }

    if (somma == 100) {
        return percentuali;
    } else {
        int a = 100 - somma;
        percentuali[4] = percentuali[4] + a;
        return percentuali;
    }
}
```

4.2.6 simulaVoto()

Questa è l'ultima funzione della simulazione che andremo ad analizzare. Con questo metodo il software estrae a caso un numero tra 1 e 100 e poi confronta il numero ottenuto con i valori del vettore probabilità, passato come parametro. Appena trova una fascia maggiore rispetto al numero estratto, il metodo si ferma e restituisce l'indice *i* che stava usando per scorrere il vettore. Quell'indice, a cui sommiamo uno visto che gli indici vanno da 0 a 9, mentre i voti vanno da 1 a 10, corrisponde al voto che noi attribuiremo allo studente.

```
private int simulaVoto(int[] probabilità) {
    double v = Math.random() * 100;
    int i = 0;
    for (i = 0; i < 10; i++) {
        if (v < probabilità[i]) {
            break;
        }
    }
    return i + 1;
}
```

4.2.7 calcolaRisparmi()

Questa funzione si occupa di calcolare le varie spese che la scuola dovrà sicuramente affrontare e fornirà in output i fondi rimanenti per i progetti scolastici. I costi fissi della scuola si dividono in Spese Amministrative, che corrispondono a circa il 20% dei finanziamenti, Spese per il personale, che corrispondono a circa 144'000€, Spese per i libri di testo, che corrispondono a circa 100€ per studente, e infine le Spese per i corsi di recupero, che corrispondono a 25€/h per corsi da 8h e dipendono dal numero di materie che hanno almeno 4 alunni rimandati (se il numero è inferiore a 4 la scuola non organizza nessun corso). Tutti questi dati si basano sui bilanci dell'istituzione analizzata.

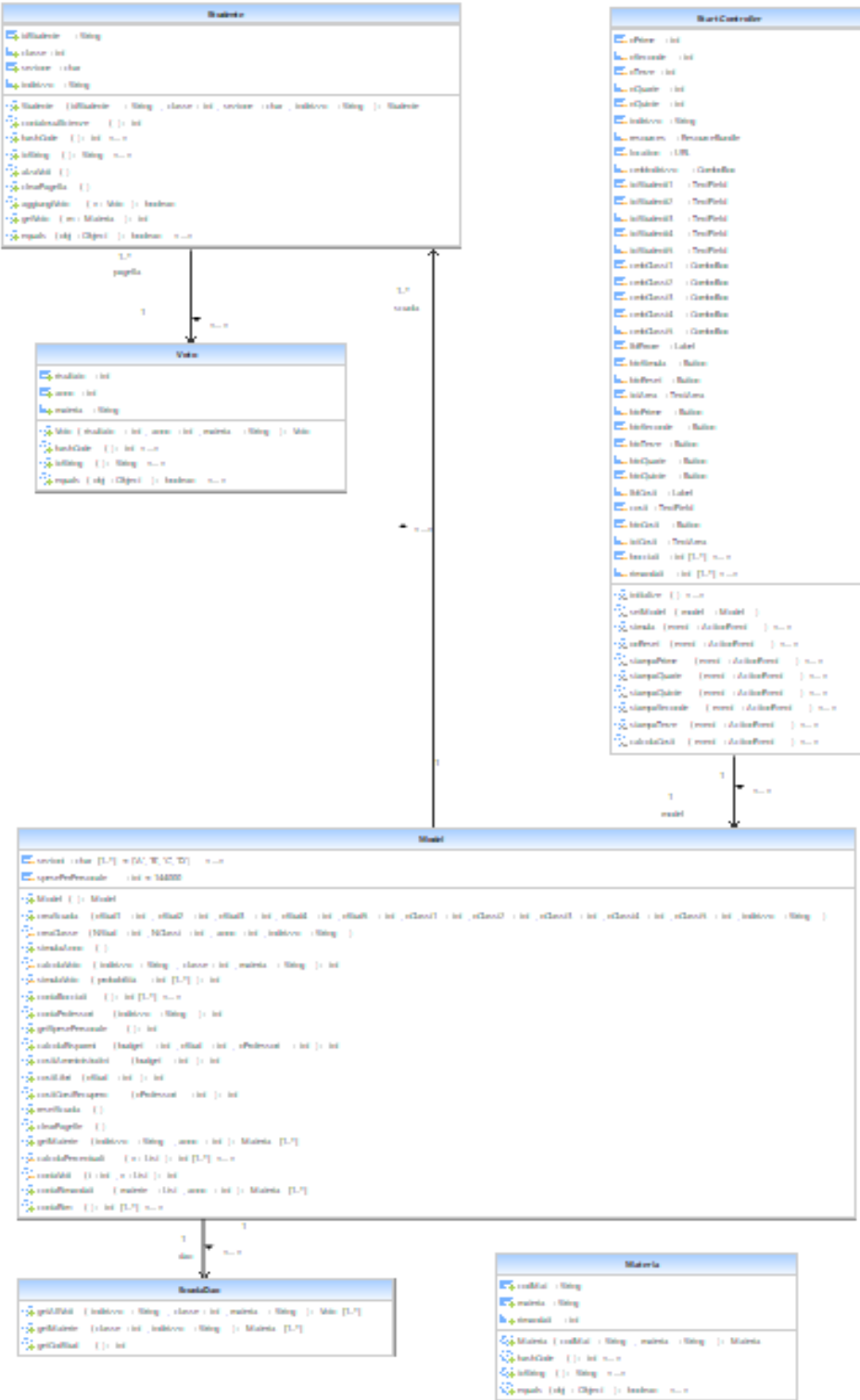
```
public int calcolaRisparmi(int budget, int nStud, int nProfessori) {
    return budget - this.spesePerPersonale - costiAmministrativi(budget) - costiLibri(nStud) - costiCorsiRecupero(nProfessori);
}

public int costiAmministrativi(int budget) {
    return (budget/100)*20;
}

public int costiLibri(int nStud) {
    return 100*nStud;
}

public int costiCorsiRecupero(int nProfessori) {
    return nProfessori*8*25;
}
```


Diagramma delle classi principali



Capitolo 6:

Videate dell'applicazione e collegamento al video

6.1 Videate dell'applicazione

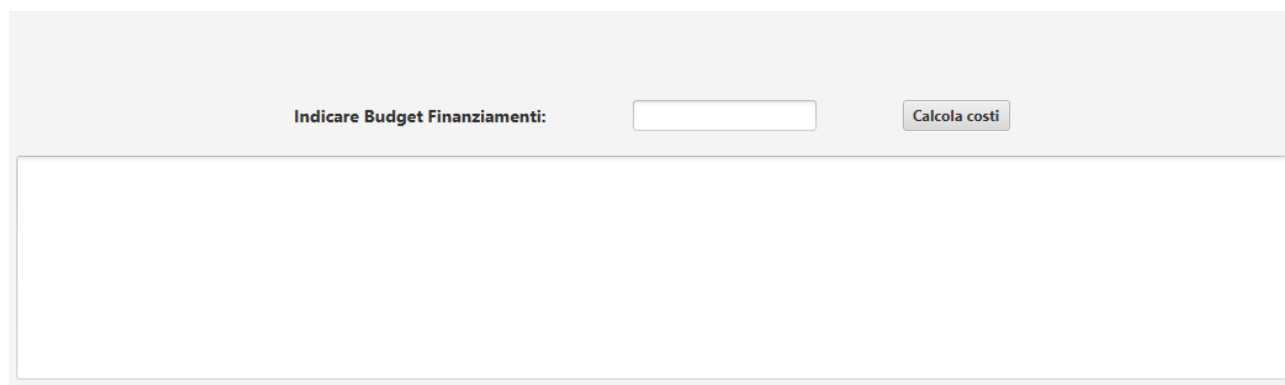
L'applicazione si presenta in questo modo:

The screenshot shows a web application titled "SIMULAZIONE ANNO SCOLASTICO". It features a form with the following elements:

- Selezione indirizzo:** A dropdown menu labeled "Indirizzo:".
- Inserire il numero di studenti per ogni anno:** Five input fields labeled "Prima:", "Seconda:", "Terza:", "Quarta:", and "Quinta:".
- Selezione il numero di classi per ogni anno:** Five dropdown menus labeled "Prima:", "Seconda:", "Terza:", "Quarta:", and "Quinta:".
- Buttons:** "Simula" and "Reset" buttons.
- Output Area:** A large empty rectangular box for displaying results.
- Footer Buttons:** "Prime", "Seconde", "Terze", "Quarte", and "Quinte" buttons.

In questa schermata è possibile selezionare uno dei tre indirizzi presenti all'interno dell'istituzione, inserire il numero di alunni per ogni anno e selezionare il numero di classi presenti per ogni anno. Se uno di questi campi non viene selezionato o riempito l'applicazione non farà partire la simulazione ma stamperà vari messaggi di errore. Se tutti i parametri sono riempiti correttamente sarà possibile iniziare la simulazione cliccando il bottone "Simula", quest'azione sbloccherà immediatamente il pulsante "Reset", per resettare tutti i campi e ritornare alla situazione di partenza, e i pulsanti "Prime", "Seconde", "Terze", "Quarte", "Quinte", che permetteranno di stampare nell'area di testo le pagelle di tutti gli alunni, oltre al numero di studenti bocciati e rimandati.

Infine, il pulsante “Reset” farà comparire una nuova parte di schermata:

A screenshot of a web form. At the top, there is a label "Indicare Budget Finanziamenti:" followed by a text input field. To the right of the input field is a button labeled "Calcola costi". Below these elements is a large, empty rectangular box, likely intended for displaying a summary or results.

Qui sarà possibile inserire la cifra corrispondente ai finanziamenti ricevuti e calcolare tutte le spese che la scuola dovrà sostenere. Un riassunto delle spese e i fondi rimanenti destinati ai progetti scolastici verranno stampati nell’area di testo sottostante.

6.2 Video dimostrativo

È possibile visionare un video dimostrativo del funzionamento dell’applicazione al seguente indirizzo:

<https://youtu.be/GMj69clfkMY>

Capitolo 7:

Risultati sperimentali

Ecco qui riportati i dati di una simulazione:

Matricola	Classe	DEE/AFM	EA/AFM	FRA/AFM	GEO/AFM	INF/AFM	ING/AFM	ITA/AFM	MAT/AFM	SIF/AFM	SITB/AFM	STO/AFM
STU1	1A	3	3	2	7	6	7	6	9	8	6	6
STU2	1A	7	7	5	6	4	8	6	6	4	6	7
STU3	1A	7	8	8	9	4	6	9	6	4	7	6
STU4	1A	8	6	7	8	7	4	4	6	6	8	5
STU5	1A	5	7	8	8	6	6	8	2	7	6	8
STU6	1A	6	5	6	6	8	6	6	8	6	6	6
STU7	1A	6	7	3	9	6	8	4	7	8	6	1
STU8	1A	8	5	7	6	7	6	6	5	4	7	4
STU9	1A	6	6	6	7	6	6	5	7	6	6	6
STU10	1A	7	6	6	7	6	5	6	6	7	5	6

Sono stati bocciati 1 studenti
Sono stati rimandati 9 studenti

Matricola	Classe	DEE/AFM	EA/AFM	FRA/AFM	GEO/AFM	INF/AFM	ING/AFM	ITA/AFM	MAT/AFM	SIC/AFM	SITB/AFM	STO/AFM
STU11	2A	8	6	7	6	6	7	6	6	6	6	6
STU12	2A	8	7	6	6	7	8	7	7	6	6	6
STU13	2A	7	6	7	7	7	7	5	4	7	6	7
STU14	2A	6	6	6	7	5	7	6	8	6	8	6
STU15	2A	6	7	7	6	7	7	6	7	6	6	6
STU16	2A	6	6	5	6	7	7	7	6	7	5	7
STU17	2A	7	3	6	7	7	7	7	6	6	6	6
STU18	2A	8	6	6	7	8	7	6	8	6	4	6
STU19	2A	7	9	6	7	6	6	6	6	8	6	7
STU20	2A	8	6	6	6	8	6	6	6	7	6	7

Sono stati bocciati 0 studenti
Sono stati rimandati 5 studenti

Matricola	Classe	DIR/AFM	EA/AFM	EP/AFM	FRA/AFM	INF/AFM	ING/AFM	ITA/AFM	MAT/AFM	STO/AFM
STU21	3A	6	6	7	6	7	6	6	9	7
STU22	3A	7	8	6	6	6	8	7	8	6
STU23	3A	8	6	7	6	7	7	6	7	6
STU24	3A	6	6	5	6	8	7	6	7	7
STU25	3A	6	6	7	6	7	7	6	6	6
STU26	3A	8	6	7	7	7	6	7	6	6
STU27	3A	6	6	7	6	6	6	7	6	7
STU28	3A	6	6	4	6	7	7	9	6	7
STU29	3A	7	6	7	7	8	6	8	6	8
STU30	3A	6	6	7	6	9	5	6	6	7

Sono stati bocciati 0 studenti
Sono stati rimandati 3 studenti

Matricola	Classe	DIR/AFM	EA/AFM	EP/AFM	FRA/AFM	INF/AFM	ING/AFM	ITA/AFM	MAT/AFM	STO/AFM
STU31	4A	5	5	7	7	8	6	7	4	7
STU32	4A	7	9	7	7	7	8	6	8	9
STU33	4A	6	7	9	7	8	6	7	6	6
STU34	4A	6	4	7	6	8	6	6	6	6
STU35	4A	8	8	8	6	7	6	6	6	6
STU36	4A	6	7	6	7	8	6	7	7	7
STU37	4A	6	6	6	6	8	8	8	7	7
STU38	4A	6	6	7	6	7	6	6	8	7
STU39	4A	7	8	7	7	7	7	7	6	8
STU40	4A	6	7	8	6	7	7	6	7	9

Sono stati bocciati 0 studenti

Sono stati rimandati 2 studenti

Matricola	Classe	DIR/AFM	EA/AFM	EP/AFM	FRA/AFM	ING/AFM	ITA/AFM	MAT/AFM	STO/AFM
STU41	5A	7	6	7	6	5	7	9	6
STU42	5A	7	9	6	6	6	6	6	7
STU43	5A	7	6	7	8	6	6	6	7
STU44	5A	7	6	9	6	6	7	7	6
STU45	5A	5	6	6	6	7	7	7	6
STU46	5A	7	8	6	6	6	7	6	6
STU47	5A	6	7	6	6	7	8	8	9
STU48	5A	8	6	7	6	6	7	7	6
STU49	5A	6	6	6	6	7	7	7	7
STU50	5A	8	6	7	6	7	5	7	8

Sono stati bocciati 0 studenti

Sono stati ammessi all'esame con insufficienze 3 studenti

ENTRATE

Finanziamenti: 300000

USCITE

Costi Amministrativi: 60000

Spese per il personale: 144000

Spese per libri di testo: 5000

Spese per corsi di recupero: 0

Totale uscite: 209000

I finanziamenti inseriti coprono i costi della scuola, il bilancio è in positivo di 91000€.

Questi soldi possono essere tutti usati per organizzare progetti scolastici.

Capitolo 8:

Conclusione e valutazione dei risultati ottenuti

Premettendo che è molto complicato riuscire a simulare il comportamento di un gruppo di persone, ritengo che i risultati forniti dall'applicazione siano molto vicini alla realtà dei fatti. Infatti, essi vengono stimati tenendo conto dell'andamento generale che hanno avuto gli studenti all'interno dell'istituzione nel corso degli anni.

Il punto di forza del software è sicuramente la sua capacità di adattamento, esso può infatti essere usato da qualsiasi scuola, apportando delle leggere modifiche al codice e sostituendo il database con quello della nuova istituzione. Questo permette una distribuzione del software su larga scala e lo rende utilizzabile da tutte le scuole, qualunque sia il loro livello.

Alcune criticità del software le possiamo trovare nella parte di calcolo delle spese. Infatti, il modello creato e utilizzato per il calcolo si basa sui bilanci dell'istituzione presso cui ho lavorato e sicuramente ogni scuola ha spese diverse. Un esempio possono essere le spese per il personale, che il software ha impostate come costanti, ma esse cambiano sicuramente da una scuola all'altra. Altro esempio possono essere le spese per i libri, che dipendono sicuramente dal numero di studenti iscritti, e questo l'applicazione lo calcola, ma anche dai tipi di libri che servono nelle differenti istituzioni.

In conclusione, possiamo dire che l'applicazione è un ottimo mezzo di supporto per i dirigenti scolastici per prevedere l'andamento dell'anno e le spese da sostenere, ma ha i suoi limiti dovuti principalmente alla difficoltà nel prevedere il comportamento di centinaia di studenti.

