UNIVERSITÀ DEGLI STUDI DI NAPOLI "PARTHENOPE"
FACOLTÀ DI SCIENZE E TECNOLOGIE

CORSO DI LAUREA IN INFORMATICA L-31

RELAZIONE DI
PROGRAMMAZIONE 3 E LABORATORIO DI PROGRAMMAZIONE 3

DOCENTI
Prof. Angelo Ciaramella
Prof. Raffaele Montella

CANDIDATI
Luca Esposito
matricola 0124001698
Vincenzo Marco De Luca
matricola 0124001523

ANNO ACCADEMICO 2019-2020

Indice

1. Descrizione e requisiti del progetto

Si vuole sviluppare un'applicazione di viaggi che consente di ricercare soluzioni catalogandole attraverso l'analisi dei feedback lasciati dagli utenti. In particolare, attraverso il Neural Language Processing (NLP), verrà effettuata un'analisi dei commenti degli utenti lasciate sulle varie piattaforme (ad esempio Booking e TripAdvisor) attraverso il Web Scraping.

A. All'avvio si visualizzerà la pagina di log-in:
   a. Se l'utente risulta essere registrato, accede attraverso e-mail e password
   b. Se l'utente non risulta essere registrato, procede con la registrazione fornendo:
      - Nome
      - Cognome
      - e-mail (credenziali accesso)
      - password (credenziali accesso)
      - Città

B. Una volta effettuato l'accesso, l'utente visualizzerà una bacheca principale da cui potrà selezionare diversi filtri (tra cui umore, città, tipo):
   a. Nella parte in alto a destra sarà presente il pulsante "Bacheca Utente"
      - L'utente potrà accedere alla propria bacheca in cui si terrà traccia di tutti i viaggi a cui ha partecipato.
   b. Nella sezione alta saranno presenti i vari filtri per poi poter avviare la ricerca:
      - Una volta inseriti i vari filtri, l'utente potrà premere sul pulsante "Cerca" attraverso cui verrà effettuata la ricerca in base ai filtri inseriti tenendo conto soprattutto dell'umore dell'utente in modo da presentare soluzioni con maggiore compatibilità.

- I filtri selezionabili saranno:
  - Umore
  - Città
  - Tipo
  - Prezzo
  - Numero di persone
  - Data Inizio
  - Data Fine

- Ogni elemento ricercato si comporrà delle seguenti informazioni:
  - Nome offerta
  - Compatibilità
  - Stato
  - Feedback
  - Data
  - Prezzo
  - Link
  - Salva

Il progetto è stato implementato attraverso le seguenti linee:

- Uso di un pattern tra quelli studiati a lezione.
- Rispetto di alcuni dei principi SOLID che ricordiamo essere i seguenti:
    1. Single Responsability Principle(SRP)
    2. Open-Closed Principle(OCP)
    3. Liskov Substitution Principle(LSP)
    4. Interface Segregation Principle(ISP)
    5. Dependency Inversione Principle(DIP)
- Uso dell'interfaccia grafica utilizzando JavaFX per permettere all'utilizzatore del software una più chiara visualizzazione.
- Uso di Database per gestire tutte le informazioni degli utenti e dei viaggi.

2. Interfaccia grafica e ambiente di lavoro

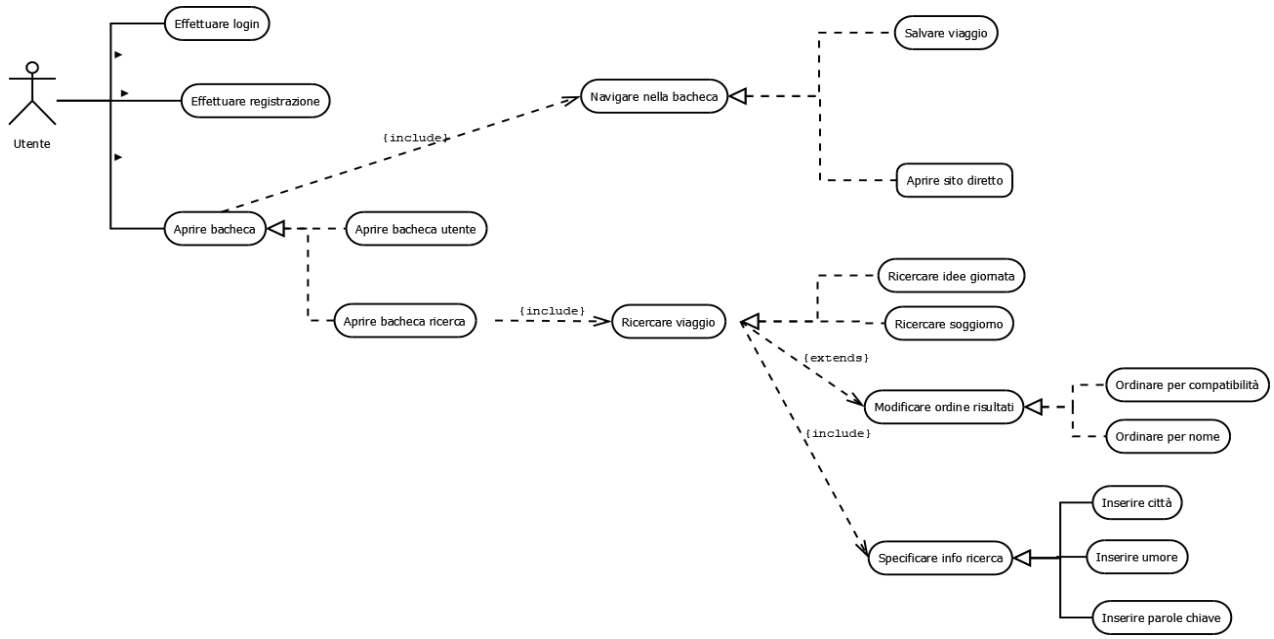
Il progetto è stato sviluppato in Java su ambiente Linux con l'utilizzo di IntelliJ, la piattaforma JavaFX per la parte grafica, la libreria Stanford coreNLP e database MySQL.

Le interfacce implementate sono:

- **Pagina di login**: consente l'accesso al sistema inserendo email e password. Nel momento in cui l'utente preme il tasto "Login" il sistema ne verificherà la presenza o meno all'interno del database confermando l'accesso o meno attraverso il cambio di stato di una label.
- **Pagina registrazione e ok**: consente la registrazione dell'utente attraverso l'inserimento di nome, cognome, email, password e città. Quando l'utente preme il tasto "Registrati" ne verrà verificata la presenza nel database e, in caso di riscontro negativo, verrà presentata la pagina di conferma registrazione.
- **Pagina bacheca principale**: consente all'utente di ricercare il viaggio attraverso l'inserimento dei filtri presenti sulla parte alta della pagina. Una volta inseriti tutti i filtri e premuto il tasto "Cerca" verrà avviato il Web Scraping con relativa interrogazione del Database da cui verranno estratte le soluzioni compatibili con i filtri dell'utente. Il modulo NLP verrà attivato per catalogare i commenti delle varie soluzioni presenti sul sito TripAdvisor in modo da rendere più "scientifica" la compatibilità e veridicità delle valutazioni.
  L'utente avrà la possibilità di accedere al Link dell'offerta ed eventualmente salvarla tramite l'apposito bottone.
- **Pagina bacheca utente**: è accessibile dalla bacheca principale e consente di avere uno storico dei viaggi salvati.

3. Use Case

L'attore principale del sistema è l'utente.



Esempio di utilizzo del sistema:

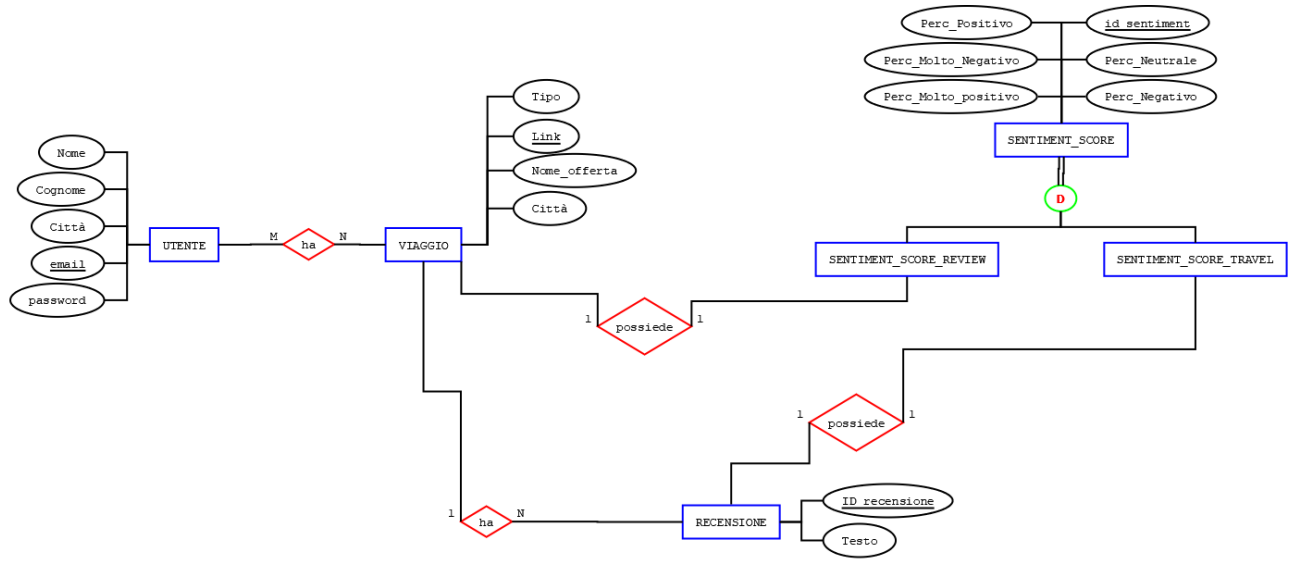
A. Un nuovo utente avvia l'applicazione ed effettua la Registrazione inserendo tutte le informazioni richieste.
   - Viene effettuata la query da parte del sistema che verifica la presenza o meno dell'utente, inserendolo o meno nel database.
B. L'utente esegue il Login inserendo le informazioni.
   - Viene effettuata una query che verifica la presenza o meno dell'utente facendolo proseguire o meno nel processo di Login.
C. A Login eseguito, l'utente avrà accesso alla Bacheca Principale da cui potrà:
   - Accedere alla Bacheca Utente
   - Inserire i parametri di ricerca per poi effettuare la ricerca che attiverà il Web Scraping e l'analisi attraverso coreNLP; in particolare, una volta premuto il tasto "Cerca":
     1. Il Web Scraping cercherà su TripAdvisor dei risultati compatibili caricando i vari viaggi nel database con le relative recensioni
     2. Stanford coreNLP si occuperà di analizzare le recensioni ed effettuerà un update al database per ogni viaggio caricando i dati relativi alla categorizzazione della compatibilità.
     3. Verranno presentati i dati all'utente all'interno di una tabella attraverso cui sarà possibile filtrare i risultati.
     4. La tabella presenterà due pulsanti: "Link" e "Salva":
        - Il pulsante "LINK" consente all'utente di andare direttamente sulla pagina del viaggio interessato.

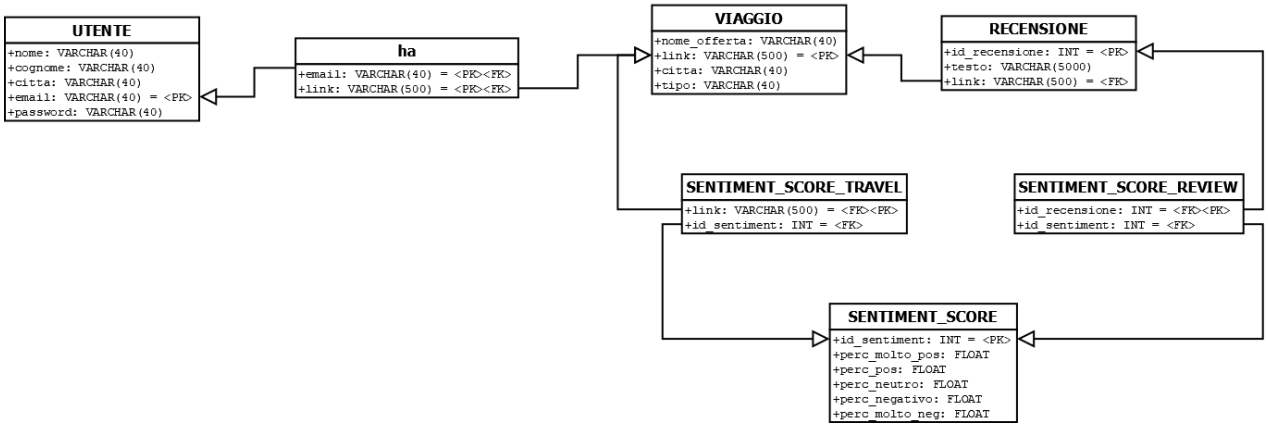
- Il pulsante “SALVA” consente all’utente di salvare il viaggio nella propria Bacheca Utente.

D. All’interno della Bacheca Principale, l’utente può accedere alla Bacheca Utente all’interno della quale avrà modo di avere uno storico dei viaggi salvati.

4. Database

Il database è stato sviluppato attraverso il relational database management Oracle MySQL ed è stato utilizzato attraverso Tomcat e il JDBC Driver.





Sono previste 5 entità:

- *Utente*: in questa entità troviamo le informazioni essenziali dell’utente che verranno memorizzate nel momento della Registrazione.
- *Viaggio*: in questa entità verranno inserite tutte le informazioni attraverso il Web Scraping. Quando l’utente avvia la ricerca, attiverà il Web Scraping che si interfaccerà con il database inserendo le soluzioni mancanti per poi dare all’utente una risposta aggiornata.
- *Recensione*: in questa entità troviamo le recensioni associate ad ogni singolo viaggio composte dal testo che verrà analizzato attraverso la libreria Stanford coreNLP.

- *Sentiment_Score_Travel*: conterrà le percentuali delle valutazioni dei viaggi. Ogni Sentiment Score Review sarà associato univocamente ad un solo viaggio.
- *Sentiment_Score_Review*: conterrà le percentuali associate ad ogni singola recensione. Ogni Sentiment Score Review è collegata ad una Recensione che sarà collegata al viaggio.

5. Diagramma delle classi

Il diagramma delle classi è stato riportato nella cartella “Diagramma Classi” ed è stato ottenuto sfruttando le funzionalità di Intellij.

Esso riporta tutte le dipendenze non sono fra classi ma anche fra i distinti package che sono stati creati durante lo sviluppo del progetto

6. Web Scraping

Il Web Scraping è una tecnica informatica di estrazione di dati da un sito web mediante un software. Si basa sull'estrazione di dati non strutturati in formato HTML presenti sul web, finalizzata a diversi scopi solitamente legati al campo del Data Science per ricavare statistiche mediante dei modelli di Machine Learning.

Nel nostro caso, utilizzando Java abbiamo optato per l'utilizzo di **HTMLUnit** che è un headless web browser che consente una manipolazione di alto livello dei siti web mediante software scritti in Java.

HTMLUnit simula il comportamento del browser includendo aspetti di basso livello legati a TCP/IP ed HTTP, consentendo la navigazione tramite hypertext e ricavando pagine web che includono HTML, JavaScript, Ajax e cookies.

L'algoritmo di WebScraping è basato sul download di informazioni da TripAdvisor, tuttavia essendo l'obiettivo dell'applicazione di analizzare quanti più dati possibili, l'obiettivo è di estendere il WebScraping ad altri siti per ottenere ulteriori informazioni relative a specifiche attività o ulteriori informazioni relative ai commenti rilasciati dagli utenti.

Per tale motivo, si è optato per l'utilizzo del pattern **Template Method** che momentaneamente include le classi WebScrapingTemplate e WebScrapingTripAdvisor ma per futuri sviluppi può comportare l'utilizzo di ulteriori estensioni di WebScrapingTemplate per sfruttare dati di altri siti con caratteristiche differenti e che richiederebbero l'override dei metodi richiamati dall'algoritmo di Web Scraping.

La classe WebScrapingTripAdvisor manipola il sottosistema relativo al sito web mediante FacadeTripAdvisor in modo tale da gestire con maggiore semplicità le operazioni di Web Scraping in quanto WebScrapingTripAdvisor si limita a chiamare le operazioni e FacadeTripAdvisor si incarica di chiamare i metodi corretti e specifici contenuti nel sottosistema del sito web.

Il sottosistema del sito web essendo molto variegato, ha richiesto l'utilizzo di molteplici classi e di differenti pattern finalizzati a semplificarne la struttura per evitare di generare ciò che Dijkstra chiamava "Spaghetti design". Per renderne maggiormente comprensibile la struttura sono stati utilizzati molteplici package, di conseguenza abbiamo utilizzato un approccio top-down per strutturarlo ed utilizzeremo lo stesso approccio per documentarlo.

La classe astratta Page indica una generica pagina web dotata di URL, di un client che la manipola e di un HTMLPage per gestirla, essa viene estesa da due sottoclassi che sono CityPage e TablePage.

- CityPage è una classe relativa alla pagina iniziale di una specifica città che contiene a sua volta una lista di pagine che da esse possono essere accessibili definite nella classe ListTableInterestPage.
- TablePage invece è una classe astratta relativa ad una pagina struttura come una tabella che può essere una tabella di attività come descritto dalla classe astratta TableInterestPage oppure una tabella di recensioni come descritto dalla classe astratta TableReviewInterestPage.

Sia TableInterestPage che TableReviewInterestPage possono essere relative ad alberghi, attrazioni o ristoranti come espresso rispettivamente dalle classi TableHotelPage, TableAttractionPage e TableRestaurantPage per quanto concerne la TableInterestPage opppure dalle classi TableReviewHotelPage, TableReviewAttractionPage e TableReviewRestaurantPage per quanto concerne la TableReviewInterestPage.

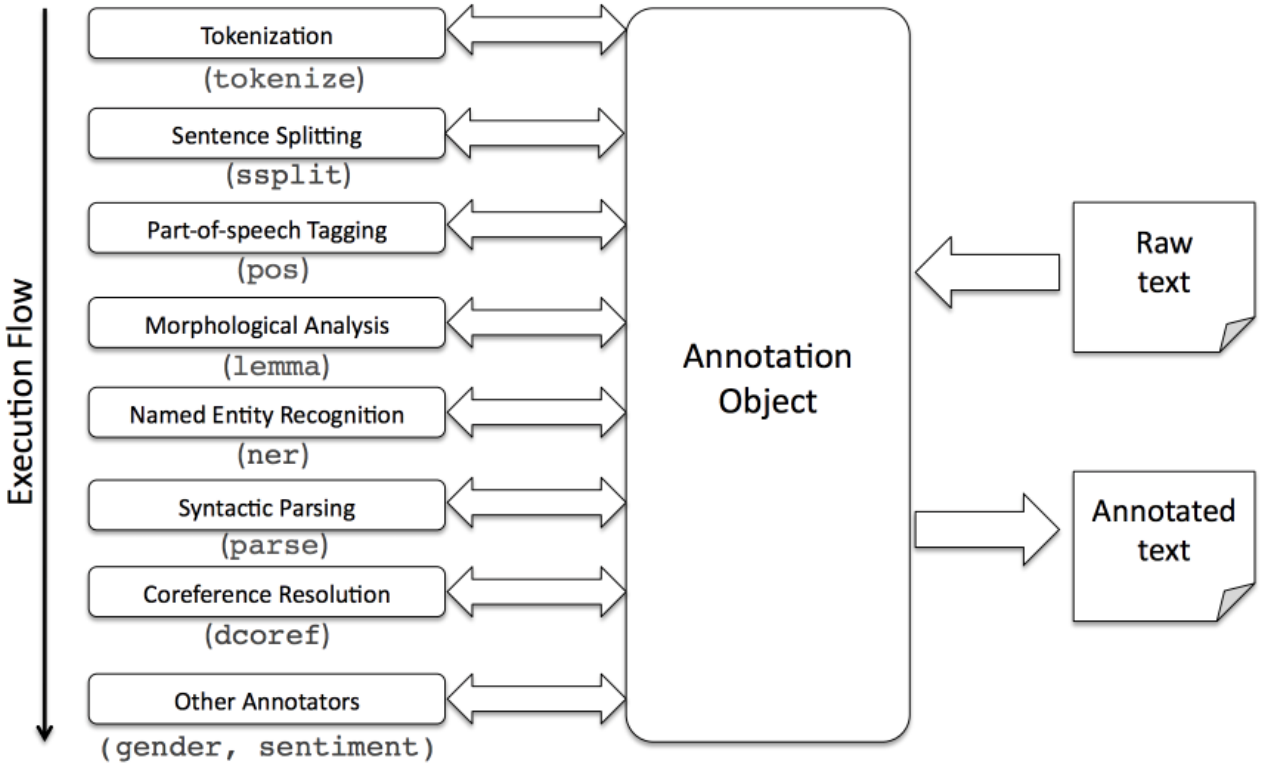
Essendoci un vincolo fra le sottoclassi di TableInterestPage e di TableReviewInterestPage, si è optato per l'utilizzo del pattern **Factory Method** in modo tale che un'istanza di TableHotelPage possa soltanto creare un'istanza di TableReviewHotelPage, un'istanza di TableAttractionPage possa soltanto creare un'istanza di TableReviewAttractionPage ed un'istanza di TableRestaurantPage possa soltanto creare un'istanza di TableReviewRestaurantPage.

Tuttavia, dato che le sottoclassi di TableInterestPage e le classi di TableReviewInterestPage indicano una singola pagina contenente rispettivamente molteplici attività o molteplici recensioni che però possono essere seguite da altre pagine con un'ulteriore specifico URL, sono state utilizzate due classi che consentono l'iterazione su tali pagine come descritto dal package TablePageList; inoltre essendo le TableReviewInterestPage caratterizzate dal fatto che contengano una lista di recensioni, esse contengono una lista di Review che indicano le singole review con annessi "SentimentClassification" e dato che tali informazioni vanno memorizzate in modo persistente sul database, si è optato per l'utilizzo del pattern Bridge che va a delegare a Bridge e le sue specializzazioni, la responsabilità dell'inserimento sul database.

Giungendo al punto più basso del WebScraping con la classe Review, è possibile ora trattare le tematiche relative al Natural Language Processing.

7. Natural Language Processing

Il Natural Language Processing è un campo del machine learning che consente ad un calcolatore di analizzare grandi quantitativi di dati in linguaggio naturale.



Nel nostro caso abbiamo optato per l'utilizzo della libreria di Stanford CoreNLP in quanto utilizzabile in Java ed in quanto forniva un supporto alla sentiment analysis. Per eseguire tale analisi, bisogna aggiungere fra gli annotators in fase di definizione un'ulteriore annotatore che è "sentiment".

Dato che tale libreria non forniva un training set per il modello in italiano, si è optato per l'utilizzo del modello relativo all'analisi di testi in inglese.

Mediante il costruttore si va a costruire un oggetto StanfordCoreNLP che richiede determinate proprietà specificate passando come argomento al costruttore un'oggetto Properties che viene richiede l'utilizzo di POS tagging, lemmatization, NER, parsing e sentiment.

L'analisi del testo viene eseguita mediante **AnalyzeText** che crea un Annotator che mediante l'utilizzo l'oggetto stanfordCoreNLP processa il testo. A questo punto viene avviato un ciclo for che richiede prima di inizializzare un albero in quanto la sentimentAnalyze si basa proprio sull'analisi logica del testo andando a suddividerlo su diversi livello e generando un ramo per distinto per ogni parte della frase che può essere composta da distinte parole.

Ogni parola ha un proprio valore ma la forza del Natural Language Processing non è legata all'analisi della singola parola ma al riconoscimento della suddivisione di un periodo sintattico mediante un'analisi logica del testo permettendo la contestualizzazione del testo.

In seguito bisogna inizializzare una **SimpleMatrix** che indica la Rete Neurale. Essendo ad ogni parola associato un punteggio fra 0 e 4 che indica se è molto negativa, negativa, neutrale, positiva e molto positiva, viene calcolata mediante un punteggio di tipo double limitato fra 0 e 1 contenuto nella SimpleMatrix inizializzata in precedenza, qual è la frazione di parti molto negative, negative, neutrali, positive e molto positive nel testo, pertanto basta moltiplicare per 100 per ricavarne la percentuale. Tale valore quindi consente di ricavare un punteggio esplicitamente calcolato sulle recensioni che fornisca una misura percentuale e maggiormente affidabile del grado di apprezzamento di un utente ad una specifica attività.

8. Rispetto dei principi SOLID e Pattern utilizzati

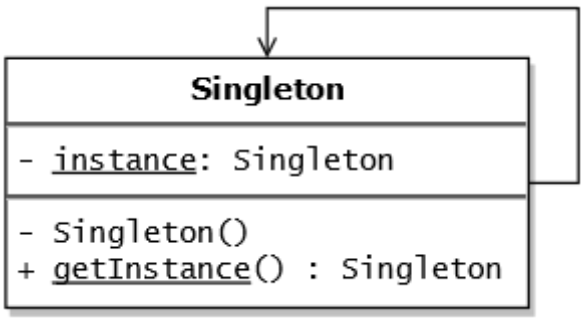
Per quanto riguarda i principi SOLID, ricordiamo che ci sono 5 principi che definiscono lo sviluppo di un buon software manutenibile ed estendibile che evita: viscosità, complessità inutile e ripetizioni inutili all'interno del codice.

1. *Single Responsability Principle* (SPR-Principio di Singola Responsabilità): dice che una classe dovrebbe avere un solo motivo per cambiare, o meglio ancora ogni elemento di un programma(classe, metodo, variabile) deve avere una sola responsabilità che deve essere incapsulata dall'elemento stesso.
2. *Open Closed Principle* (OCP-Principio Aperto/Chiuso): le entità software come classi, moduli e funzioni dovrebbero essere aperte per l'estensione, ma chiuse per le modifiche: ciò vuol dire che in ogni classe si deve garantire la sua estensione senza cambiare la classe stessa. Per il rispetto di questo principio, si devono usare classi astratte e classi concrete che ne implementino il comportamento.
3. *Liskov's Substitution Principle* (LSP-Principio di Sostituzione di Liskov): dice che se q(x) è una proprietà che si può dimostrare essere valida per oggetti x di tipo T, allora q(y) deve essere valida per oggetti y di tipo S dove S è un sottotipo di T. Dobbiamo essere sicuri che le classi derivate debbano essere capaci di rimpiazzare le classi base senza cambiamenti del codice.
4. *Interface Segregation Principle* (ISP-Principio di Segregazione delle Interfacce): dice che un client non dovrebbe dipendere da metodi che non usa. L'ISP preferisce l'uso di molte interfacce, specifiche e piccole piuttosto che poche, generali e grandi.
5. *Dependency Inversion Principle* (DIP-Principio di Inversione delle Dipendenze): i moduli di alto livello non dovrebbero dipendere dai moduli di basso livello. Entrambi dovrebbero dipendere dalle astrazioni. Le astrazioni non dovrebbero dipendere dai dettagli. I dettagli dovrebbero dipendere dalle astrazioni. Quindi le classi a alto livello non devono dipendere pesantemente dalle classi di basso livello.

Per quanto riguarda i Design Patterns ricordiamo che ne esistono 3 categorie principali:

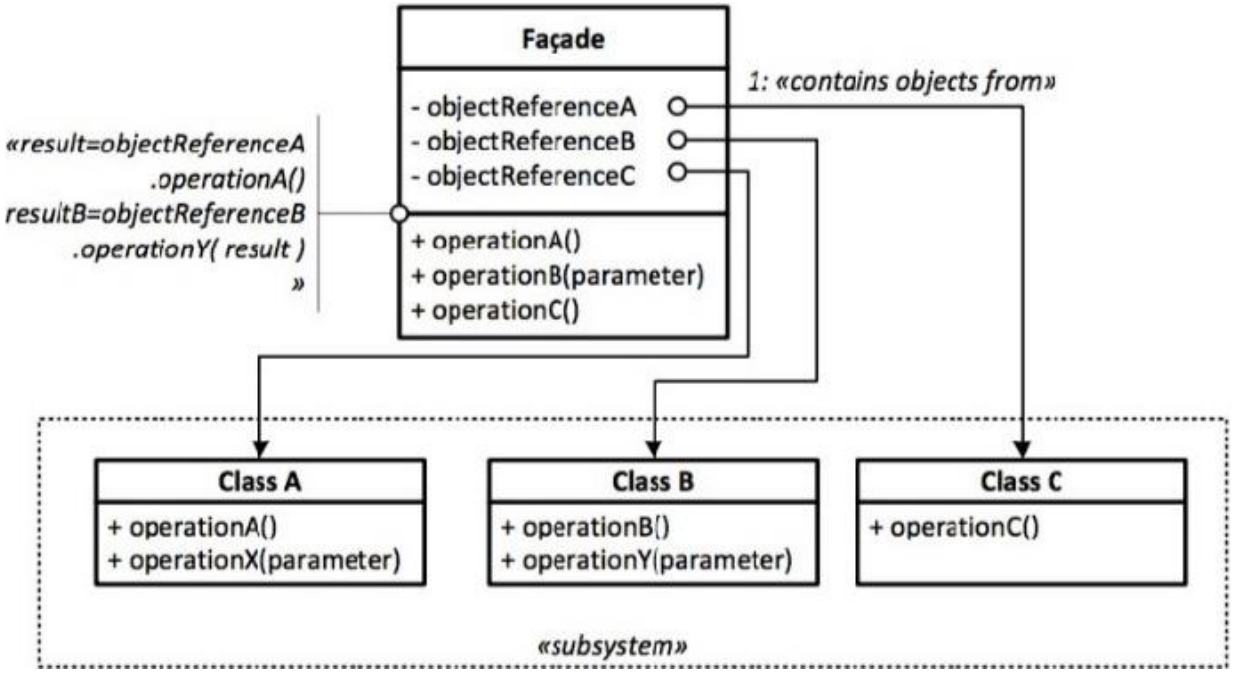
- Creational Patterns, ossia Pattern creazionali: propongono soluzioni per creare oggetti.
- Behavioral Pattern, ossia Pattern comportamentali: propongono soluzioni per gestire il modo in cui vengono suddivise le responsabilità delle classi e degli oggetti.
- Structural Patterns, ossia Pattern strutturali: propongono soluzioni per la composizione strutturale di classi e oggetti

8.1. Pattern Singleton



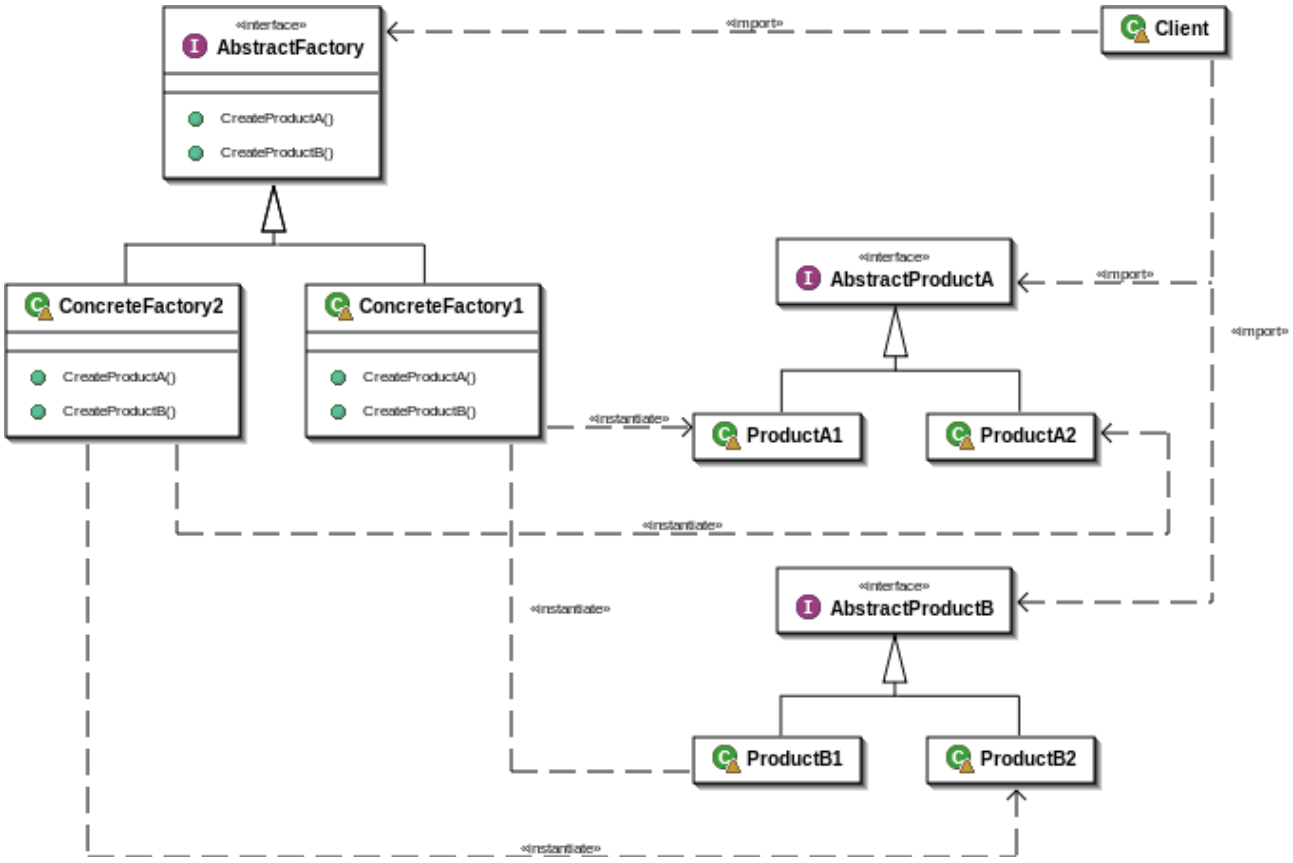
- Scopo: permette di ritornare la stessa istanza dell'oggetto.
- Motivazione: permette a una classe singleton separata di focalizzarsi sulla corretta costruzione di un'istanza evitando che si possano creare più istanze dello stesso oggetto.

8.2. Pattern Facade



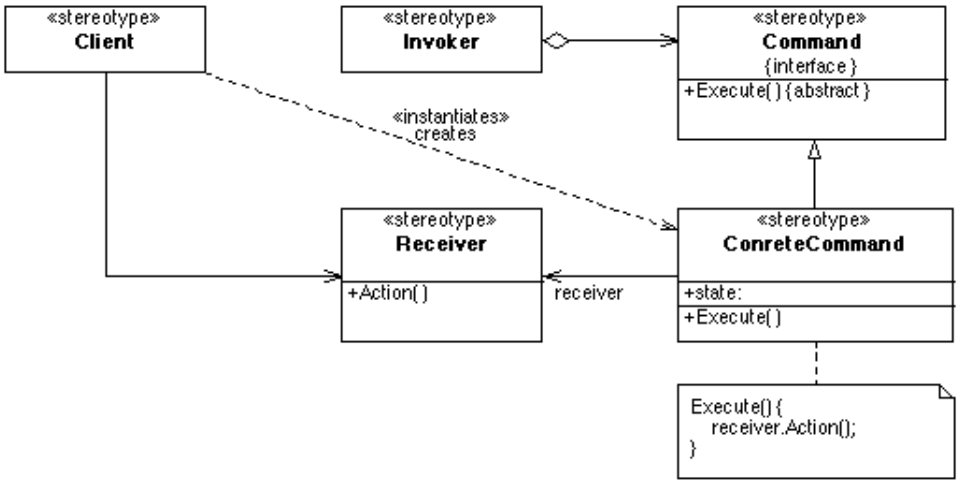
- Scopo: permette, attraverso un'interfaccia più semplice, l'accesso a sottosistemi che espongono interfacce complesse e molto diverse tra loro, nonché a blocchi di codice complessi.
- Motivazione: una classe che espone un singolo metodo, che a sua volta richiamerà i metodi delle classi interne. In questo modo quando un'altra classe deve svolgere un metodo di business che si è già implementato si deve ridurre semplicemente a richiamare un solo metodo della classe che fa da Facade (facciata appunto).

8.3. Pattern Factory



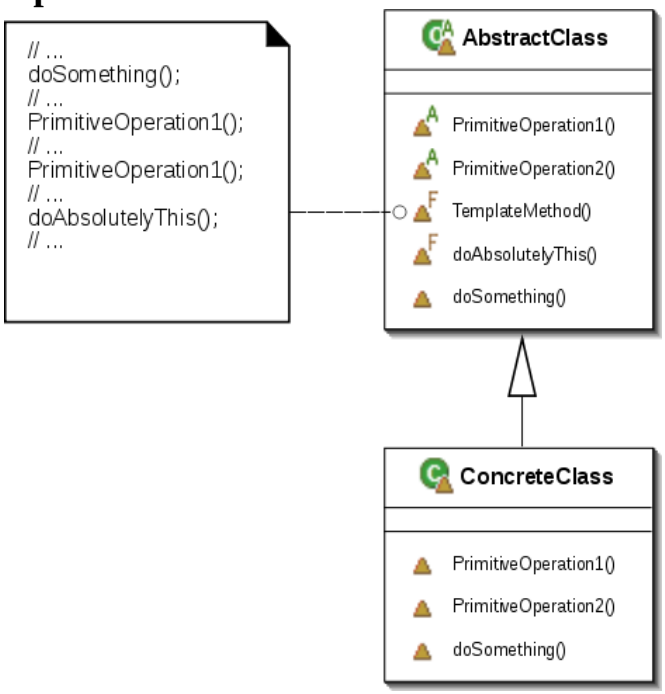
- Scopo: fornisce un'interfaccia per creare un oggetto, ma lascia che le sottoclassi decidano quale oggetto istanziare.
- Motivazione: indirizza il problema della creazione di oggetti senza specificarne l'esatta classe. La creazione di un oggetto può spesso richiedere processi complessi la cui collocazione all'interno della classe di composizione potrebbe non essere appropriata. Esso può, inoltre, comportare la duplicazione di codice, richiedere informazioni non accessibili alla classe di composizione, o non provvedere un sufficiente livello di astrazione. Il factory method indirizza questi problemi definendo un metodo separato per la creazione degli oggetti; tale metodo può essere ridefinito dalle sottoclassi per definire il tipo derivato di prodotto che verrà effettivamente creato.

8.4. Pattern Command



- Scopo: permette di isolare la porzione di codice che effettua un'azione (eventualmente molto complessa) dal codice che ne richiede l'esecuzione; l'azione è incapsulata nell'oggetto Command.
- Motivazione: rende variabile l'azione del client senza però conoscere i dettagli dell'operazione stessa. Altro aspetto importante è che il destinatario della richiesta può non essere deciso staticamente all'atto dell'istanziazione del command ma ricavato a tempo di esecuzione.

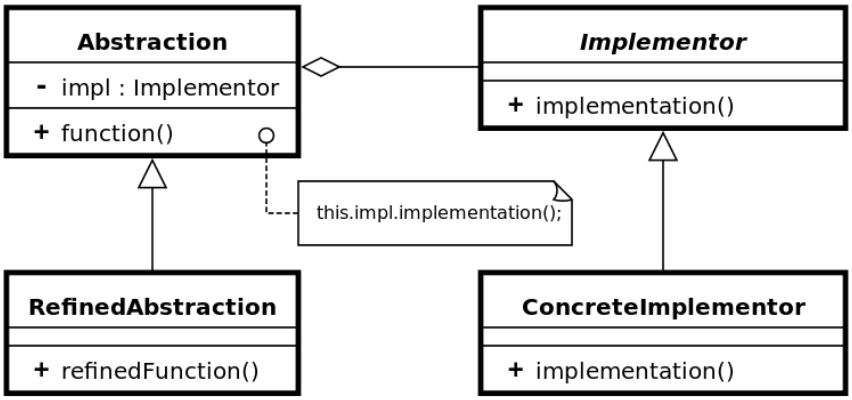
8.5. Pattern Template



- Scopo: permette di definire la struttura di un algoritmo lasciando alle sottoclassi il compito di implementarne alcuni passi come preferiscono. In questo modo si può ridefinire e personalizzare parte del comportamento nelle varie sottoclassi senza dover riscrivere più volte il codice in comune.

- Motivazione: ribalta il meccanismo dell'ereditarietà. Normalmente siano le sottoclassi a chiamare i metodi delle classi genitrici; in questo pattern è il metodo template, appartenente alla classe genitrice, a chiamare i metodi specifici ridefiniti nelle sottoclassi.

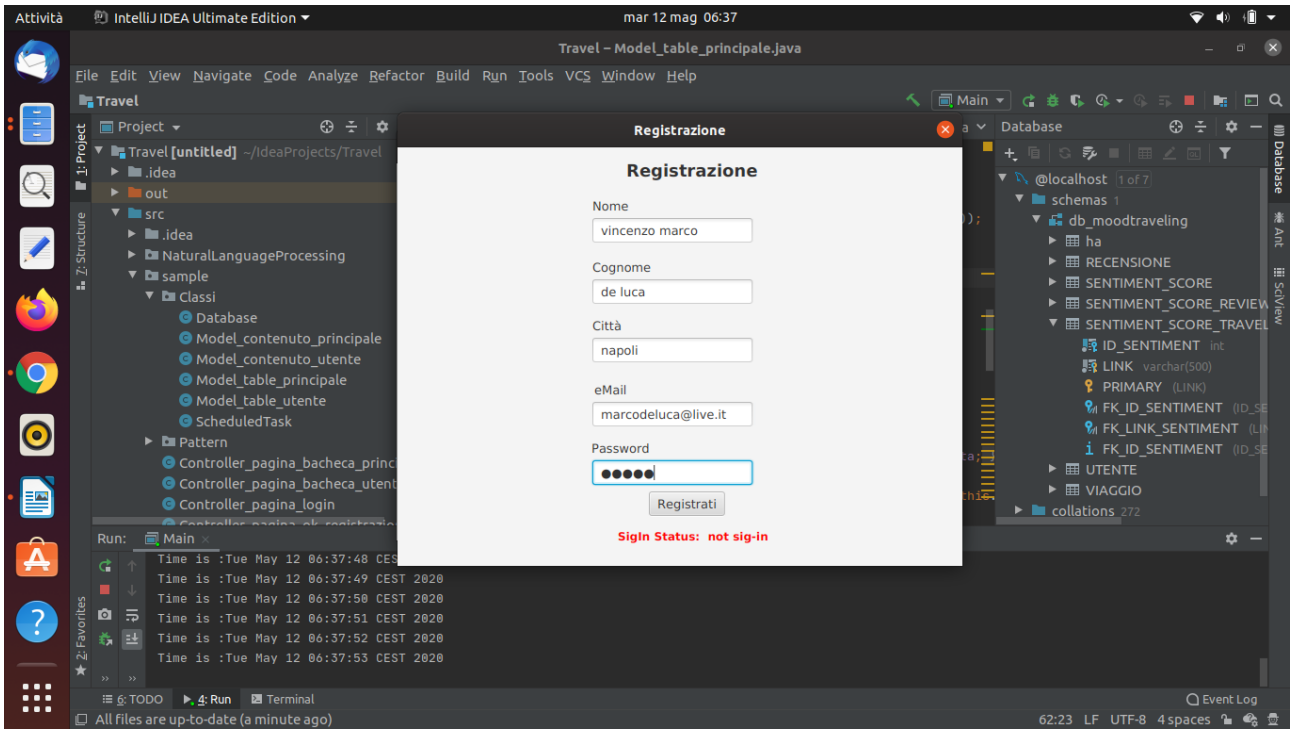
8.6. Pattern Bridge



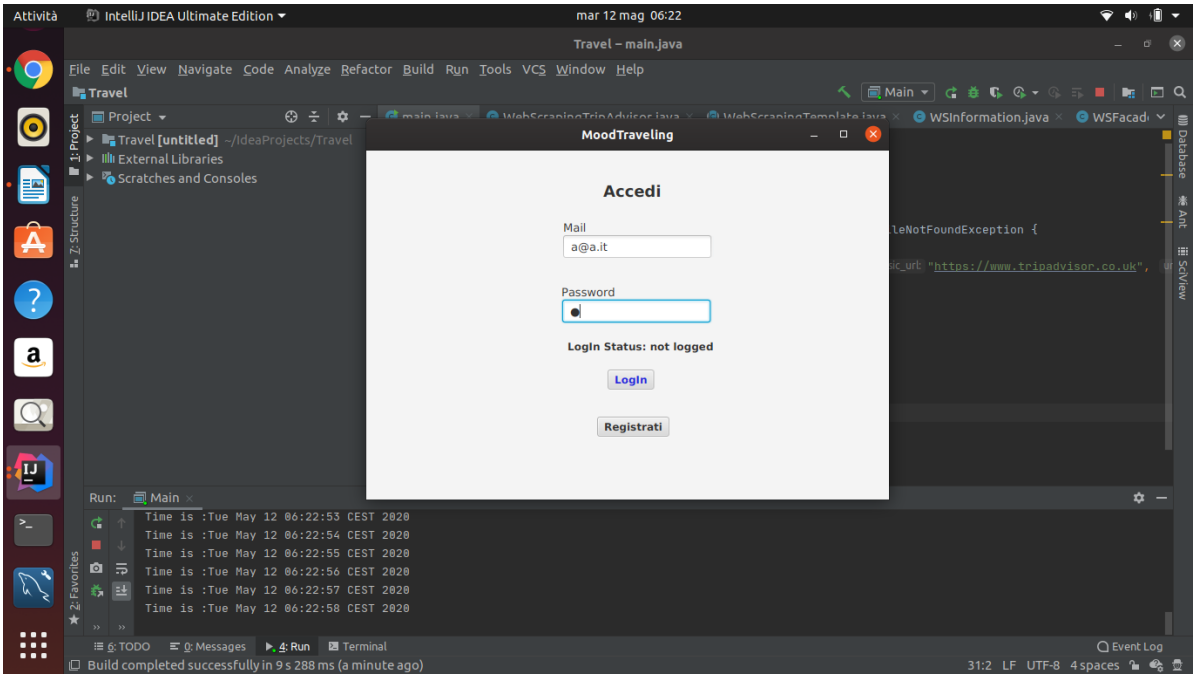
- Scopo: permette di separare l'interfaccia di una classe (che cosa si può fare con la classe) dalla sua implementazione (come lo fa). In tal modo si può usare l'ereditarietà per fare evolvere l'interfaccia o l'implementazione in modo separato.
- Motivazione: disaccoppia l'astrazione dall'implementazione in modo che entrambi possano variare in modo indipendente. L'astrazione e l'implementazione possono cambiare indipendentemente l'una dall'altra e non sono vincolate al momento della compilazione.

9. **Screen di esecuzione con commenti**

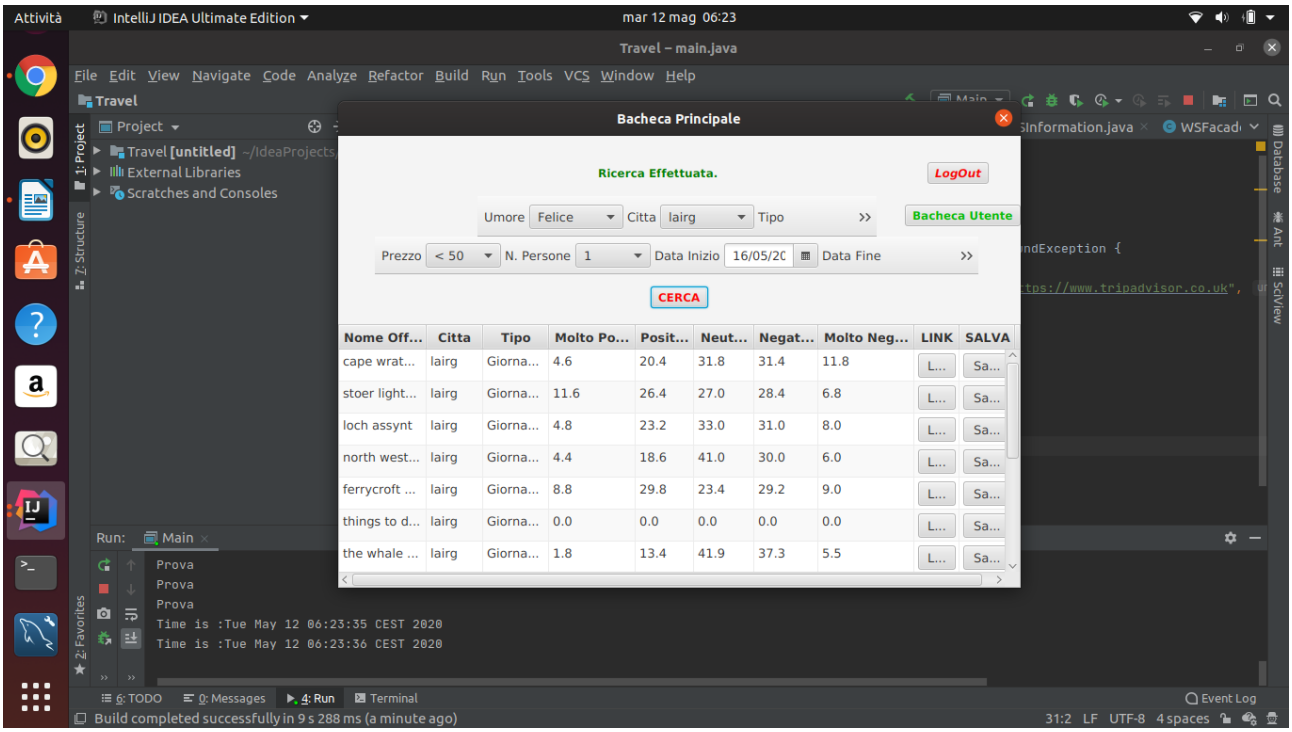
L’utente ha la possibilità di iscriversi se non possiede già un account



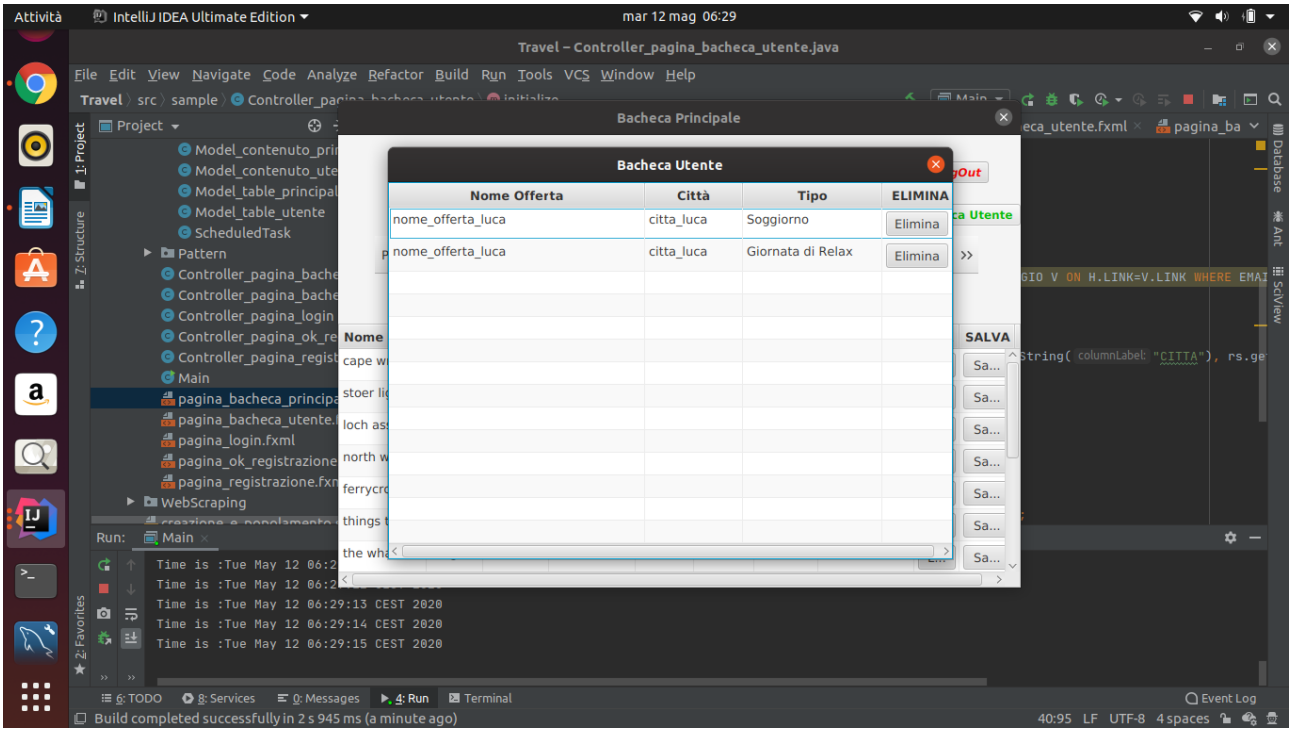
L’utente ha la possibilità di loggarsi se ha già un account



L’utente ha la possibilità di eseguire ricerche sulla bacheca principale


Attività
IntelliJ IDEA Ultimate Edition
mar 12 mag 06:23
Travel – main.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Travel
Project
Travel [untitled] ~/IdeaProjects
External Libraries
Scratches and Consoles
Bacheca Principale
Ricerca Effettuata.
LogOut
Umore Felice Citta lairg Tipo
Bacheca Utente
Prezzo < 50 N. Persone 1 Data Inizio 16/05/2C Data Fine
CERCA
Nome Off... Citta Tipo Molto Po... Posit... Neut... Negat... Molto Neg... LINK SALVA
cape wrat... lairg Giorna... 4.6 20.4 31.8 31.4 11.8 L... Sa...
stoer light... lairg Giorna... 11.6 26.4 27.0 28.4 6.8 L... Sa...
loch assynt lairg Giorna... 4.8 23.2 33.0 31.0 8.0 L... Sa...
north west... lairg Giorna... 4.4 18.6 41.0 30.0 6.0 L... Sa...
ferrycroft ... lairg Giorna... 8.8 29.8 23.4 29.2 9.0 L... Sa...
things to d... lairg Giorna... 0.0 0.0 0.0 0.0 0.0 L... Sa...
the whale ... lairg Giorna... 1.8 13.4 41.9 37.3 5.5 L... Sa...
Run: Main
Prova
Prova
Prova
Time is :Tue May 12 06:23:35 CEST 2020
Time is :Tue May 12 06:23:36 CEST 2020
6: TODO 0: Messages 4: Run Terminal
Event Log
Build completed successfully in 9 s 288 ms (a minute ago)
31:2 LF UTF-8 4 spaces

L’utente ha la possibilità di salvare dei viaggi a cui è interessato nella bacheca utente



10.Sviluppi futuri

L’intenzione del software è quella di rendere le valutazioni dei viaggi “scientificamente” valide attraverso l’analisi delle recensioni non delegando all’utente finale una valutazione su scale arbitrarie. Le implementazioni future possono prevedere analisi su siti differenti e l’eliminazione di stelle o scale delegate all’utente che ha un personale metro di giudizio. Attraverso l’analisi del linguaggio e delle caratteristiche che esso porta, lo strumento sviluppato risulta analiticamente vantaggioso.

Future funzionalità:

- Possibilità di effettuare la ricerca in tutti i siti di viaggi.
- Analisi dell’utente in relazione ai viaggi scelti in modo da garantire una ricerca più accurata.
- Possibilità di effettuare la prenotazione direttamente dall’applicazione.
- Possibilità di eliminare la prenotazione di un viaggio.
- Ampliamento dell’applicazione su tematiche come Trekking o Gite all’aperto.
- Possibilità di eliminare la prenotazione di un viaggio.