

Università degli Studi di Napoli Federico II

**Corso di Laurea in Ingegneria Informatica**

**Corso di Ingegneria del Software**

Immagine che contiene testo, libreria, libro, computer

Il contenuto generato dall'IA potrebbe non essere corretto.**Prof. A.R. Fasolino - A.A. 2024 - 25**

*PROGETTO DI  
INGEGNERIA DEL SOFTWARE*

***“SISTEMA SOFTWARE PER GESTIONE BIBLIOTECA ONLINE”***

**INDICE**

[1. Specifiche informali 1](#_Toc204776464)

[2. Analisi e specifica dei requisiti 2](#_Toc204776465)

[2.1 Analisi nomi-verbi 2](#_Toc204776466)

[2.2 Revisione dei requisiti 3](#_Toc204776467)

[2.3 Glossario dei termini 4](#_Toc204776468)

[2.4 Classificazione dei requisiti 4](#_Toc204776469)

[2.4.1 Requisiti funzionali 4](#_Toc204776470)

[2.4.2 Requisiti sui dati 5](#_Toc204776471)

[2.4.3 Vincoli / Altri requisiti 6](#_Toc204776472)

[2.5 Modellazione dei casi d’uso 7](#_Toc204776473)

[2.5.1 Attori e casi d’uso 7](#_Toc204776474)

[2.5.2 Diagramma dei casi d’uso 9](#_Toc204776475)

[2.5.3 Scenari 9](#_Toc204776476)

[2.6 Diagramma delle classi 10](#_Toc204776477)

[2.7 Diagrammi di sequenza 12](#_Toc204776478)

[2.7.1 Accesso 12](#_Toc204776479)

[2.7.2 AggiungiAuto 12](#_Toc204776480)

[2.8 Diagramma delle classi raffinato 13](#_Toc204776481)

[3. Piano di test funzionale 13](#_Toc204776482)

[3.1 Registrazione Utente 13](#_Toc204776483)

[4. Progettazione 16](#_Toc204776484)

[4.1 Diagramma delle classi 16](#_Toc204776485)

[4.1.1 Traduzione classi ed associazioni 16](#_Toc204776486)

[4.1.2 Pattern BCED 16](#_Toc204776487)

[4.1.2.1 Package Boundary 16](#_Toc204776488)

[4.1.2.2 16](#_Toc204776489)

[4.1.2.3 Package Controller 16](#_Toc204776490)

[4.1.2.4 Package Entity 17](#_Toc204776491)

[4.1.2.5 Package Database 17](#_Toc204776492)

[4.2 Diagrammi di sequenza 17](#_Toc204776493)

[4.2.1 Registrazione 18](#_Toc204776494)

[5. Implementazione 19](#_Toc204776495)

[5.1 Package Database 19](#_Toc204776496)

[5.2 Package Entity 19](#_Toc204776497)

[5.3 Package Controller 19](#_Toc204776498)

[5.4 Package Boundary 19](#_Toc204776499)

[5.5 Package DTO 19](#_Toc204776500)

[5.6 Diagramma di Deployment 20](#_Toc204776501)

[6. Testing 21](#_Toc204776502)

[6.1 Test Strutturale 21](#_Toc204776503)

[6.1.1 Complessità ciclomatica 21](#_Toc204776504)

[6.1.1.1 inserisciAutoModifiche – GestioneParcoAuto 21](#_Toc204776505)

[6.2 JUnit – Test di Unità 23](#_Toc204776506)

[6.3 Test funzionale 24](#_Toc204776507)

# Specifiche informali

*Si desidera sviluppare un sistema software per la gestione di una biblioteca digitale che consenta agli utenti di cercare e prenotare libri online e ritirali in sede.*

La biblioteca avrà un catalogo digitale che gli utenti potranno consultare, attraverso opportuna interfaccia grafica, per trovare libri di loro interesse, utilizzando filtri come autore, genere, o anno di pubblicazione. Ogni libro nel catalogo avrà una scheda dettagliata che include il titolo, l'autore, l'anno di pubblicazione, il genere, il codice ISBN, una breve descrizione e la disponibilità attuale di copie di quel libro. Ogni copia è identificata da un identificativo numerico univoco e può assumere lo stato di disponibile, prenotata, o in prestito.

Gli utenti potranno registrarsi al sito fornendo i propri dati personali, tra cui nome, cognome, indirizzo e-mail e una password. Una volta completata la registrazione, potranno accedere, tramite autenticazione, al proprio profilo personale. All’interno del profilo sarà possibile visualizzare e modificare i dati personali, caricare un’immagine del profilo, consultare l’elenco dei libri prenotati e la cronologia di quelli precedentemente consultati.

Dal profilo personale, inoltre, l’utente potrà accedere al catalogo dei libri disponibili nella biblioteca, al fine di poter prenotare eventualmente un libro di interesse, e gestire le prenotazioni attive, con la possibilità di annullarle in qualsiasi momento.

Quando un utente prenota un libro, il sistema ne verificherà la disponibilità di una copia e, in caso affermativo, la riserverà per l'utente. Quando un libro viene prenotato viene generato una ricevuta della prenotazione, che include, tra le altre informazioni, un ID della prenotazione, data di prevista consegna, e costo complessivo del prestito (calcolato in base al numero di giorni).

Gli addetti alla biblioteca invece hanno una interfaccia grafica riservata che gli permette di visualizzare l’elenco delle prenotazioni dei libri e di procedere a registrare il ritiro di una copia prenotata. All’atto del ritiro, l’utente fornirà l’ID della prenotazione ricevuto e l’addetto potrà cambiare lo stato della copia in “non disponibile”.

Una ulteriore interfaccia invece consente agli addetti di generare un report di tutti i libri che sono in prestito, riportando per ciascun libro il numero di copie in prestito e quelle disponibili e di poter registrare la avvenuta restituzione di una copia di un libro alla biblioteca (attraverso l’ID della prenotazione), rendendola di nuovo disponibile a future prenotazioni.

La biblioteca digitale consentirà anche agli amministratori di gestire il catalogo aggiungendo nuovi libri, nuove copie di libri, modificando quelli esistenti o eliminando quelli obsoleti. Inoltre, gli amministratori potranno generare report mensili sull'uso del servizio (incasso), quali i libri più prenotati, gli utenti più attivi e le tendenze di lettura (genere più letto in un determinato intervallo di tempo).

# Analisi e specifica dei requisiti

## Analisi nomi-verbi

La biblioteca avrà un catalogo digitale che gli utenti potranno consultare, attraverso opportuna interfaccia grafica, per trovare libri di loro interesse, utilizzando filtri come autore, genere, o anno di pubblicazione. Ogni libro nel catalogo avrà una scheda dettagliata che include il titolo, l'autore, l'anno di pubblicazione, il genere, il codice ISBN, una breve descrizione e la disponibilità attuale di copie di quel libro. Ogni copia è identificata da un identificativo numerico univoco e può assumere lo stato di disponibile, prenotata, o in prestito.

Gli utenti potranno registrarsi al sito fornendo i propri dati personali, tra cui nome, cognome, indirizzo e-mail e una password. Una volta completata la registrazione, potranno accedere, tramite autenticazione, al proprio profilo personale. All’interno del profilo sarà possibile visualizzare e modificare i dati personali, caricare un’immagine del profilo, consultare l’elenco dei libri prenotati e la cronologia di quelli precedentemente consultati.

Dal profilo personale, inoltre, l’utente potrà accedere al catalogo dei libri disponibili nella biblioteca, al fine di poter prenotare eventualmente un libro di interesse, e gestire le prenotazioni attive, con la possibilità di annullarle in qualsiasi momento.

Quando un utente prenota un libro, il sistema ne verificherà la disponibilità di una copia e, in caso affermativo, la riserverà per l'utente. Quando un libro viene prenotato viene generato una ricevuta della prenotazione, che include, tra le altre informazioni, un ID della prenotazione, data di prevista consegna, e costo complessivo del prestito (calcolato in base al numero di giorni).

Gli addetti alla biblioteca invece hanno una interfaccia grafica riservata che gli permette di visualizzare l’elenco delle prenotazioni dei libri e di procedere a registrare il ritiro di una copia prenotata. All’atto del ritiro, l’utente fornirà l’ID della prenotazione ricevuto e l’addetto potrà cambiare lo stato della copia in “non disponibile”.

Una ulteriore interfaccia invece consente agli addetti di generare un report di tutti i libri che sono in prestito, riportando per ciascun libro il numero di copie in prestito e quelle disponibili e di poter registrare la avvenuta restituzione di una copia di un libro alla biblioteca (attraverso l’ID della prenotazione), rendendola di nuovo disponibile a future prenotazioni.

La biblioteca digitale consentirà anche agli amministratori di gestire il catalogo aggiungendo nuovi libri, nuove copie di libri, modificando quelli esistenti o eliminando quelli obsoleti. Inoltre, gli amministratori potranno generare report mensili sull'uso del servizio (incasso), quali i libri più prenotati, gli utenti più attivi e le tendenze di lettura (genere più letto in un determinato intervallo di tempo).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Classe* | *Attributo* | *Funzionalità* | *Attore* | *Classe-Attore* |

## Revisione dei requisiti

1. *Il sistema deve offrire all’Utente una funzionalità per registrarsi*
2. *Per ogni Utente si vuole memorizzare nome, cognome, indirizzo e-mail e password*
3. *Il sistema deve offrire all’Utente una funzionalità per accedere*
4. *Il sistema deve offrire all’Utente una funzionalità per gestire il profilo*
5. *Il sistema deve offrire all’Utente una funzionalità per gestire l’elenco delle Prenotazioni attive*
6. *Il sistema deve offrire all’Utente una funzionalità per effettuare una Prenotazione*
7. *Il sistema deve offrire all’Utente una funzionalità di ricerca dei Libri nel Catalogo, tramite autore, genere e anno di pubblicazione*
8. *Il sistema deve permettere di verificare la disponibilità di almeno una copia prima di consentire la prenotazione all’Utente*
9. *Il sistema deve generare all’Utente una ricevuta della Prenotazione con i relativi dati*
10. *Per ogni Prenotazione si vuole memorizzare: ID della prenotazione, ID copia, ID utente, data di prevista consegna, costo totale*
11. *Il sistema deve offrire all’Utente una funzionalità per annullare una Prenotazione attiva in qualsiasi momento prima del ritiro*
12. *Il sistema deve offrire all’Addetto una funzionalità per visualizzare l’elenco delle Prenotazioni*
13. *Il sistema deve offrire all’Addetto una funzionalità per gestire le Prenotazioni*
14. *Il sistema deve offrire all’Addetto una funzionalità di generare un Report dei libri in prestito*
15. *Per ogni Report si deve indicare numero di copie in prestito e numero di copie disponibili*
16. *Il sistema deve offrire agli Amministratori una funzionalità gestire il Catalogo*
17. *Il sistema deve offrire agli Amministratori una funzionalità per generare Report mensili*
18. *Per ogni Libro si vuole memorizzare: ISBN, titolo, autore, anno di pubblicazione, genere, descrizione, disponibilità attuale di copie*
19. *Per ogni Copia si vuole memorizzare ID della copia e stato*
20. *Per ogni Copia lo stato deve essere uno tra: “disponibile”, “prenotata”, “in prestito”*
21. *Il sistema deve offrire all’Utente una funzionalità per visualizzare le consultazioni*

## Glossario dei termini

|  |  |  |
| --- | --- | --- |
| Termine | Descrizione | Sinonimi |
| Biblioteca | Sistema software online per consultazione e prenotazione di libri | Sistema, biblioteca digitale |
| Utente non registrato | Persona non registrata | Guest |
| Utente registrato | Persona registrata con accesso alle funzionalità base | User |
| Addetto | Operatore interno con compiti di gestione prenotazioni e restituzioni | Addetti alla biblioteca, Staff |
| Amministratore | Utente con permessi avanzati di gestione e reportistica | Admin |
| Libro | Oggetto bibliografico descritto nel catalogo | - |
| Copia | Unità fisica del libro, identificata univocamente | - |
| Prenotazione | Richiesta di riservare una copia di un libro | - |
| Ritiro | Azione fisica di prelevare una copia prenotata presso la sede | - |
| Restituzione | Riconsegna della copia ritirata alla biblioteca | - |
| Stato della copia | Situazione attuale della copia: disponibile, prenotata, in prestito | - |
| Ricevuta | Documento generato dal sistema al momento della prenotazione | - |
| Report | Documento riepilogativo di dati generati dal sistema | - |
| Catalogo | Elenco dei libri presenti nel sistema | Catalogo digitale |
| Interfaccia | Parte visibile del sistema interattiva per l’utente | Interfaccia grafica, GUI |

## Classificazione dei requisiti

### Requisiti funzionali

|  |  |  |
| --- | --- | --- |
| **ID** | **Requisito** | **Origine (n. frase dei requisiti revisionati)** |
| **RF01** | Il sistema deve offrire all’Utente una funzionalità per registrarsi | 1 |
| **RF02** | Il sistema deve offrire all’Utente una funzionalità per accedere | 3 |
| **RF03** | Il sistema deve offrire all’Utente una funzionalità per gestire il profilo | 4 |
| **RF04** | Il sistema deve offrire all’Utente una funzionalità per gestire l’elenco delle Prenotazioni Attive | 5 |
| **RF05** | Il sistema deve offrire all’Utente una funzionalità per effettuare una Prenotazione | 6 |
| **RF06** | Il sistema deve offrire all’Utente una funzionalità di ricerca dei Libri nel Catalogo tramite autore, genere e anno di pubblicazione | 7 |
| **RF07** | Il sistema deve permettere di verificare la disponibilità di almeno una copia prima di consentire la prenotazione all’Utente | 8 |
| **RF08** | Il sistema deve generare all’Utente una ricevuta della Prenotazione con i relativi dati | 9 |
| **RF09** | Il sistema deve offrire all’Utente una funzionalità per annullare una Prenotazione attiva in qualsiasi momento prima del ritiro | 11 |
| **RF10** | Il sistema deve offrire all’Addetto una funzionalità per visualizzare le Prenotazioni | 12 |
| **RF11** | Il sistema deve offrire all’Addetto una funzionalità per gestire le Prenotazioni | 13 |
| **RF12** | Il sistema deve offrire all’Addetto una funzionalità di generare un Report dei libri in prestito | 14 |
| **RF13** | Il sistema deve offrire agli Amministratori una funzionalità per gestire il Catalogo | 16 |
| **RF14** | Il sistema deve offrire agli Amministratori una funzionalità per generare Report mensili | 17 |
| **RF15** | Il sistema deve offrire all’Utente una funzionalità per visualizzare le consultazioni | 21 |

### Requisiti sui dati

|  |  |  |
| --- | --- | --- |
| **ID** | **Requisito** | **Origine (n. frase dei requisiti revisionati)** |
| **RD01** | Per ogni Utente si vuole memorizzare nome, cognome, indirizzo e-mail e password | 2 |
| **RD02** | Per ogni Libro si vuole memorizzare: ISBN, titolo, autore, anno di pubblicazione, genere, descrizione, disponibilità attuale di copie | 18 |
| **RD03** | Per ogni Prenotazione si vuole memorizzare: ID della prenotazione, ID copia, ID utente, data di prevista consegna, costo totale | 10 |
| **RD04** | Per ogni Report si deve indicare numero di copie in prestito e numeri di copie disponibili | 15 |
| **RD05** | Per ogni Copia si vuole memorizzare ID della copia e stato | 19 |

### Vincoli / Altri requisiti

|  |  |  |
| --- | --- | --- |
| **ID** | **Requisito** | **Origine (n. frase dei requisiti revisionati)** |
| **V01** | Per ogni Copia lo stato deve essere uno tra: “disponibile”, “prenotata”, “in prestito” | 20 |

## Modellazione dei casi d’uso

### Attori e casi d’uso

***Attori Primari:***

* Utente non registrato (*Guest*)
* Utente registrato (*User*)
* Addetto biblioteca (*Staff*)
* Amministratore (*Admin*)

***Attori Secondari:***

* Sistema di Autenticazione

***Casi d’uso:***

* **UC1**: Ricerca libro
* **UC2**: Registrazione
* **UC3**: Accesso
* **UC4**: Visualizzare consultazioni
* **UC5**: Gestire profilo
* **UC6**: Gestire prenotazioni attive
* **UC7**: Effettuare prenotazione
* **UC8**: Genera report prestiti
* **UC9:** Visualizzare prenotazioni
* **UC10**: Gestire prenotazioni
* **UC11**: Gestire catalogo
* **UC12**: Genera report mensile

***Casi d’uso di inclusione:***

* **UC6.1**: Annullare prenotazione
* **UC7.1**: Verifica disponibilità
* **UC7.2**: Genera ricevuta

***Casi d’uso di estensione:***

* **UC1.1**: Registra consultazione (estende **UC1**, solo se l’utente è loggato)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Caso d’uso | Attori Primari | Attori Secondari | Incl. / Ext. | Requisiti corrispondenti |
| UC1:  Ricerca libro | Guest /  User | - |  | RF06 |
| UC1.1:  Registrare consultazione | User | - | Extend:  **UC1** (solo se l’utente è User) | RF15 |
| UC2:  Registrazione | Guest | - |  | RF01 |
| UC3:  Accesso | User | Sistema di Autenticazione |  | RF02 |
| UC4:  Visualizzare consultazioni | User | - |  | RF15 |
| UC5:  Gestire profilo | User | - |  | RF03 |
| UC6:  Gestire prenotazioni attive | User | - | Include:  **UC6.1** | RF04 |
| UC6.1:  Annullare prenotazione | - | - | Incluso in:  UC6 | RF09 |
| UC7:  Effettuare prenotazione | User | - | Include:  **UC7.1**, **UC7.2** | RF05 |
| UC7.1:  Verifica disponibilità | - | - | Incluso in:  UC7 | RF07 |
| UC7.2:  Genera ricevuta | - | - | Incluso in:  UC7 | RF08 |
| UC8:  Genera report prestiti | Staff | - |  | RF12 |
| UC9:  Visualizzare prenotazioni | Staff | - |  | RF10 |
| UC10:  Gestire prenotazioni | Staff | User |  | RF11 |
| UC11:  Gestire catalogo | Admin | - |  | RF13 |
| UC12:  Generare report mensile | Admin | - |  | RF14 |

### Diagramma dei casi d’uso

Immagine che contiene testo, calligrafia, diagramma, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

### Scenari

Selezionare un caso d’uso (dunque una funzionalità) per ogni membro del gruppo, da sviluppare fino alla codifica in Java (dunque, diagramma di sequenza di analisi raffinato, diagramma di sequenza di progettazione, implementazione e test del caso d’uso scelto).

Riportare in questa sezione lo scenario principale per il caso d’uso scelto, come nel seguente esempio

|  |  |
| --- | --- |
| Caso d’uso: | PrenotaAuto |
| Attore primario | Cliente |
| Attore secondario | - |
| Descrizione | Un cliente richiede di noleggiare un’auto specificando data e ora di inizio e fine noleggio |
| Pre-Condizioni | Il Cliente ha effettuato l’accesso |
| Sequenza di eventi  principale | 1. Il caso d’uso inizia quando il Cliente accede alla sezione dedicata al noleggio 2. Il Cliente specifica data e ora di inizio e fine noleggio 3. Il sistema mostra le Auto disponibili per il periodo di interesse 4. *<<include>> VisualizzaAutoDisponibili* 5. **if** ci sono Auto disponibili    1. Il Cliente seleziona l’Auto di suo gradimento    2. Il sistema verifica che la patente del Cliente sia idonea alla guida del veicolo di interesse    3. *<<include>> VerificaPatente*    4. **if** le verifiche sono andate a buon fine       1. Il sistema verifica che il Cliente abbia la disponibilità sufficiente per il noleggio       2. *<<include>> VerificaCredito*       3. **if** le verifiche sono andate a buon fine          1. Il sistema procede con la prenotazione          2. Il sistema aggiunge il Noleggio all’archivio dei Noleggi          3. Il sistema invia un messaggio di conferma dell’avvenuta prenotazione al Cliente          4. *<<include>> InvioConferma* |
| Post-Condizioni | Il Cliente ha prenotato il noleggio e il sistema valuterà come *non disponibile* l’Auto selezionata per l’intero periodo di utilizzo |
| Casi d’uso correlati | *VerificaPatente, VerificaCredito, InvioConferma, AggiornaStato* |
| Sequenza di eventi  alternativi | Al punto 5, se non ci sono Auto disponibili per il Noleggio, il sistema restituisce un messaggio di errore  Al punto 5.4, se le verifiche non sono andate a buon fine, il sistema restituisce un messaggio di errore e chiede al Cliente di selezionare un’altra autovettura  Al punto 5.4.3, se le verifiche non sono andate a buon fine, il sistema restituisce un messaggio di errore e chiede al Cliente di selezionare un’altra autovettura, ritornando al punto 3, o di aggiornare la propria disponibilità |

## Diagramma delle classi

Sviluppare il Diagramma delle classi di analisi corrispondente al System Domain Model.

Realizzare anche una versione raffinata che includa le responsabilità delle classi.

A seguire, una tabella che riassume alcune responsabilità:

|  |  |
| --- | --- |
| **RESPONSABILITÀ** | **CLASSE** |
| *Registrazione* | **Autonoleggio** |
| *Accesso* | **Autonoleggio** |
| *PrenotaAuto* | **Cliente** |
| *AggiungiAuto* | **Autonoleggio** |
| *RimuoviAuto* | **Autonoleggio** |
| *ModificaAuto* | **Auto** |
| *-* | **-** |
| *-* | **-** |
| *-* | **-** |
| *-* | **-** |
| *-* | **--** |
| *VerificaCredito* | **Cliente** |
| *-* | **-** |
| *AggiornaCredito* | **Cliente** |

[Qui va aggiunta una descrizione/giustificazione di come è stata assegnata la responsabilità, tenendo presente anche i pattern GRASP.]

*Registrazione* e *Accesso* sono responsabilità di **Autonoleggio**, in quanto <<information expert>> di Clienti.

*PrenotaAuto* è responsabilità di **Cliente**, perché è <<Creator>> della classe Noleggio.

........

## Diagrammi di sequenza

Riportare il diagramma di sequenza di analisi per le funzionalità (ossia i caso d’uso) da implementare (sceglierne una non banale per ogni membro del gruppo) e che saranno sviluppate fino alla codifica in Java ed al test.

Si riportano alcuni esempi di diagrammi di sequenza di analisi per i casi d’uso precedentemente individuati.

### Accesso

La creazione del suddetto sequence diagram, sviluppato a partire dalla descrizione dello scenario del caso d’uso *Accesso*, ha fatto sorgere la necessità di definire un metodo, specifico per la classe **Autonoleggio**, **verificaCredenziali(nomeUtente, password)**, privato, per consentire all’autonoleggio di verificare che le credenziali inserite dall’utente siano valide.

Immagine che contiene testo, schermata, linea, diagramma

Descrizione generata automaticamente

### AggiungiAuto

Immagine che contiene testo, schermata, linea, diagramma

Descrizione generata automaticamente

Per le stesse ragioni, è stato necessario inserire all’interno della classe **Autonoleggio** il metodo privato **verificaPresenzaAuto(targa)**, col fine ultimo di individuare se l’auto che il gestore intende aggiungere è già presente all’interno del parco auto.

……..

## Diagramma delle classi raffinato

Le aggiunte e le modifiche fatte nel corso della costruzione dei Sequence Diagrams hanno determinato lo sviluppo di un **Diagramma delle Classi** raffinato che riporta maggiori dettagli sugli attributi e le principali operazioni delle classi:

Riportare CD raffinato

# Piano di test funzionale

Progettare i casi di test funzionale con la tecnica del *Category Partition Testing*. Descrivere il procedimento di calcolo.

Si intende progettare i casi di test funzionale con la tecnica del ***Category Partition Testing***.

## Registrazione Utente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **REGISTRAZIONE** | | | | |
| **NOME** | **COGNOME** | **PATENTE** | **TIPOPATENTE** | **CREDITO** |
| * Stringa di caratteri di lunghezza <= 40 * Stringa di caratteri di lunghezza > 40 **[ERROR]** * Stringa che contiene simboli che non sono caratteri **[ERROR]** | * Stringa di caratteri di lunghezza <= 40 * Stringa di caratteri di lunghezza > 40 **[ERROR]** * Stringa che contiene simboli che non sono caratteri **[ERROR]** | * Stringa di caratteri alfanumerici di lunghezza = 10 * Stringa di caratteri alfanumerici di lunghezza > 10 **[ERROR]** * Stringa di caratteri alfanumerici di lunghezza < 10 **[ERROR]** * Stringa contenente caratteri speciali **[ERROR]** | * Stringa di caratteri alfanumerici di lunghezza <= 3 * Stringa di caratteri alfanumerici di lunghezza > 3 **[ERROR]** * Stringa contenente caratteri speciali **[ERROR]** | * Numero decimale >= 0 * Numero decimale < 0 **[ERROR]** * Numero che contiene simboli che non sono numeri **[ERROR]** |

Il numero di test da effettuarsi senza particolari vincoli è: .

Con i vincoli [ERROR], invece, il numero di test da eseguire per testare singolarmente i vincoli è 11 (2 per Nome, 2 per Cognome, 3 per Patente, 2 per TipoPatente, 2 per Credito).

Il numero di test risultante è: (1\*1\*1\*1\*1) + 11 = 12.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TEST SUITE** | | | | | | |
| **Test Case ID** | **Descrizione** | **Classi di equivalenza coperte** | **Pre-condizioni** | **Input** | **Output attesi** | **Post-condizioni attese** |
| 1 | Tutti input validi | Nome valido  Cognome valido  Patente valida  TipoPatente valido  Credito valido |  | {Nome: “Mario”, Cognome: “Rossi”, Patente: “NAH68I903B”, TipoPatente: “B1”, Credito “230.50”} | Cliente registrato | Si ricevono le credenziali di accesso per email |
| 2 | Nome stringa > 40 caratteri | Nome stringa > 40 caratteri [ERROR],  Cognome, Patente, TipoPatente, Credito validi |  | {Nome: “Marioooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo”, Cognome: “Rossi”, Patente: “NAH68I903B”, TipoPatente: “B1”, Credito “230.50”} | Nome troppo lungo! |  |
| 3 | Nome stringa con simboli | Nome stringa con simboli [ERROR], Cognome, Patente, TipoPatente, Credito validi |  | {Nome: “Ma-r.o”, Cognome: “Rossi”, Patente: “NAH68I903B”, TipoPatente: “B1”, Credito “230.50”} | Formato Nome errato! |  |
| 4 | Cognome stringa > 40 caratteri | Nome valido,  Cognome stringa > 40 caratteri [ERROR], Patente, TipoPatente, Credito validi |  | {Nome: “Mario”, Cognome: “Rossiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii”, Patente: “NAH68I903B”, TipoPatente: “B1”, Credito “230.50”} | Cognome troppo lungo! |  |
| 5 | Cognome stringa con simboli | Nome valido,  Cognome stringa con simboli [ERROR], Patente, TipoPatente, Credito validi |  | {Nome: “Mario”, Cognome: “R!si-”, Patente: “NAH68I903B”, TipoPatente: “B1”, Credito “230.50”} | Formato Cognome errato! |  |
| 6 | Patente stringa > 10 caratteri | Nome, Cognome validi, Patente stringa > 10 caratteri [ERROR], TipoPatente, Credito validi |  | {Nome: “Mario”, Cognome: “Rossi”, Patente: “NA80830D023930DL23”, TipoPatente: “B1”, Credito “230.50”} | Patente troppo lunga! |  |
| 7 | Patente stringa < 10 caratteri | Nome, Cognome validi, Patente stringa < 10 caratteri [ERROR], TipoPatente, Credito validi |  | {Nome: “Mario”, Cognome: “Rossi”, Patente: “NA80LP”, TipoPatente: “B1”, Credito “230.50”} | Patente troppo breve! |  |
| 8 | Patente stringa con caratteri speciali | Nome, Cognome validi, Patente stringa con caratteri speciali [ERROR], TipoPatente, Credito validi |  | {Nome: “Mario”, Cognome: “Rossi”, Patente: “NA.68@9!3-”, TipoPatente: “B1”, Credito “230.50”} | Formato Patente errato! |  |
|  | ETCCCC |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

# Progettazione

## Diagramma delle classi

Riportare il diagramma delle classi di progettazione che rispetti il modello architetturale del BCED (senza violarne le regole di dipendenza fra i livelli). Il class diagram dovrà includere le classi Entity (corrispondenti a quelle di dominio), nonché le Classi Boundary, Controller, e Dati (ossia le classi DAO o Wrapper per l’accesso al Database)

Il diagramma dovrà essere organizzato utilizzando i package per raggruppare le classi dello stesso layer.

In questo diagramma saranno inoltre documentate le scelte di progetto fatte, come ad esempio:

* Reificare eventuali classi associative del diagramma delle classi di analisi.
* Specificare argomenti e tipo di ritorno delle operazioni (per quelle più significative, coinvolte nei casi d’uso sviluppati fino alla implementazione).
* Decidere i versi di navigabilità delle associazioni

### Traduzione classi ed associazioni

Spiegare le scelte effettuate

### Pattern BCED

#### Package Boundary

Il package Boundary contiene tutti gli oggetti responsabili dell’interfaccia utente e della logica di presentazione; a questo livello tutte le classi corrispondono a delle interfacce e i relativi attributi non sono altro che gli elementi che le compongono, visualizzati a video.

Riportare il Package Boundary

#### 

#### Package Controller

Questo package contiene gli oggetti che percepiscono gli eventi generati dalle interazioni con l’interfaccia utente e ne demandano la gestione all’unico componente del sistema software responsabile della gestione della Business Logic, il package Entity.

Riportare il Package Controller

#### Package Entity

Il Package Entity contiene tutti gli oggetti che rappresentano la semantica delle entità del dominio applicativo e corrispondono alle strutture dati presenti all’interno del database di persistenza.

Riportare il Package Entity

#### Package Database

Da questo punto di vista, il Package Database contiene tutte le classi responsabili dell’estrazione dei dati dal DB, esponendo una vera e propria interfaccia che di fatto rende indipendenti le classi della Business Logic (Entity) dalla tecnologia di persistenza utilizzata.

In particolare, tra le strategie di risoluzione del problema dell’**impedance mismatch**, che nasce dalla mancata corrispondenza tra il modello Object Oriented e quello relazionale, si è deciso di adottare quella delle classi **DAO** (Data Access Objects), che consiste nell’utilizzo di appositi oggetti per l’accesso ai dati.

Ognuna di queste classi conterrà i metodi CRUD per l’interrogazione e la manipolazione della corrispondente classe di dominio (*query*), implementati in funzione di un’ulteriore classe, ***DBManager***, che costituisce di fatto l’unico punto di accesso vero e proprio al DB, sfruttando i metodi che questa mette a disposizione.

Riportare il Package Database

## Diagrammi di sequenza

Si riportino di seguito i diagrammi di sequenza di progetto per il/i casi d’uso sviluppati fino alla codifica in Java.

### Registrazione



I sequence progettati sono stati fondamentali per la corretta implementazione dell’applicazione software ed ha fatto nascere la necessità di definire ulteriori classi, metodi e funzioni, che hanno arricchito passo dopo passo il **Diagramma delle Classi di Progettazione**, fino ad ottenere la seguente versione finale:

Riportare il Diagramma delle Classi di Progettazione di Dettaglio

# Implementazione

Non includere il codice sorgente, ma descrivere l’implementazione in Java, descrivendo gli artefatti di codifica:

* Elencare:
  + package, classi, tipi di eccezione definiti
* Elencare gli artefatti necessari per l’installazione ed esecuzione del programma, senza ovviamente l’ambiente di sviluppo come Eclipse (DB h2, eventuali librerie e versioni di Java che l’utilizzatore deve avere installati, file .class, .jar, …)
* Produrre un eventuale diagramma di deployment
* Eventualmente inserire la documentazione del codice prodotta con Javadoc (relativamente alle funzionalità implementate)

## Package Database

TBD

## Package Entity

TBD

## Package Controller

TBD

## Package Boundary

TBD

## Package DTO

L’introduzione di tale package, estraneo al pattern BCED, nasce dall’esigenza di mostrare sulla GUI collezioni di elementi.

Da questo punto di vista, il problema principale è proprio quello che, qualora una determinata chiamata a funzione restituisse alla GUI un elenco di entity, questa, per poterlo visualizzare correttamente a video, dovrebbe conoscere di fatto la struttura interna di tale classe Entity, ma ciò porterebbe con sé un accoppiamento troppo elevato e quindi una chiara violazione dei vincoli del pattern a livelli adottato.

Si introduce allora il concetto di **Data Transfer Object** (**DTO**), un oggetto in grado di trasportare dati tra processi (nel caso in oggetto tra livelli). Le classi DTO hanno tipicamente una struttura che rispecchia quella dell’entity di cui vanno a supporto, in particolare gli attributi coincidono con quelli dell’entity che si intendono visualizzare a schermo.

## Diagramma di Deployment

I diagrammi di deployment sono utilizzati per mostrare l’architettura fisica del sistema software realizzato; sono particolarmente utili per valutare, durante lo sviluppo, come un’applicazione si distribuisce tra le varie macchine.

* .

# Testing

## Test Strutturale

Costruire il Control Flow Graph per uno o due dei metodi delle classi implementate (si scelgano metodi non proprio banali), e:

- si mostri il calcolo del numero ciclomatico;

- si indichino i percorsi linearmente indipendenti;

- si progettino i casi di test per coprire i percorsi individuati.

Prima o a fianco del CFG riportare il codice Java del metodo.

### Complessità ciclomatica

Si intende costruire il Control Flow Graph per due dei metodi delle classi implementate e mostrare il calcolo del numero ciclomatico e i percorsi linearmente indipendenti.

#### inserisciAutoModifiche – GestioneParcoAuto

public String inserisciAutoModifiche(String targa, String modello, String cilindrata, String costoGiornaliero, String patenteRichiesta) {

 EntityElencoAuto elenco = new EntityElencoAuto(); **(1)**

ArrayList<AutoDTO> elencoAuto = elenco.getListaAuto(); **(2)**

boolean controlloTarga = false; **(3)**

for(int i=0; i<elencoAuto.size(); i++) { **(4)**

if(targa.compareTo(elencoAuto.get(i).getTarga())==0) { **(5)**

controlloTarga = true; **(6)**

}

}

if(!controlloTarga) { **(7)**

return "Errore, l'auto selezionata è inesistente!"; **(8)**

}

ArrayList<AutoDTO> elencoAutoDeposito = visualizzaElencoAuto(); **(9)**

controlloTarga = false; **(10)**

for(int i=0; i<elencoAutoDeposito.size(); i++) { **(11)**

if(targa.compareTo(elencoAutoDeposito.get(i).getTarga())==0) { **(12)**

controlloTarga = true; **(13)**

}

}

if(!controlloTarga) { **(14)**

return "Errore, l'auto selezionata non rientra tra quelle in deposito!"; **(15)**

}

EntityAuto autoDaModificare = new EntityAuto (targa); **(16)**

return autoDaModificare.modificaAuto(modello, cilindrata, costoGiornaliero, patenteRichiesta); **(17)**

}

**NUMERO CICLOMATICO**:

* numero di regioni chiuse del grafo + 1 = 7
* numero di nodi predicati (4,5,7,11,12,14) +1 = 7
* # archi – # nodi + 2 = (22 - 17) + 2 = 7

**CAMMINI**:

1. 1-2-3-4-7-8
2. 1-2-3-4-5-4-7-8
3. 1-2-3-4-5-6-4-7-8
4. 1-2-3-4-7-9-10-11-14-15
5. 1-2-3-4-7-9-10-11-12-11-14-15
6. 1-2-3-4-7-9-10-11-12-13-11-14-15
7. 1-2-3-4-7-9-10-11-14-16-17

**CASI di TEST**

**…..**

## JUnit – Test di Unità

[Opzionale] Riportare a scopo esemplificativo alcuni casi di utilizzo di **JUnit** per testare alcune classi del progetto, il framework di testing di unità automatizzato per il linguaggio di programmazione Java.

## Test funzionale

Segue una descrizione in forma tabellare dei risultati dell’esecuzione dei test funzionali precedentemente pianificati.

|  |  |  |
| --- | --- | --- |
| **REGISTRAZIONE** | | |
| **TEST SUITE** | | | | | | | | | | | |
| **Test Case ID** | **Descrizione** | **Classi di equivalenza coperte** | | **Pre-condizioni** | **Input** | **Output attesi** | **Post-condizioni attese** | **Output ottenuti** | **Post-condizioni ottenute** | **Esito (FAIL, PASS)** |
| 1 | Tutti input validi | Nome valido  Cognome valido  Patente valida  TipoPatente valido  Credito valido | |  | {Nome: “Mario”, Cognome: “Rossi”, Patente: “NAH68I903B”, TipoPatente: “B1”, Credito “230.50”} | Cliente registrato | Si ricevono le credenziali di accesso per email | Registrazione avvenuta con successo, controlla la mail per visualizzare le credenziali di accesso | Il Cliente si è registrato e ha ottenuto l’accesso alla piattaforma | **PASS** |
| 2 | Nome stringa > 40 caratteri | Nome stringa > 40 caratteri [ERROR],  Cognome, Patente, TipoPatente, Credito validi | |  | {Nome: “Marioooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo”, Cognome: “Rossi”, Patente: “NAH68I903B”, TipoPatente: “B1”, Credito “230.50”} | Nome troppo lungo! |  | Errore, il nome inserito è troppo lungo! |  | **PASS** |
| 3 | Nome stringa con simboli | Nome stringa con simboli [ERROR], Cognome, Patente, TipoPatente, Credito validi | |  | {Nome: “Ma-r.o”, Cognome: “Rossi”, Patente: “NAH68I903B”, TipoPatente: “B1”, Credito “230.50”} | Formato Nome errato! |  | Errore, il formato del nome inserito non è valido! |  | **PASS** |
|  | …… |  | |  |  |  |  |  |  |  |
| 12 | Credito numero con simboli | Nome, Cognome, Patente, TipoPatente validi, Credito numero con simboli [ERROR] | |  | {Nome: “Mario”, Cognome: “Rossi”, Patente: “NAH68I903B”, TipoPatente: “B1”, Credito “7@.1!”} | Formato Credito errato! |  | Errore, il formato del credito inserito non è valido! |  | **PASS** |