****

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Documentazione**  Easy Pass   |  |  | | --- | --- | |  |  | | Versione | 1.0 | | Data | 14/02/2022 | | Destinatario | Prof. F. Palomba | | Presentato da | Montefusco Alberto  Rinaldi Viviana  Spina Gennaro | |  |  | |

Sommario

[**Sommario** 2](#_Toc530825397)

[**Team members**](#_Toc530825396) 3

[**Repository GitHub**](#_Toc530825396) 3

1. [**Introduzione**](#_Toc530825398) 4

1.1 Sistema corrente 5

1.2 Sistema proposto 5

1. [**Definizione del problema**](#_Toc530825398) 6

2.1 Obiettivi 7

2.2 Specifica PEAS 7

2.2.1 Caratteristiche dell'ambiente 8

2.3 Analisi del problema ........................................................................................................................... 8

**3.**   **Algoritmo Genetico** ................................................................................................................................... 10

3.1 Parametri …........................................................................................................................................ 11

3.1.1 Encoding degli individui ..................................................................................................... 11

3.1.2 Selezione ................................................................................................................................ 11

3.1.3 Crossover .............................................................................................................................. 11

3.1.4 Mutazione ............................................................................................................................. 12

3.1.5 Vincoli ................................................................................................................................... 12

3.1.6 Calcolo dei conflitti ............................................................................................................. 12

3.1.7 Funzione di fitness .............................................................................................................. 13

3.1.8 Criteri di arresto ................................................................................................................... 14

3.1.7 Passi dell'algoritmo .............................................................................................................. 14

**4.** **Testing dei parametri scelti** ................................................................................................................... 16

4.1 Testing ................................................................................................................................................ 17

**5. Conclusione** ................................................................................................................................................. 22

5.1 Considerazioni finali ......................................................................................................................... 23

Team members

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome** | **Ruolo** | **Acronimo** | **Informazioni di contatto** |
| Montefusco Alberto | Team member | MA | a.montefusco28@studenti.unisa.it |
| Viviana Rinaldi | Team member | VR | v.rinaldi26@studenti.unisa.it |
| Spina Gennaro | Team member | SG | g.spina5@studenti.unisa.it |

Repository GitHub

|  |
| --- |
| **Link** |
| https://github.com/Alberto-00/EasyPass-AI |

**CAPITOLO 1**

|  |
| --- |
| **Introduzione** |

* 1. Sistema corrente

Per applicare le regole anti-Covid in ambito universitario, imposte dal Governo a seguito dell’emergenza sanitaria, è previsto un meccanismo di controllo del Green Pass degli Studenti presenti in aula, effettuato da parte dei Docenti che sostengono la lezione.

Al momento, il Docente è tenuto a spostarsi fra i banchi per raggiungere lo Studente che vuole controllare, comportando alcuni disagi, primo tra tutti il mancato rispetto delle distanze di sicurezza quando il Docente deve effettuare la scansione.

Inoltre, nelle aule sono presenti posti contrassegnati che lo studente può occupare in modo tale da rispettare il distanziamento; tuttavia, queste distanze non sono sempre rispettate.

* 1. Sistema proposto

Per eliminare i disagi esposti nel precedente paragrafo, Easy Pass viene sviluppato come una Web Application, accessibile da Internet, mirata all’informatizzazione della procedura stessa.

Il Sistema è basato sull’utilizzo di sessioni di validazione identificate da un codice QR che, una volta condiviso dal Docente (tramite il proiettore) e scansionato dagli Studenti, permetterà a questi ultimi di inserire il proprio Green Pass per sottoporlo alla verifica.

Easy Pass prevede anche un modulo di Intelligenza Artificiale che permette di ricercare i posti che gli studenti potranno occupare nell’aula indicata dal Docente, non solo rispettando il distanziamento tra questi, ma anche posizionando lo studente in un posto ritenuto migliore per seguire la lezione.

**Nota**: attenendoci alle norme universitarie, il metro di distanza sarà rispettato se uno studente non ha colleghi seduti in maniera a lui adiacente. Questo significa anche che la capienza massima dell’aula è del 50%.

**CAPITOLO 2**

|  |
| --- |
| **Definizione del problema** |

* 1. Obiettivi

Lo scopo del progetto è quello di realizzare un agente intelligente che sia in grado di trovare una soluzione ottimale tale da:

* disporre gli studenti in aula in modo da rispettare il metro di distanza tra due individui, imposto dalle norme sulla sicurezza del Ministero della Salute;
* scegliere un posto ritenuto migliore per la visualizzazione della lavagna o dello schermo proiettato.

Una volta realizzato l’agente intelligente, quest’ultimo dovrà essere integrato all’interno di Easy Pass.

* 1. Specifica PEAS

|  |  |
| --- | --- |
| ***PEAS*** | |
| **Performance** | La misura di **performance** dell’agente si basa sull’accuratezza di assegnare un posto ad uno studente cercando di soddisfare gli obiettivi dati (*già descritti nel paragrafo 2.1*). |
| **Enviroment** | L’**ambiente** dell’agente è formato dall’aula, i singoli posti e gli studenti. |
| **Actuators** | Gli **attuatori** consistono nel visualizzare la *seating map* tramite una mappa grafica per mostrare a video la disposizione consigliata agli studenti. |
| **Sensors** | I **sensori** tramite il quale l’agente reperisce gli stimoli dall’ambiente sono rappresentati da un form per richiedere la visualizzazione della *seating* *map,* dopo averla configurata tramite l’inserimento dell’aula desiderata e del numero di studenti. |

* + 1. Caratteristiche dell’ambiente

L’ambiente è:

* **Completamente osservabile**: l’agente ha sempre accesso a tutte le informazioni relative alla disposizione dei posti;
* **Deterministico**: lo stato successivo dell’ambiente è completamente determinato dallo stato corrente e dall’azione eseguita dall’agente;
* **Sequenziale**: la posizione che lo studente può occupare è influenzata dalla sequenza di posti occupati precedentemente;
* **Statico**: durante la ricerca dei posti, il Docente attende l’esecuzione dell’algoritmo;
* **Discreto**: l’insieme delle possibili percezioni e azioni dell’agente sono distinte e definite;
* **Agente singolo**: l’ambiente consente la presenza di un unico agente.
  1. Analisi del problema

**Formulazione a stati completi del problema**:

* **Stati**. Ogni disposizione nell’aula degli N studenti è uno stato.
* **Stato iniziale**. Aula con studenti già disposti.
* **Azioni**. Spostare gli studenti in modo che non ci siano conflitti e che il punteggio di fitness sia il più alto possibile.
* **Modello di transizione**. Restituisce l’aula con una nuova disposizione di studenti.
* **Test obiettivo**. Nell’aula sono disposti tutti gli studenti col minor numero di conflitti (laddove possibile, 0) e il valore di massimizzazione è il più alto.

**Scelta dell’algoritmo**. Formulato il problema, era chiara la necessità di utilizzare un algoritmo di ottimizzazione. In particolare, abbiamo implementato un algoritmo genetico multi-obiettivo in quanto gli scopi da raggiungere sono 2. È stato utilizzato NSGAII poiché tiene conto del trade-off tra i vari obiettivi contrastanti e, inoltre, è in grado di restituire un insieme di soluzioni non dominate in cui sono favoriti gli individui più vicini al vero Fronte di Pareto. Il nostro primo pensiero era stato di utilizzare NSGA nella sua versione primitiva; tuttavia, abbiamo scelto di utilizzare questa variante in quanto integra meccanismi quali l’elitismo e la crowding distance nella selezione degli individui restituendo soluzioni migliori.

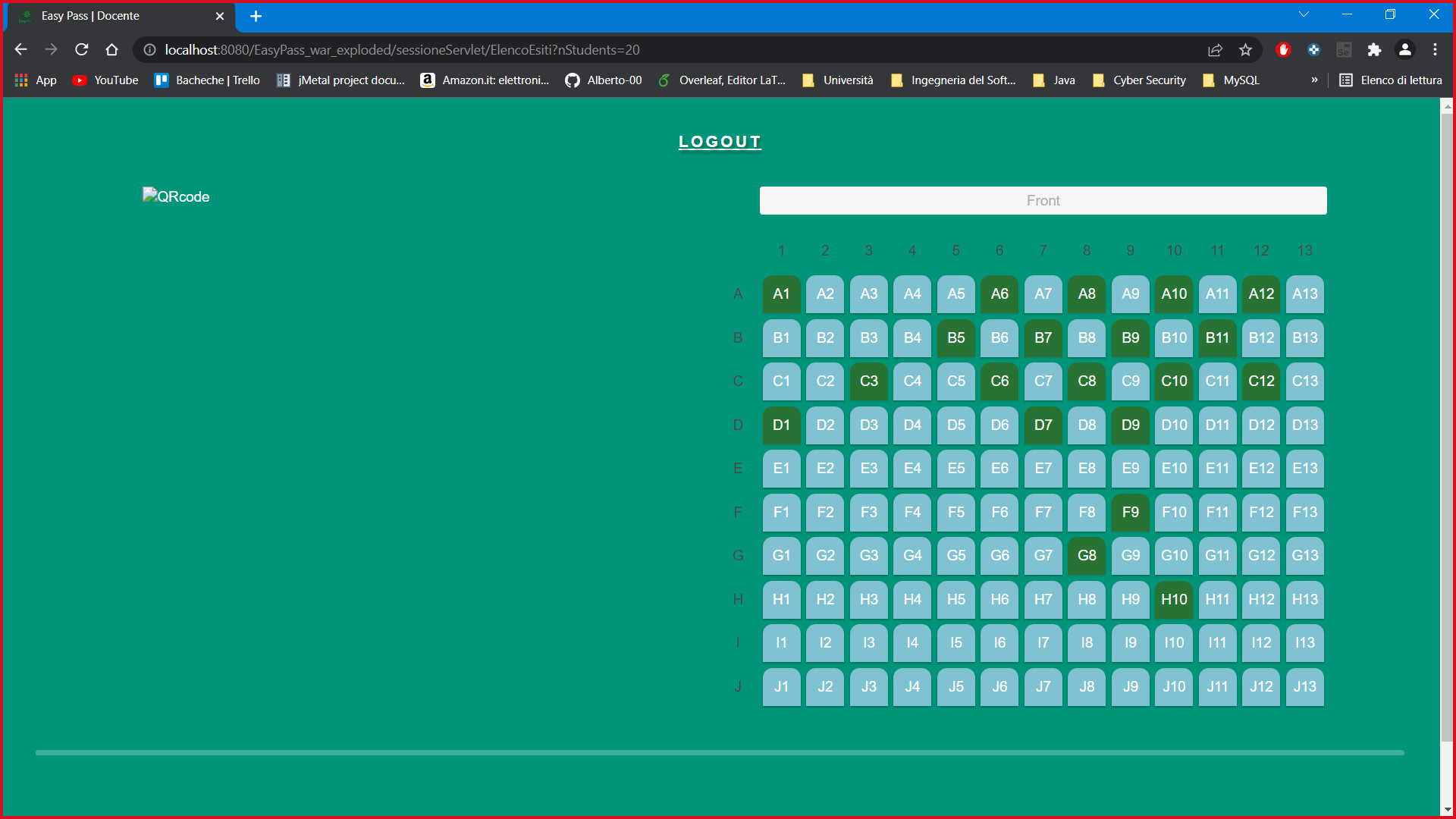
**Soluzione.**

La soluzione ottima è un individuo che ha 2 qualità:

* Tramite l’implementazione di una funzione di minimizzazione, si restituisce l’individuo col minor numero di conflitti;
* Tramite l’implementazione di una funzione di massimizzazione, avremo l’individuo con più alto punteggio di fitness, definito dalla posizione occupata in aula da ciascun studente.

**Esempio**:

* Aula 13 x 10;
* 20 studenti da disporre.



**CAPITOLO 3**

|  |
| --- |
| **Algoritmo Genetico** |

* 1. Parametri

Durante la progettazione dell’algoritmo genetico si è cercato un modo per stabilire la bontà di una

configurazione di parametri. Di seguito, vengono elencati i parametri utilizzati per il setup dell’algoritmo genetico multi-obiettivo.

* + 1. Encoding degli individui

Un individuo è una disposizione di N studenti in un’aula di posti (dove rappresenta il numero di righe e il numero di colonne).

La codifica utilizzata è una codifica reale in quanto più compatta e più vicina al problema in questione (poiché prende degli input numerici).

* + 1. Selezione

Per la selezione degli individui migliori si è scelto di utilizzare la strategia “Binary Tournament” che confronta due individui sulla base di due fattori: la crowding distance e il rango. Ciò accade per favorire da una parte soluzioni non troppo simili (che porterebbero altrimenti ad una convergenza prematura e a una popolazione finale poco migliorata) e dall’altra per premiare le soluzioni che hanno un maggiore punteggio di fitness. Quest’ultimo concetto è rafforzato dall’utilizzo dell’elitismo.

* + 1. Crossover

La strategia adottata è stata “K-Point Crossover” che consiste nel dividere due individui di una stessa generazione in due parti (questo è valido nel nostro caso in quanto ) in un punto randomico e scambiarle tra di loro per aumentare la diversità della popolazione. La probabilità che avvenga il crossover è 0.8%.

* + 1. Mutazione

La “Polynomial Mutation” riguarda i singoli geni di ciascun individuo, il cui valore muterà in un valore randomico, generato mediante opportuni calcoli, secondo una probabilità di mutazione dello 0.01%.

* + 1. Vincoli

Oltre ad indicare il limite superiore e inferiore di ogni variabile, è stato imposto un altro vincolo per prevenire che l’algoritmo posizioni uno studente in un posto già assegnato ad altri (in altri termini, non possono essere generate all’interno di uno stesso individuo due coppie di variabili uguali).

* + 1. Calcolo dei conflitti

Nel contesto del nostro problema, i conflitti vengono rilevati ogni qualvolta due studenti sono seduti a meno di un metro di distanza. Ossia, ciò accade se i due studenti sono seduti direttamente uno dietro l’altro o di fianco. Non rileviamo conflitti se sono seduti sulla stessa diagonale.

Immagine che contiene testo, screenshot, elettronico, computer

Descrizione generata automaticamente

* + 1. Funzione di fitness

La funzione di fitness è implementata tramite una funzione di massimizzazione in cui a ogni studente viene associato un punteggio a seconda dell’ottimalità del settore in cui si trova. Sommando i punteggi così ottenuti, si ha il valore della funzione di fitness di quella disposizione (individuo).

Nel nostro problema, abbiamo deciso di suddividere l’aula in 9 settori ognuno dei quali ha un punteggio in un range da 1 a 4, definito dalla sua posizione rispetto alla lavagna. In particolare, ogni studente è rappresentato da un gene, il quale indica le coordinate del posto da lui occupato.

**Divisione in settori**.

Ogni aula analizzata è divisa in 9 settori. Un settore è una matrice contenente due coppie di coordinate (che indicano rispettivamente lo spigolo in alto a sinistra, in cui inizia il settore, e lo spigolo in basso a destra, in cui termina il settore) e il punteggio del settore.

Per calcolare le dimensioni di ogni settore verifichiamo:

* se il modulo prodotto dal rapporto tra il numero di righe della matrice per 3 è diverso da 0, allora i primi due settori avranno **numero di righe = (righe della matrice / 3) + 1** mentre il terzo settore avrà il restante numero di righe;
* altrimenti, il **numero di righe del settore = (righe della matrice / 3)**.

In modo analogo calcoliamo il numero di colonne per ogni settore.

**Assegnazione punteggio al settore**.

La distribuzione dei punteggi avviene secondo uno schema a “V” utilizzato anche in altri problemi di assegnazione dei posti (quali teatri, cinema, etc.):

**Settore 1**: punteggio **3**;

**Settore 2**: punteggio **4**;

**Settore 3**: punteggio **3**;

**Settore 4**: punteggio **2**;

**Settore 5**: punteggio **3**;

**Settore 6**: punteggio **2**;

**Settore 7**: punteggio **1**;

**Settore 8**: punteggio **2**;

**Settore 9**: punteggio **1**;

**Esempio divisione in settori**.

Consideriamo un’aula formata da 7 righe e 8 colonne. La divisione in settori, con il relativo punteggio, sarà strutturata in questo modo:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  | **3** |  |  | **4** |  | **3** |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  | **2** |  |  | **3** |  | **2** |  |
|  |  |  |  |  |  |  |  |
|  | **1** |  |  | **2** |  |  | **1** |

* + 1. Criteri di arresto

L’algoritmo si arresta nel momento in cui ha eseguito la funzione di valutazione 1000 volte per ogni disposizione da analizzare.

* + 1. Passi dell’algoritmo

**Inizializzazione**: inanzitutto vengono impostati gli obiettivi, i vincoli del problema, la taglia della popolazione, il numero di valutazioni e le probabilità di crossover e mutazione; inoltre, sono indicati anche i bound di ogni variabile. Viene quindi generata una disposizione randomica di studenti in un’aula con m righe e n colonne.

**Valutazione popolazione**: generata la popolazione, viene effettuato il calcolo dei conflitti e della funzione di fitness. Viene quindi controllato il numero di valutazioni e, se questo risulta essere inferiore al numero di valutazioni, si accederà al ciclo per migliorare ancora la popolazione.

**Selezione**: tramite la Binary Tournament Selection otteniamo un mating pool di 100 individui, in cui una parte si è direttamente *salvata* grazie all’elitismo mentre un’altra ha dovuto superare la selezione basata su rango e crowding distance.

**Crossover**: dal mating pool, un numero casuale di individui viene combinato a coppie per aumentare la diversità genetica tramite 1-Point Crossover.

**Mutazione**: quindi, per mezzo della Polynomial Mutation il valore di alcuni geni potrebbe essere cambiato per evitare la convergenza prematura.

**Aggiornamento della popolazione**: a questo punto, si è arrivati a una nuova generazione che viene nuovamente valutata e, se i criteri di arresto sono soddisfatti, viene restituita come risultato dell’algoritmo.

**Risultato**: nel nostro caso, ai fini del nostro problema, estrarremo dalla popolazione finale l’individuo migliore, valutando come tale quello che ha in assoluto il minor numero di conflitti e, successivamente, quello che ha il punteggio di fitness più alto. Qualora vi siano due soluzioni uguali, sarà restituita la prima riscontrata.

**CAPITOLO 4**

|  |
| --- |
| **Testing dei parametri scelti** |

* 1. Testing

I parametri a cui si fa riferimento, oltre quelli già indicati sono:

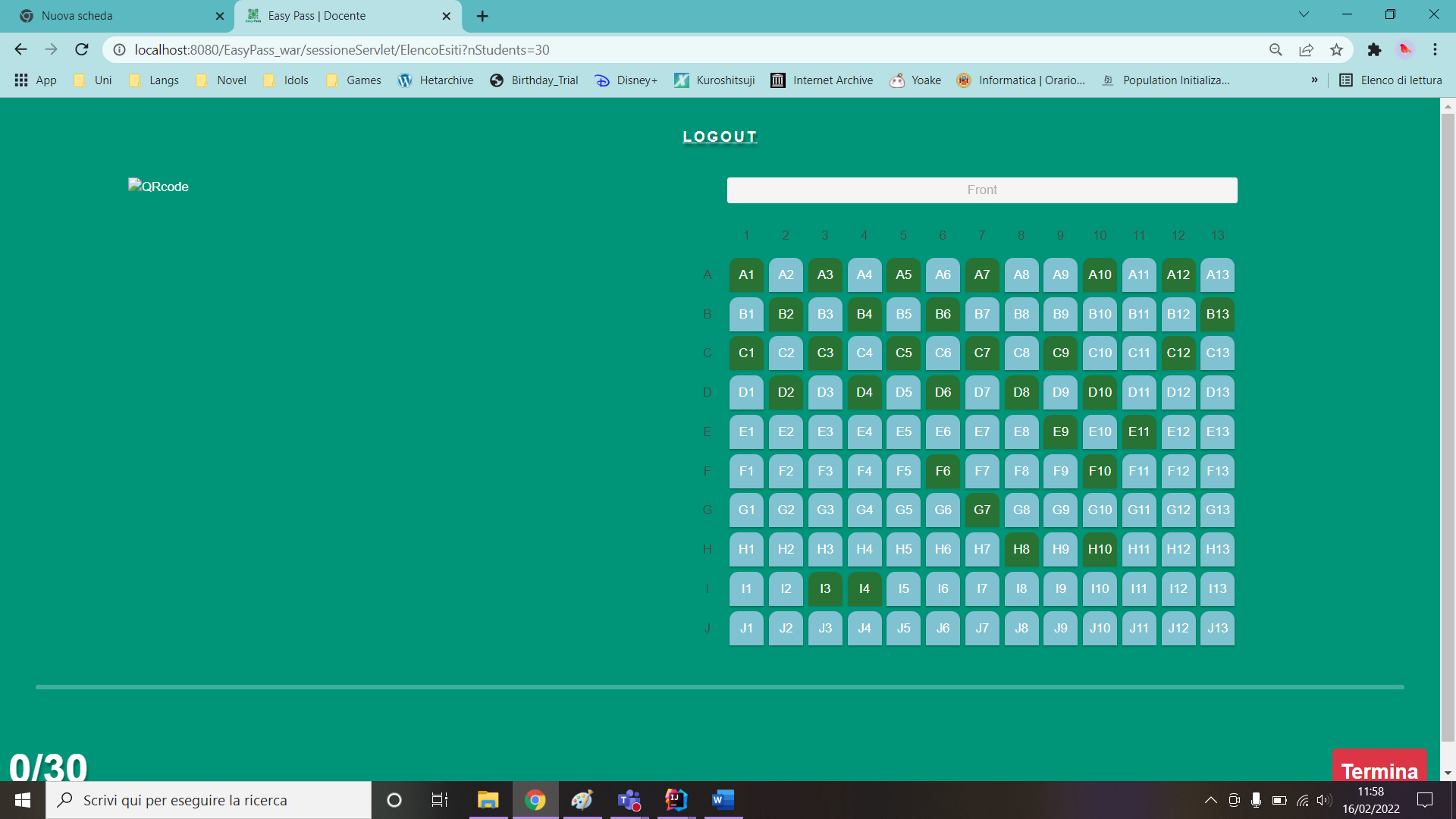
* dimensioni dell’aula: 10 x 13 (capienza dell’aula al 50% quindi 65 posti max occupabili);
* numero di valutazioni: 100.000 valutazioni.

1. **Caso**

Numero studenti: 30 Dimensioni popolazione: 20

**Immagine che contiene testo, monitor, computer, screenshot

Descrizione generata automaticamente**

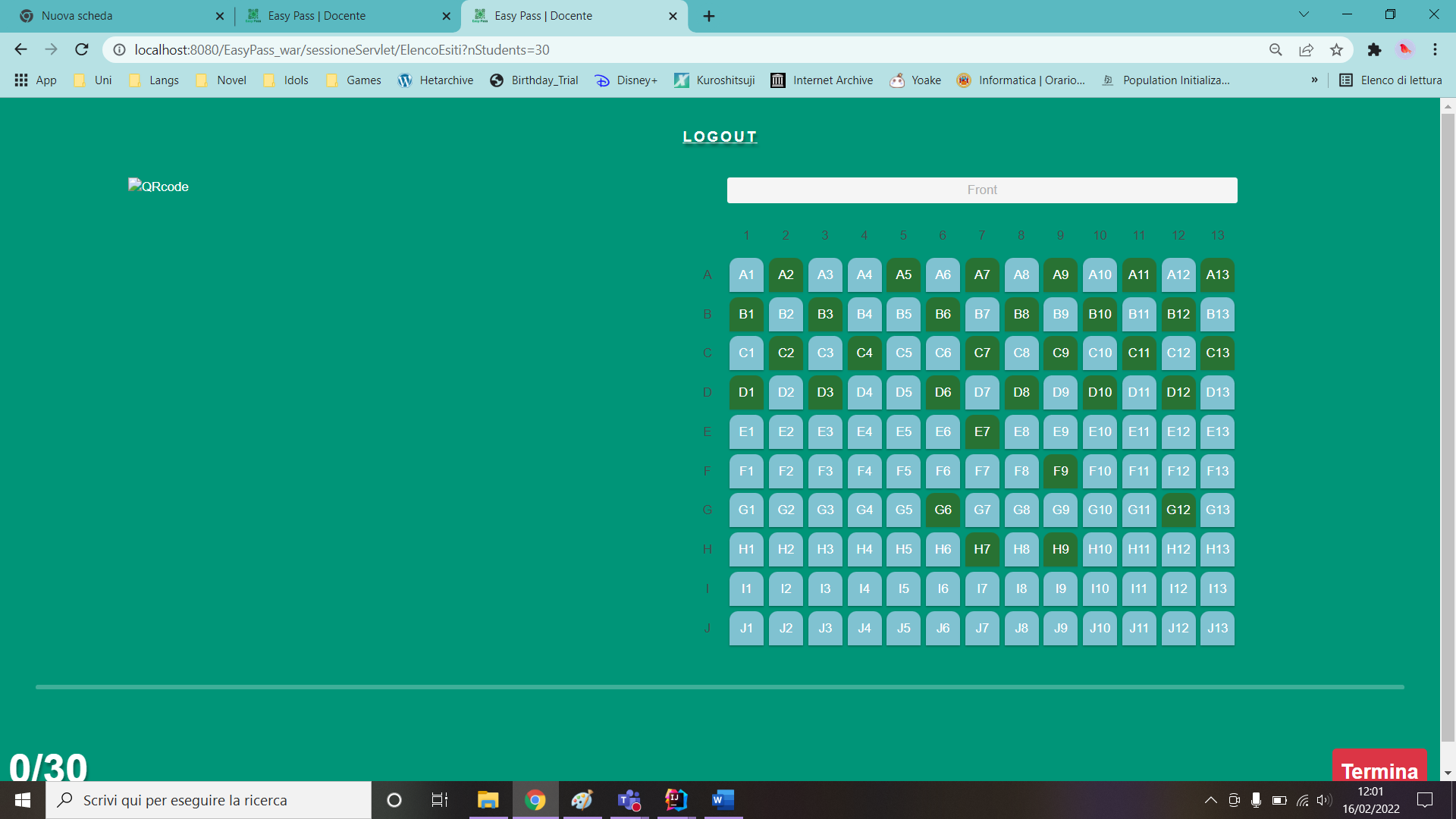


1. **Caso**

Numero studenti: 30 Dimensioni popolazione: 50

Immagine che contiene testo, monitor, computer, screenshot

Descrizione generata automaticamente



1. **Caso**

Numero studenti: 30 Dimensioni popolazione: 100

Immagine che contiene testo, monitor, computer, interni

Descrizione generata automaticamente



1. **Caso**

Numero studenti: 45 Dimensioni popolazione: 100

Immagine che contiene testo, tabellonesegnapunti

Descrizione generata automaticamente

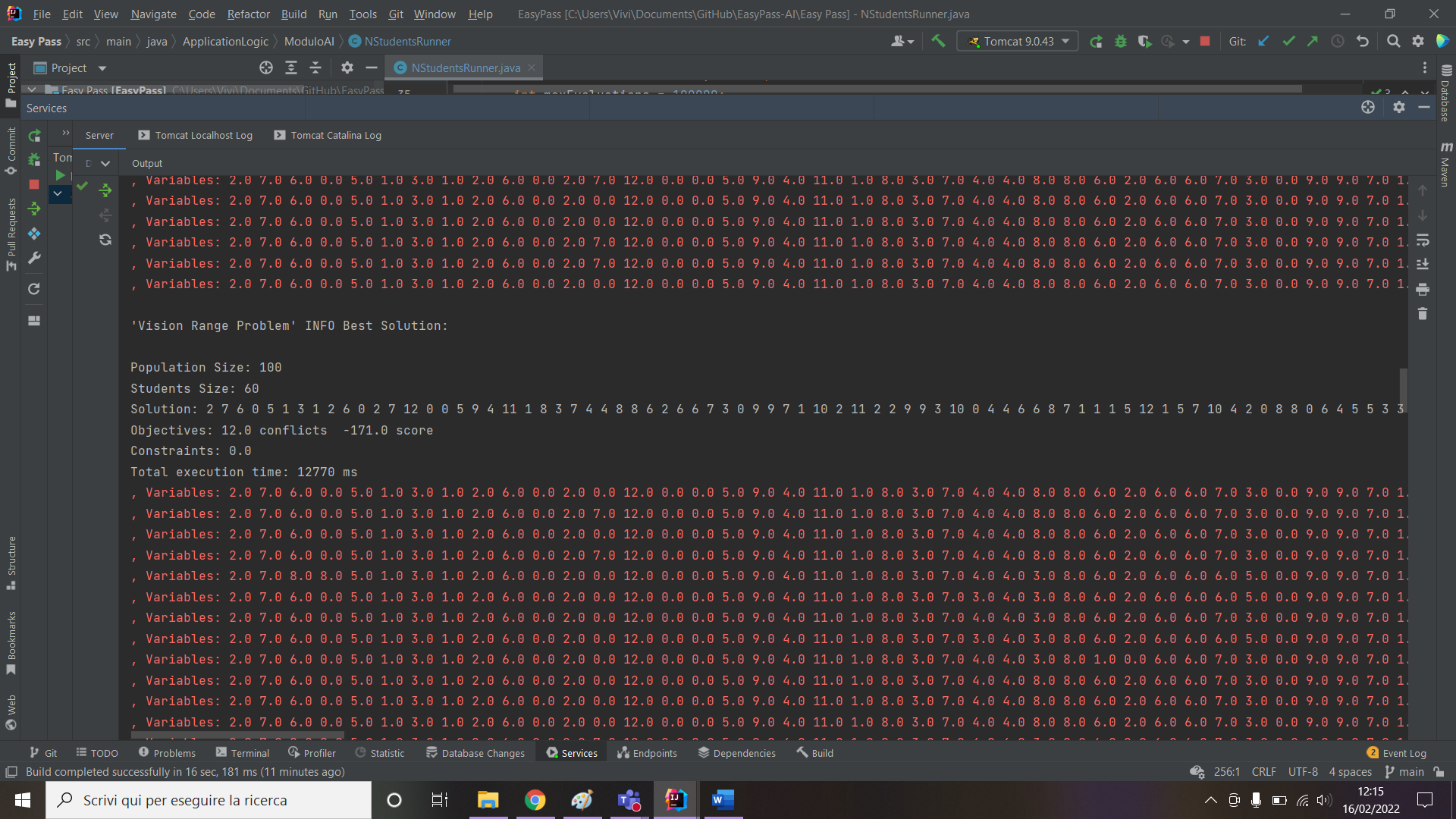
Immagine che contiene testo, computer, monitor, portatile

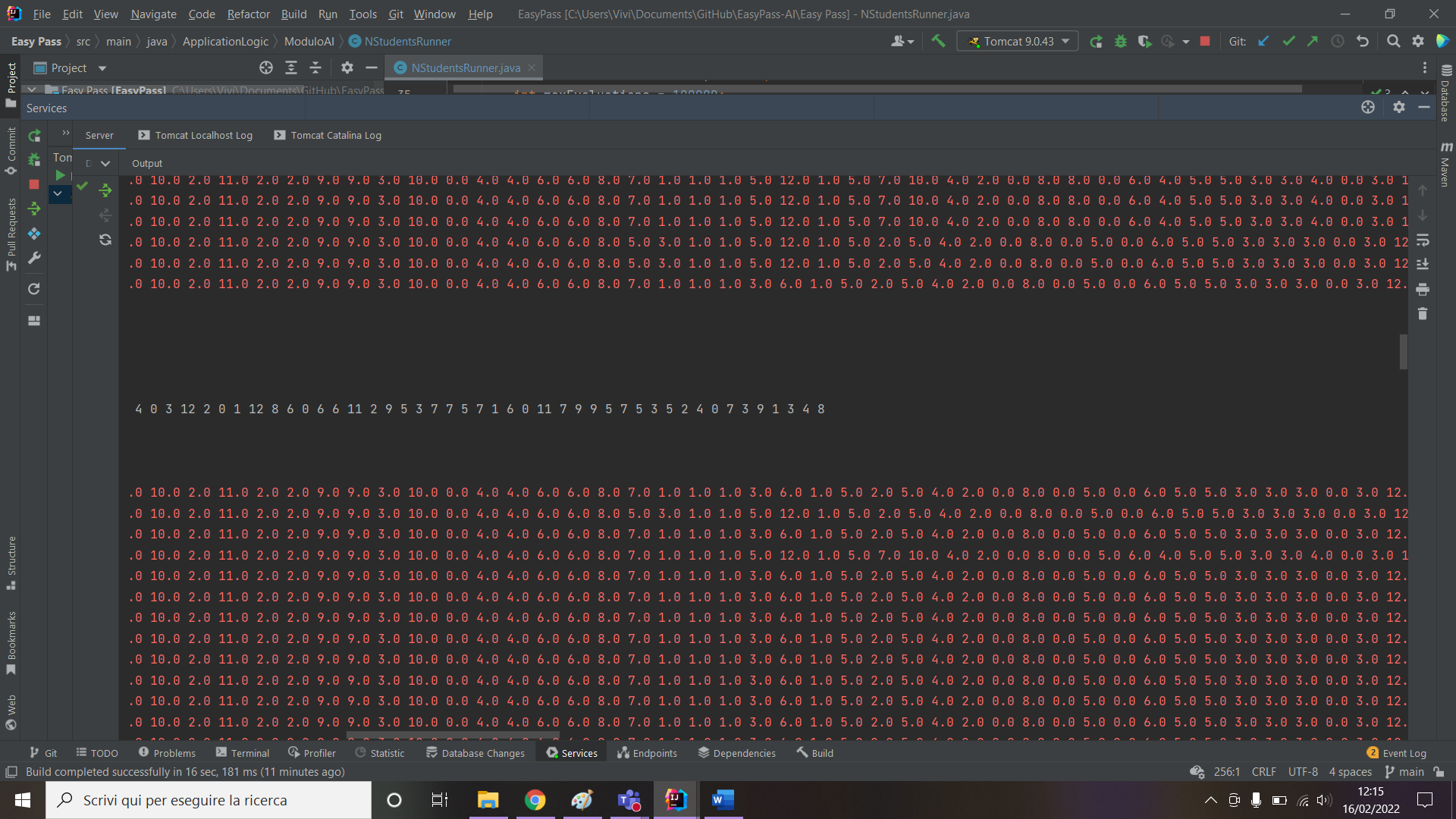
Descrizione generata automaticamente



1. **Caso**

Numero studenti: 60 Dimensioni popolazione: 100







**CAPITOLO 5**

|  |
| --- |
| **Conclusione** |

* 1. Considerazioni finali

Come si può notare dai test effettuati, risulta evidente che una popolazione più numerosa porti a soluzioni con un numero di conflitti inferiori seppur il tempo di elaborazione aumenti. D’altra parte, il numero di conflitti aumenta inevitabilmente con l’avvicinarsi del numero degli studenti alla capienza massima dell’aula.

Inoltre, non sono stati effettuati test considerando un numero più elevato di valutazioni a causa dell’aumento del tempo di esecuzione dell’algoritmo genetico.