DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE

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

"SISTEMA SOFTWARE PER GESTIONE NOLEGGIO AUTO ONLINE"

INDICE

1. Specifiche informali

Riportare la Traccia assegnata

Si intende sviluppare un sistema software per la gestione di un servizio di noleggio auto online.

Il progetto prevede la creazione di un sistema software per gestire un servizio di noleggio auto online. Il sistema consentirà ai clienti registrati di effettuare prenotazioni, mentre ai gestori del servizio fornirà gli strumenti per amministrare il parco auto, gestire le prenotazioni e le tariffe.

Il sistema deve consentire la registrazione e l'accesso degli utenti, permettendo loro di visualizzare un elenco di auto disponibili per il noleggio, con informazioni dettagliate come modello, cilindrata, costo giornaliero e tipo di patente richiesta per la guida (tipo A, B, C, etc.) e targa. Un cliente è caratterizzato da nome, cognome, patente (e sua relativa tipologia) e credito disponibile.

Attraverso una interfaccia grafica, un cliente può visualizzare l'elenco delle auto disponibili al momento della richiesta. I clienti potranno prenotare un'auto specificando data e ora di inizio e fine del noleggio, e ricevere conferma della prenotazione.
In fase di prenotazione, il sistema dovrà verificare la corrispondenza tra la patente del cliente e il tipo di veicolo selezionato per assicurare che il cliente sia abilitato alla guida di quel veicolo.
Il sistema deve calcolare il costo totale del noleggio in base alla durata e al costo giornaliero dell'auto selezionata e verificare che il cliente disponga di credito sufficiente per coprire tale costo.
In caso di esito positivo dei controlli, il sistema dovrà ancora aggiornare lo stato dell'automobile a non disponibile per tutto il periodo del noleggio richiesto.

È necessario che i gestori possano aggiungere, rimuovere o modificare le informazioni sulle auto del parco auto, inclusi cilindrata, costo giornaliero e disponibilità.
Il sistema deve permettere agli amministratori di includere funzioni per la generazione di report settimanali e mensili sulle prenotazioni, guadagni e numero medio di noleggi da parte dei clienti registrati in un dato periodo di tempo.

2. Analisi e specifica dei requisiti

2.1 Analisi nomi-verbi

Il sistema deve consentire la registrazione e l'accesso degli utenti, permettendo loro di visualizzare un elenco di auto disponibili per il noleggio, con informazioni dettagliate come modello, cilindrata, costo giornaliero e tipo di patente richiesta per la guida (tipo A, B, C, etc.) e targa. Un cliente è caratterizzato da nome, cognome, patente (e sua relativa tipologia) e credito disponibile.

Attraverso una interfaccia grafica, un cliente può visualizzare l'elenco delle auto disponibili al momento della richiesta. I clienti potranno prenotare un'auto specificando data e ora di inizio e fine del noleggio, e ricevere conferma della prenotazione.
In fase di prenotazione, il sistema dovrà verificare la corrispondenza tra la patente del cliente e il tipo di veicolo selezionato per assicurare che il cliente sia abilitato alla guida di quel veicolo. Il sistema deve calcolare il costo totale del noleggio in base alla durata e al costo giornaliero dell'auto selezionata e verificare che il cliente disponga di credito sufficiente per coprire tale costo. In caso di esito positivo dei controlli, il sistema dovrà ancora aggiornare lo stato dell'automobile a non disponibile per tutto il periodo del noleggio richiesto.

È necessario che i gestori possano aggiungere, rimuovere o modificare le informazioni sulle auto del parco auto, inclusi cilindrata, costo giornaliero e disponibilità.
Il sistema deve permettere agli amministratori di includere funzioni per la generazione di report settimanali e mensili sulle prenotazioni, guadagni e numero medio di noleggi da parte dei clienti registrati in un dato periodo di tempo.

Parco auto = Autonoleggio.

LEGENDA:
Classe
Attributo
Funzionalità
Attore
Classe-Attore

2.2 Revisione dei requisiti

Riportare la traccia revisionata, a seguito di una interazione (simulata) con gli *stakeholders*, che produce chiarimenti, disambiguazione di termini, ristrutturazione delle frasi che esprimono più requisiti, separazione delle frasi che esprimono requisiti funzionali e requisiti sui dati, etc.
Nella revisione, frasi lunghe, frasi che esprimono più di un requisito, frasi ambigue, etc., vanno opportunamente separate, riformulate, disambiguate. Esprimere i requisiti funzionali con frasi del tipo "Il sistema deve …"

Esempio:

1. *Il sistema deve offrire all'Utente una funzionalità per registrarsi.*
2. *Il sistema deve offrire al Cliente una funzionalità per accedere.*
3. *Il sistema deve offrire al Cliente una funzionalità per visualizzare un elenco di auto disponibili per il noleggio.*
4. *Di ogni Auto si vuole memorizzare modello, cilindrata, costo giornaliero, disponibilità e tipo di patente richiesta per la guida.*
5. *Di ogni Cliente si vuole memorizzare nome, cognome, patente, con relativa tipologia, e credito disponibile.*
6. *Il sistema deve offrire al Cliente una funzionalità per prenotare un'auto.*
7. *Il Cliente, per prenotare un Noleggio, deve specificare data e ora di inizio e fine.*
8. *Il sistema deve inviare un messaggio di conferma al Cliente, dopo una prenotazione.*
9. *Il sistema deve verificare la corrispondenza tra la patente del Cliente e il tipo di veicolo selezionato per assicurare che il cliente sia abilitato alla guida di quel veicolo.*
10. *XXX*
11. *……*
12. *…….*

1.1 Glossario dei termini

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

Termine	Descrizione	Sinonimi
Auto	Una delle auto facenti parte del parco auto	Automobile, Autovettura
Parco Auto	Insieme delle auto a disposizione dei clienti per il noleggio	Autonoleggio
Utente	Una generica persona che intende registrarsi per fruire del servizio di noleggio	
……..	…….	

2.3 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.

2.3.1 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 al Cliente una funzionalità per accedere	2
RF03	Il sistema deve offrire al Cliente una funzionalità per visualizzare un elenco di auto disponibili per il Noleggio	3
RF04	Il sistema deve offrire al Cliente una funzionalità per prenotare un'auto	6
RF05	Il sistema deve inviare un messaggio di conferma al Cliente, dopo una prenotazione	8
….	……	

1.1.1 Requisiti sui dati

ID	Requisito	Origine (n. frase dei requisiti revisionati)
RD01	Di ogni Auto si vuole memorizzare modello, cilindrata, costo giornaliero e tipo di patente richiesta per la guida	4
RD02	Di ogni Cliente si vuole memorizzare nome, cognome, patente, con relativa tipologia, e credito disponibile	5
…..	…….	

1.1.2 Vincoli / Altri requisiti

ID	Requisito	Origine (n. frase dei requisiti revisionati)
V01	Per l'invio della conferma di prenotazione, deve essere disponibile un sistema di messaggistica esterno al sistema	
V02	Per certificare la validità della patente inserita dall'Utente in fase di registrazione, deve essere disponibile il sistema della Motorizzazione Civile	

2.4 Modellazione dei casi d'uso

2.4.1 Attori e casi d'uso

Attori Primari:

- Utente
- Cliente
- Gestore

Attori Secondari:

- Servizio di messaggistica
- Sistema Motorizzazione Civile

Casi d'uso:

- UC1: Registrazione
- UC2: Accesso
- UC3: PrenotaAuto
- UC4: AggiungiAuto
- ....
-

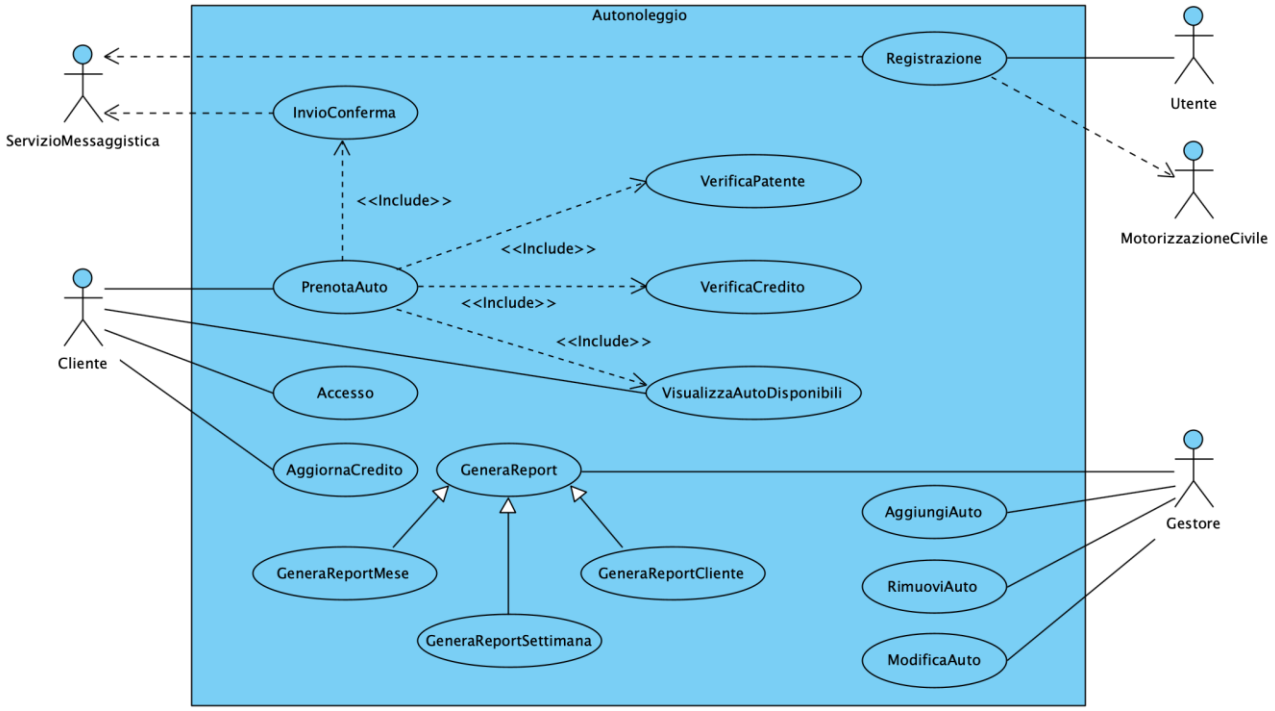
Casi d'uso di inclusione:

- UC12: VisualizzaAutoDisponibili
- UC13: VerificaPatente
- UC14: VerificaCredito
- UC15: InvioConferma

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

1.1.3 Diagramma dei casi d'uso

Riportare il diagramma dei casi d'uso.



1.1.4 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 5.1.Il Cliente seleziona l'Auto di suo gradimento 5.2.Il sistema verifica che la patente del Cliente sia idonea alla guida del veicolo di interesse *5.3.<<include>> VerificaPatente*

	5.4.**if** le verifiche sono andate a buon fine 5.4.1. Il sistema verifica che il Cliente abbia la disponibilità sufficiente per il noleggio 5.4.2. <<*include*>> *VerificaCredito* 5.4.3. **if** le verifiche sono andate a buon fine 5.4.3.1. Il sistema procede con la prenotazione 5.4.3.2. Il sistema aggiunge il Noleggio all'archivio dei Noleggi 5.4.3.3. Il sistema invia un messaggio di conferma dell'avvenuta prenotazione al Cliente 5.4.3.4. <<*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

2.5 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.
.....

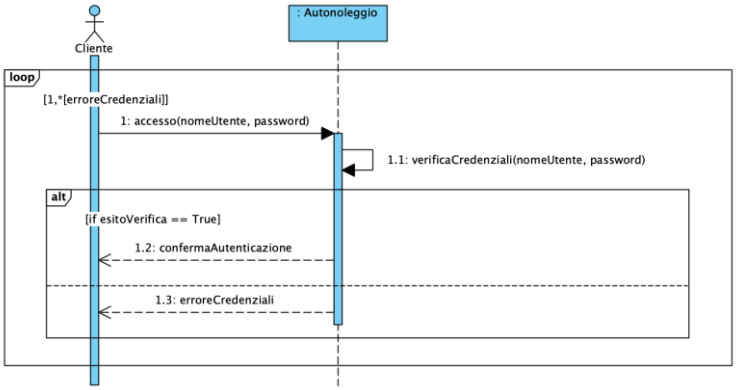
2.6 Diagrammi di sequenza

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.

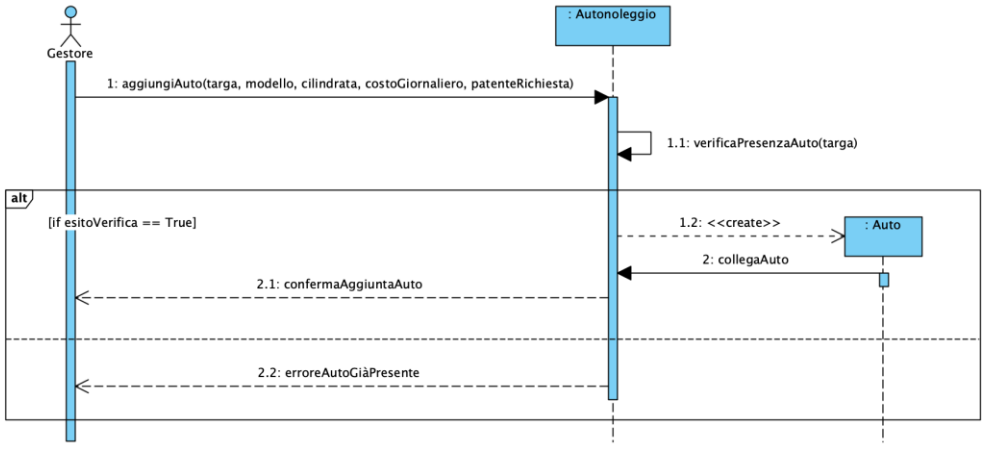
Si riportano alcuni esempi di diagrammi di sequenza di analisi per i casi d'uso precedentemente individuati.

2.6.1 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.



2.6.2 AggiungiAuto



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.

……..

2.7 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

3. 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***.

3.1 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: $3 * 3 * 4 * 3 * 3 = 324$.
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: "Mariooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo", 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: "Rossiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii", 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					

4. Progettazione

4.1 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

4.1.1 Traduzione classi ed associazioni

Spiegare le scelte effettuate

4.1.2 Pattern BCED

4.1.2.1 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.

1.

Riportare il Package Boundary

4.1.2.2 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

4.1.2.3 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

4.1.2.4 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.

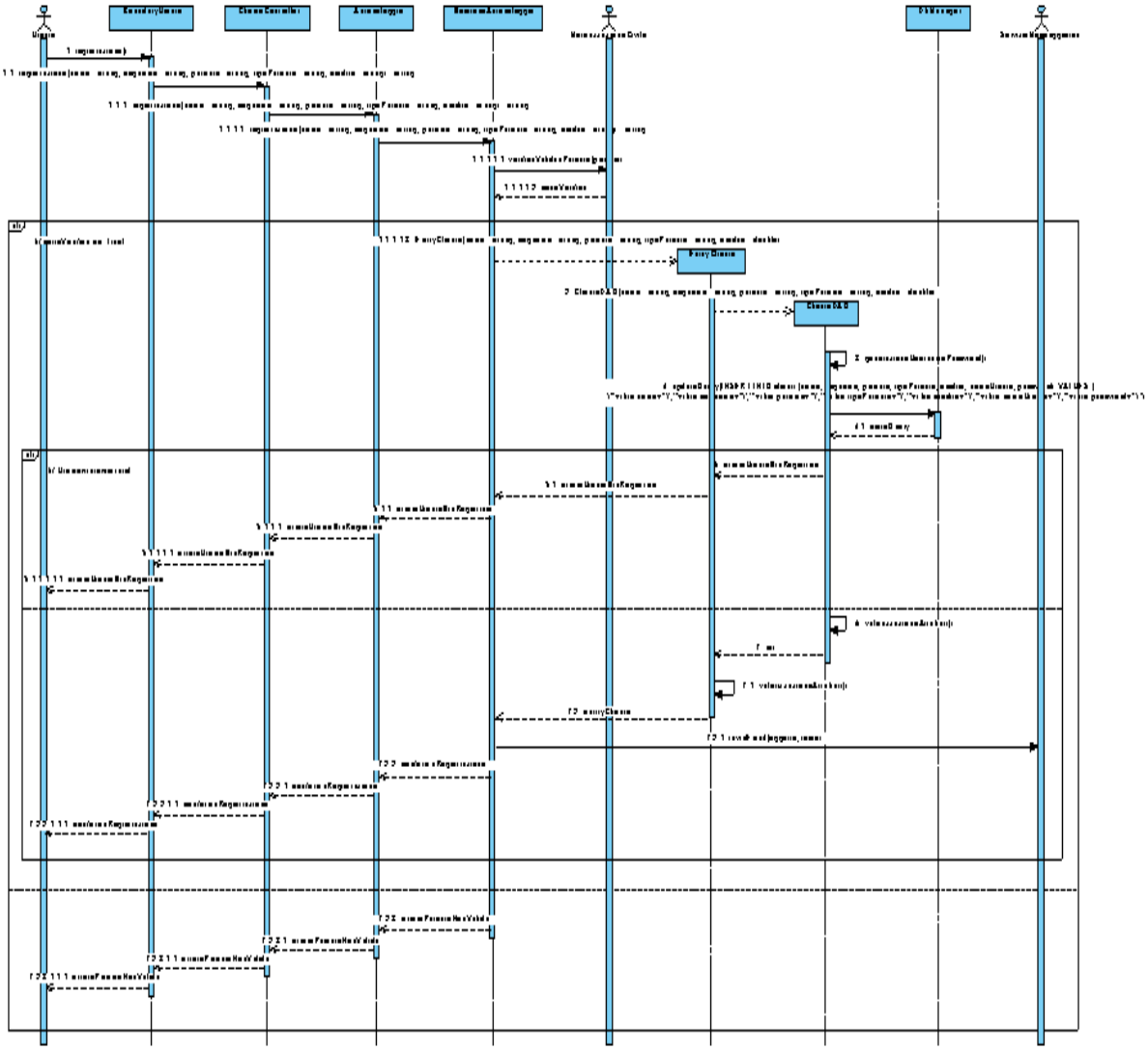
Ognuna di queste classi conterrà i metodi CRUD per l'interrogazione e la manipolazione della corrispondente classe di dominio (*query*), implementati in funzione di un'ulteriore classe, ***DBManager***, che costituisce di fatto l'unico punto di accesso vero e proprio al DB, sfruttando i metodi che questa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

Riportare il Package Database

4.2 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.

4.2.1 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

5. 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)

5.1 Package Database

TBD

5.2 Package Entity

TBD

5.3 Package Controller

TBD

5.4 Package Boundary

TBD

5.5 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.

5.6 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.

- .

2. Testing

5.7 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.

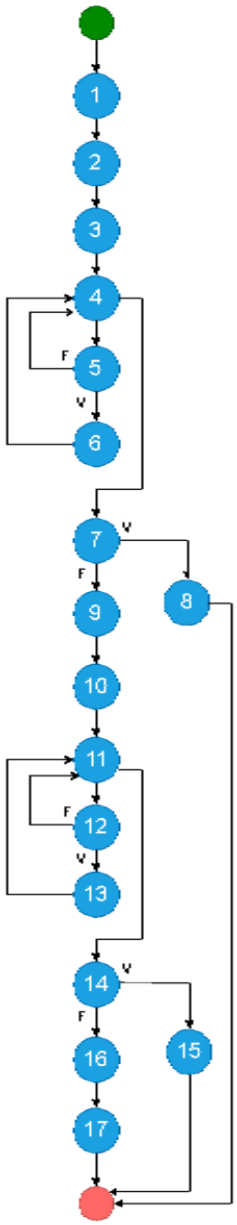
5.7.1 Complessità ciclomatica

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.

5.7.1.1 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

.....

5.8 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.

2.1 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: “Mariooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo”, 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**