

Università degli Studi di Napoli Federico II

**Corso di Laurea in Ingegneria Informatica**

**Corso di Ingegneria del Software**

**Prof. A.R. Fasolino - A.A. 2024 - 25**

******

***TEMPLATE PER IL PROGETTO   
DI INGEGNERIA DEL SOFTWARE***

***“Piattaforma di Condivisione di Ricette”***

**INDICE**

[1. Specifiche informali 1](#_Toc195300113)

[2. Analisi e specifica dei requisiti 2](#_Toc195300114)

[2.1 Analisi nomi-verbi 2](#_Toc195300115)

[2.2 Revisione dei requisiti 3](#_Toc195300116)

[2.3 Glossario dei termini 3](#_Toc195300117)

[2.4 Classificazione dei requisiti 3](#_Toc195300118)

[2.4.1 Requisiti funzionali 4](#_Toc195300119)

[2.4.2 Requisiti sui dati 4](#_Toc195300120)

[2.4.3 Vincoli / Altri requisiti 5](#_Toc195300121)

[2.5 Modellazione dei casi d’uso 6](#_Toc195300122)

[2.5.1 Attori e casi d’uso 6](#_Toc195300123)

[2.5.2 Diagramma dei casi d’uso 7](#_Toc195300124)

[2.5.3 Scenari 7](#_Toc195300125)

[2.6 Diagramma delle classi 8](#_Toc195300126)

[2.6.1 Responsabilità **Errore. Il segnalibro non è definito.**](#_Toc195300127)

[2.7 Diagrammi di sequenza 10](#_Toc195300128)

[2.7.1 Registrazione **Errore. Il segnalibro non è definito.**](#_Toc195300129)

[2.7.2 Accesso 10](#_Toc195300130)

[2.7.3 AggiungiAuto 10](#_Toc195300131)

[3. Piano di test funzionale 11](#_Toc195300132)

[3.1 Registrazione 11](#_Toc195300133)

[4. Progettazione 13](#_Toc195300134)

[4.1 Diagramma delle classi 13](#_Toc195300135)

[4.1.1 Traduzione classi ed associazioni 13](#_Toc195300136)

[4.1.2 Pattern BCED 13](#_Toc195300137)

[4.1.2.1 Package Boundary 13](#_Toc195300138)

[4.1.2.2 Package Controller 13](#_Toc195300139)

[4.1.2.3 Package Entity 14](#_Toc195300140)

[4.1.2.4 Package Database 14](#_Toc195300141)

[4.2 Diagrammi di sequenza 14](#_Toc195300142)

[4.2.1 Registrazione 15](#_Toc195300143)

[4.2.2 Accesso **Errore. Il segnalibro non è definito.**](#_Toc195300144)

[5. Implementazione 16](#_Toc195300145)

[5.1 Package Database 16](#_Toc195300146)

[5.2 Package Entity 16](#_Toc195300147)

[5.3 Package Controller 16](#_Toc195300148)

[5.4 Package Boundary 16](#_Toc195300149)

[5.5 Package DTO 16](#_Toc195300150)

[5.6 Diagramma di Deployment 17](#_Toc195300151)

[6. Testing 18](#_Toc195300152)

[6.1 Test strutturale 18](#_Toc195300153)

[6.1.1 Complessità ciclomatica 18](#_Toc195300154)

[6.1.1.1 inserisciAutoModifiche – GestioneParcoAuto 18](#_Toc195300155)

[6.2 JUnit – Test di Unità 20](#_Toc195300156)

[6.3 Test funzionale 21](#_Toc195300157)

# Specifiche informali

*Riportare la Traccia assegnata*

*Si desidera sviluppare un sistema software per la gestione di una piattaforma sociale dedicata alla condivisione di ricette culinarie semplici e veloci. La piattaforma è pensata per permettere agli utenti di pubblicare le proprie ricette, organizzarle in raccolte tematiche e interagire con i contenuti pubblicati dagli altri utenti attraverso apprezzamenti e commenti.*

*Il sistema consente la registrazione degli utenti tramite l’inserimento di e-mail, nome, cognome e password. Ogni utente dispone di un profilo personale modificabile che include i propri dati anagrafici, una breve biografia ed eventualmente un’immagine del profilo. Dopo l’autenticazione, l’utente può pubblicare nuove ricette tramite un’apposita interfaccia grafica. Ogni ricetta è caratterizzata da un titolo, una lista di ingredienti, una descrizione dei passaggi (con un limite massimo di 800 caratteri), un tempo stimato di preparazione espresso in minuti, uno o più tag tematici (ad esempio “vegetariana”, “dolci”, “pranzo veloce”). L’utente può scegliere se aggiungere la ricetta a una raccolta già esistente oppure crearne una nuova al momento della pubblicazione.*

*Ogni raccolta è composta da ricette dello stesso autore, è identificata da un titolo e da una descrizione e può contenere ricette pubbliche o private. Le ricette pubbliche sono visibili nel feed principale degli utenti.*

*Una volta autenticato, ciascun utente può visualizzare nel proprio feed personale un insieme di nuove ricette pubblicate da altri utenti.* *Il feed mostra al massimo cinque ricette, ordinate in ordine cronologico decrescente, selezionate tra le ultime pubblicazioni effettuate da altri autori.*

*Gli utenti hanno la possibilità di interagire con le ricette pubblicate da altri tramite un sistema di apprezzamenti (rappresentati da un "like") e con l’aggiunta di commenti testuali. Ogni commento riporta l’autore, il testo e la data di pubblicazione. Quando una ricetta viene visualizzata, il sistema mostra anche il numero totale di like ricevuti e i tre commenti più recenti associati.*

*La piattaforma offre a ogni utente una sezione dedicata alle statistiche personali. In tale sezione l’utente può visualizzare il numero complessivo di like ricevuti dalle proprie ricette, il numero totale di commenti ottenuti e la ricetta più apprezzata tra quelle pubblicate, ossia quella con il numero più alto di like.*

*Gli amministratori della piattaforma, previa autenticazione, possono accedere a funzionalità di reportistica. Tali funzionalità consentono di generare report contenenti dati aggregati come il numero totale di ricette pubblicate in un determinato intervallo temporale, l’elenco degli autori più attivi in base al numero di ricette inserite, i tag tematici più utilizzati e le ricette con il maggior numero di interazioni.*

*Il sistema dovrà essere accessibile sia da dispositivi desktop che mobili,* *prevedere un sistema di notifiche in tempo reale per segnalare all’utente l’arrivo di nuovi commenti o apprezzamenti, garantire la protezione dei dati personali, la sicurezza delle informazioni archiviate e l’adozione di un meccanismo di autenticazione robusto e affidabile.*

# Analisi e specifica dei requisiti

## Analisi nomi-verbi

Il sistema consente la registrazione degli utenti tramite l’inserimento di e-mail, nome, cognome e password. Ogni utente dispone di un profilo personale modificabile che include i propri dati anagrafici, una breve biografia ed eventualmente un’immagine del profilo. Dopo l’autenticazione, l’utente può pubblicare nuove ricette tramite un’apposita interfaccia grafica. Ogni ricetta è caratterizzata da un titolo, una lista di ingredienti, una descrizione dei passaggi (con un limite massimo di 800 caratteri), un tempo stimato di preparazione espresso in minuti, uno o più tag tematici (ad esempio “vegetariana”, “dolci”, “pranzo veloce”). L’utente può scegliere se aggiungere la ricetta a una raccolta già esistente oppure crearne una nuova al momento della pubblicazione.

Ogni raccolta è composta da ricette dello stesso autore, è identificata da un titolo e da una descrizione e può contenere ricette pubbliche o private. Le ricette pubbliche sono visibili nel feed principale degli utenti.

Una volta autenticato, ciascun utente può visualizzare nel proprio feed personale un insieme di nuove ricette pubblicate da altri utenti. Il feed mostra al massimo cinque ricette, ordinate in ordine cronologico decrescente, selezionate tra le ultime pubblicazioni effettuate da altri autori.

Gli utenti hanno la possibilità di interagire con le ricette pubblicate da altri tramite un sistema di apprezzamenti (rappresentati da un "like") e con l’aggiunta di commenti testuali. Ogni commento riporta l’autore, il testo e la data di pubblicazione. Quando una ricetta viene visualizzata, il sistema mostra anche il numero totale di like ricevuti e i tre commenti più recenti associati.

In tale sezione l’utente può visualizzare il numero complessivo di like ricevuti dalle proprie ricette, il numero totale di commenti ottenuti e la ricetta più apprezzata tra quelle pubblicate, ossia quella con il numero più alto di like. La piattaforma offre a ogni utente una sezione dedicata alle statistiche personali

Gli amministratori della piattaforma, previa autenticazione, possono accedere a funzionalità di reportistica. Tali funzionalità consentono di generare report contenenti dati aggregati come il numero totale di ricette pubblicate in un determinato intervallo temporale, l’elenco degli autori più attivi in base al numero di ricette inserite, i tag tematici più utilizzati e le ricette con il maggior numero di interazioni.

Il sistema dovrà essere accessibile sia da dispositivi desktop che mobili, prevedere un sistema di notifiche in tempo reale per segnalare all’utente l’arrivo di nuovi commenti o apprezzamenti, garantire la protezione dei dati personali, la sicurezza delle informazioni archiviate e l’adozione di un meccanismo di autenticazione robusto e affidabile.

*LEGENDA:  
Classe  
Attributo  
Funzionalità  
Attore*

*Classe-Attore*

## Revisione dei requisiti

1. Il sistema deve consentire la registrazione degli utenti.
2. Ogni Utente è caratterizzato da e-mail, nome, cognome, password.
3. Il sistema deve permette a ogni utente di creare un profilo.
4. Il sistema deve permettere all’utente di apportare modifiche al profilo.
5. Ogni profilo personale include i dati anagrafici, una breve biografia ed eventualmente un’immagine del profilo.
6. Il sistema deve avere un sistema di autenticazione
7. Il sistema deve avere un’interfaccia grafica per la pubblicazione di una nuova ricetta da parte di un utente
8. Ogni Ricetta è caratterizzata da un titolo, una lista di ingredienti, una descrizione dei passaggi, un tempo stimato, uno o più tag tematici.
9. Il sistema deve consentire di inserire una nuova ricetta in una raccolta esistente.
10. Il sistema deve consentire di creare una Raccolta.
11. Ogni Raccolta è caratterizzata da un autore, un titolo, una descrizione e un insieme di ricette.
12. Ogni ricetta può essere pubblica o privata all’interno di una raccolta.
13. Il sistema deve consentire ad ogni Utente di visualizzare un insieme di nuove ricette pubblicate da altri utenti (feed).
14. Ogni feed può mostrare al massimo cinque ricette.
15. Il sistema deve avere un sistema di ordinamento delle ricette.
16. Il sistema deve consentire agli utenti l'interazione con le ricette pubblicate da altri tramite un sistema di apprezzamenti (like).
17. Il sistema deve permettere l’aggiunta di commenti testuali.
18. Ogni Commento è caratterizzato da autore, testo e data di pubblicazione.
19. Il sistema deve mostrare per ricetta il numero totale di like ricevuti e i tre commenti più recenti.
20. Il sistema deve mostrare per ogni utente le sue statistiche.
21. Il sistema deve mostrare il numero commenti ricevuti per ogni ricetta
22. Il sistema deve permettere ad utenti amministratori di generare reportistica
23. I report possono contenere: numero totale di ricette pubblicate in un determinato intervallo temporale, l’elenco degli autori più attivi in base al numero di ricette inserite, i tag tematici più utilizzati e le ricette con il maggior numero di interazioni.
24. Il sistema dove essere accessibile sia da dispositivi desktop che mobili.
25. Il sistema deve prevedere un meccanismo di notifiche in tempo reale.
26. Il sistema deve garantire la protezione dei dati personali e la sicurezza delle informazioni.

## Glossario dei termini

[OPZIONALE] Riportare un glossario dei termini in una tabella termine-descrizione (significato)-eventuali sinonimi, come nel seguente esempio.

|  |  |  |
| --- | --- | --- |
| **Termine** | **Descrizione** | **Sinonimi** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Classificazione dei requisiti

Classificare i requisiti (funzionali, sui dati, vincoli/altri requisiti), riportando le frasi dei requisiti (revisionati) che li esprimono (ciascuna frase deve esprimere un singolo requisito), numerandoli secondo una codifica (es.: RF01, RF02, RFn per i requisiti funzionali; RD01, RD02, RDm per i requisiti sui dati). Verificare che i requisiti siano chiari, completi e verificabili.

### Requisiti funzionali

|  |  |  |
| --- | --- | --- |
| **ID** | **Requisito** | **Origine (n. frase dei requisiti revisionati)** |
| **RF01** | Il sistema deve consentire la registrazione degli utenti | 1 |
| **RF02** | Il sistema deve permette a ogni utente di creare un profilo | 3 |
| **RF03** | Il sistema deve permettere all’utente di apportare modifiche al profilo | 4 |
| **RF05** | Il sistema deve avere un’interfaccia grafica per la pubblicazione di una nuova ricetta da parte di un utente | 7 |
| **RF06** | Il sistema deve consentire di inserire una nuova ricetta in una raccolta esistente | 9 |
| **RF07** | Il sistema deve consentire di creare una Raccolta | 10 |
| **RF08** | Il sistema deve consentire ad ogni Utente di visualizzare un insieme di nuove ricette pubblicate da altri utenti(feed) | 13 |
| **RF09** | Il sistema deve avere un sistema di ordinamento delle ricette | 15 |
| **RF10** | Il sistema deve consentire agli utenti l'interazione con le ricette pubblicate da altri tramite un sistema di apprezzamenti (like) | 16 |
| **RF11** | Il sistema deve permettere l’aggiunta di commenti testuali | 17 |
| **RF12** | Il sistema deve mostrare per ricetta il numero totale di like ricevuti e i tre commenti più recenti | 19 |
| **RF13** | Il sistema deve mostrare per ogni utente le sue statistiche | 20 |
| **RF14** | Il sistema deve mostrare il numero commenti ricevuti per ogni ricetta | 21 |
| **RF15** | Il sistema deve permettere ad utenti amministratori di generare reportistica | 22 |
| **RF16** | Il sistema deve prevedere un meccanismo di notifiche in tempo reale | 25 |

### Requisiti sui dati

|  |  |  |
| --- | --- | --- |
| **ID** | **Requisito** | **Origine (n. frase dei requisiti revisionati)** |
| **RD01** | Di ogni Utente si vuole memorizzare e-mail, nome, cognome, password | 2 |
| **RD02** | Ogni profilo personale include i dati anagrafici, una breve biografia ed eventualmente un’immagine del profilo | 5 |
| **RD03** | Ogni Ricetta è caratterizzata da un titolo, una lista di ingredienti, una descrizione dei passaggi, un tempo stimato, uno o più tag tematici | 8 |
| **RD04** | Ogni Ricetta può essere pubblica o privata all’interno di una raccolta | 12 |
| **RD05** | Ogni Raccolta è caratterizzata da un autore, un titolo, una descrizione e un insieme di ricette | 11 |
| **RD06** | Ogni Commento è caratterizzato da autore, testo e data di pubblicazione | 18 |
| **RD07** | I report possono contenere: numero totale di ricette pubblicate in un determinato intervallo temporale, l’elenco degli autori più attivi in base al numero di ricette inserite, i tag tematici più utilizzati e le ricette con il maggior numero di interazioni. | 23 |

### Vincoli / Altri requisiti

|  |  |  |
| --- | --- | --- |
| **ID** | **Requisito** | **Origine (n. frase dei requisiti revisionati)** |
| **V01** | Il sistema deve avere un sistema di autenticazione | 6 |
| **V02** | Ogni feed può mostrare al massimo cinque ricette | 14 |
| **V03** | Il sistema dove essere accessibile sia da dispositivi desktop che mobili | 24 |
| **V04** | Il sistema deve garantire la protezione dei dati personali e la sicurezza delle informazioni | 26 |

## Modellazione dei casi d’uso

### Attori e casi d’uso

***Attori Primari:***

* Utente
* Amministratore
* Sezione ?????

***Attori Secondari:***

* Servizio di autenticazione Google
* Servizio di reCAPTCHA

***Casi d’uso:***

* **UC1**: Registrazione
* **UC2**: Accesso
* **UC3**: PrenotaAuto
* **UC4**: AggiungiAuto
* **….**

***Casi d’uso di inclusione:***

* **UC12**: VisualizzaAutoDisponibili
* **UC13**: VerificaPatente
* **UC14**: VerificaCredito

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Caso d’uso | Attori Primari | Attori Secondari | Incl. / Ext. | Requisiti corrispondenti |
| UC1: Registrazione | Utente | MotorizzazioneCivile, ServizioMessaggistica | - | RF01 |
| UC2: Accesso | Cliente | - | - | RF02 |
| UC3: PrenotaAuto | Cliente | - | Include VisualizzaAutoDisponibili, VerificaPatente e VerificaCredito | RF03 |
| UC4: AggiungiAuto | Gestore | - | - | RF10 |
| UC5: RimuoviAuto | Gestore | - | - | RF11 |
| UC6: ModificaAuto | Gestore | - | - | RF12 |
| UC7: GeneraReport | Gestore | - | Generalizzazione di GeneraReportSettimana, GeneraReportMese e GeneraReportCliente | RF13, RF14 |
| UC8: GeneraReportMese | Gestore | - | - | RF13 |
| UC9: GeneraReportSettimana | Gestore | - | - | RF13 |
| UC10: GeneraReportCliente | Gestore | - | - | RF14 |
| UC11: AggiornaCredito | Cliente |  |  | RF15 |
| UC12: VisualizzaAutoDisponibili | Cliente | - | Incluso in PrenotaAuto | RF03 |
| UC13: VerificaPatente | - | - | Incluso in PrenotaAuto | RF06 |
| UC14: VerificaCredito | - | - | Incluso in PrenotaAuto | RF07, RF08 |
| UC15: InvioConferma | - | ServizioMessaggistica | Incluso in PrenotaAuto | RF05 |

* **UC15**: InvioConferma

### Diagramma dei casi d’uso

Riportare il diagramma dei casi d’uso.

Immagine che contiene diagramma, schermata, testo, Diagramma

Descrizione generata automaticamente

### 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** |