PROGETTO INGEGNERIA DEL SOFTWARE

Applicazione Noleggi Auto

IL TEAM

BENIAMINO
INFANTE

IBRAHIMA
SARR

ALEXANDER
RUBINO

L'OBIETTIVO

La nostra idea è quella di realizzare un applicativo che gestisca i noleggi di auto.

Vi sono 3 tipi di noleggi:

- **Car Sharing** (noleggio giornaliero o weekend)
- **Noleggio di breve periodo** (noleggio di massimo 3 mesi)
- **Noleggio di lungo periodo** (noleggio di massimo un anno)

Viene inoltre offerta la possibilità di scegliere tra 3 tipi di auto:

- **Utilitarie**
- **Business**
- **Luxury**

I clienti potranno inoltre iscriversi al **programma fedeltà** con cui è possibile raccogliere punti in base ai km percorsi e alle condizioni in cui viene riconsegnata l'auto

LE DIFFICOLTA' INCONTRATE

Per la realizzazione di questo progetto abbiamo incontrato diverse difficoltà di vario tipo:

DIFFICOLTA' TEMPORALI

Ciascuno di noi, durante questi mesi è stato impegnato con diverse attività tra cui lavoro ed altri esami, per cui incontrarci periodicamente ha rappresentato un problema per la realizzazione del progetto

CREAZIONE DEL FRONTEND

Abbiamo provato ad implementare un'interfaccia grafica per la nostra app ma abbiamo incontrato diverse difficoltà provando sia con Vaadin che con Angular

IL PARADIGMA DI PROGRAMMAZIONE E MODELLAZIONE

Si è scelto di utilizzare come paradigma di programmazione un paradigma **orientato agli oggetti**.
Di fatti abbiamo dato priorità alla produzione di modelli, quali il **class diagram**, da cui poi abbiamo ricavato lo scheletro del codice. Come linguaggio di programmazione abbiamo scelto di usare **JAVA**, con **SQL** per i DB.

I TOOLS UTILIZZATI

COME ABBIAMO USATO GITHUB

Github ci ha dato una mano con la fase di **Configuration Management**.

Prevalentemente ciascuno di noi svolgeva il proprio compito e poi creava una **pull request** per ottenere la conferma che le modifiche effettuate andassero bene.
Sono state create inoltre diverse **issues** per i medesimi motivi.
Ciascuno di noi ha anche lavorato sui **branch locali** al fine di testare dei cambiamenti "internamente" per poi riportarli sul **main branch**.

IL CICLO DI VITA DEL SOFTWARE

Il metodo che abbiamo deciso di utilizzare in fase di progettazione è un **metodo Agile**. In particolare abbiamo deciso di usare l'**Extreme Programming (XP)** in quanto si base su delle best practices come: design **semplice**, **refactoring** e **programmazione in coppia**.
XP è caratterizzato da **5 principi**:

FEEDBACK RAPIDO

Controllo e testing continui dei cambiamenti a parti di sistema prima di rilasciare al cliente

SEMPLICITA'

Si cerca sempre di tenere un design semplice, aggiungendo funzionalità solo quando richiesto ed il codice è spesso oggetto di refactoring

CAMBIAMENTO INCREMENTALE

Non si prendono decisioni improvvise, i cambiamenti vengono effettuati in maniera graduale

ABBRACCIARE IL CAMBIAMENTO

Non si fanno pianificazioni a lungo termine, i problemi più gravi vengono risolti il prima possibile.

LAVORO DI QUALITA'

Si cerca di offrire sempre un prodotto qualitativo

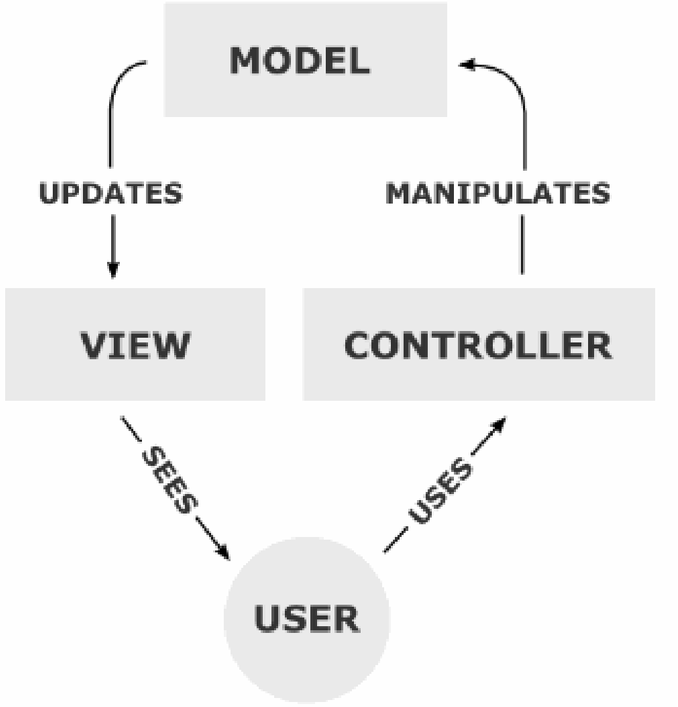
I REQUISITI DEL SOFTWARE

Abbiamo seguito i quattro steps principali per l'Ingegneria dei requisiti:

- **ELICITATION:** in questa fase abbiamo cercato di capire come realizzare l'app. Come? Ci siamo basati sull'analisi di altri sistemi già esistenti simili alla nostra idea
- **SPECIFICATION:** una volta compreso il problema abbiamo steso il project plan attraverso l'uso di linguaggio naturale e da lì abbiamo realizzato le tecniche di modellazione UML per descrivere al meglio il tutto
- **VALIDATION E VERIFICATION:** ci siamo accertati che tutti i requisiti siano stati correttamente riportati e verificate da entrambe le parti (sviluppatori e cliente)
- **NEGOTIATION:** in questa fase abbiamo concordato sui vincoli legati alla realizzazione dell'app tramite anche l'uso di Github

L'ARCHITETTURA DEL SOFTWARE

Per l'architettura del software abbiamo optato per un approccio di tipo **Model View Controller (MVC)**. Questo modello permette di gestire direttamente i dati, la logica e le regole dell'applicazione.
MVC è caratterizzato da **3 componenti**:



MODEL

Fornisce i metodi per accedere ai dati utili dell'applicazione

VIEW

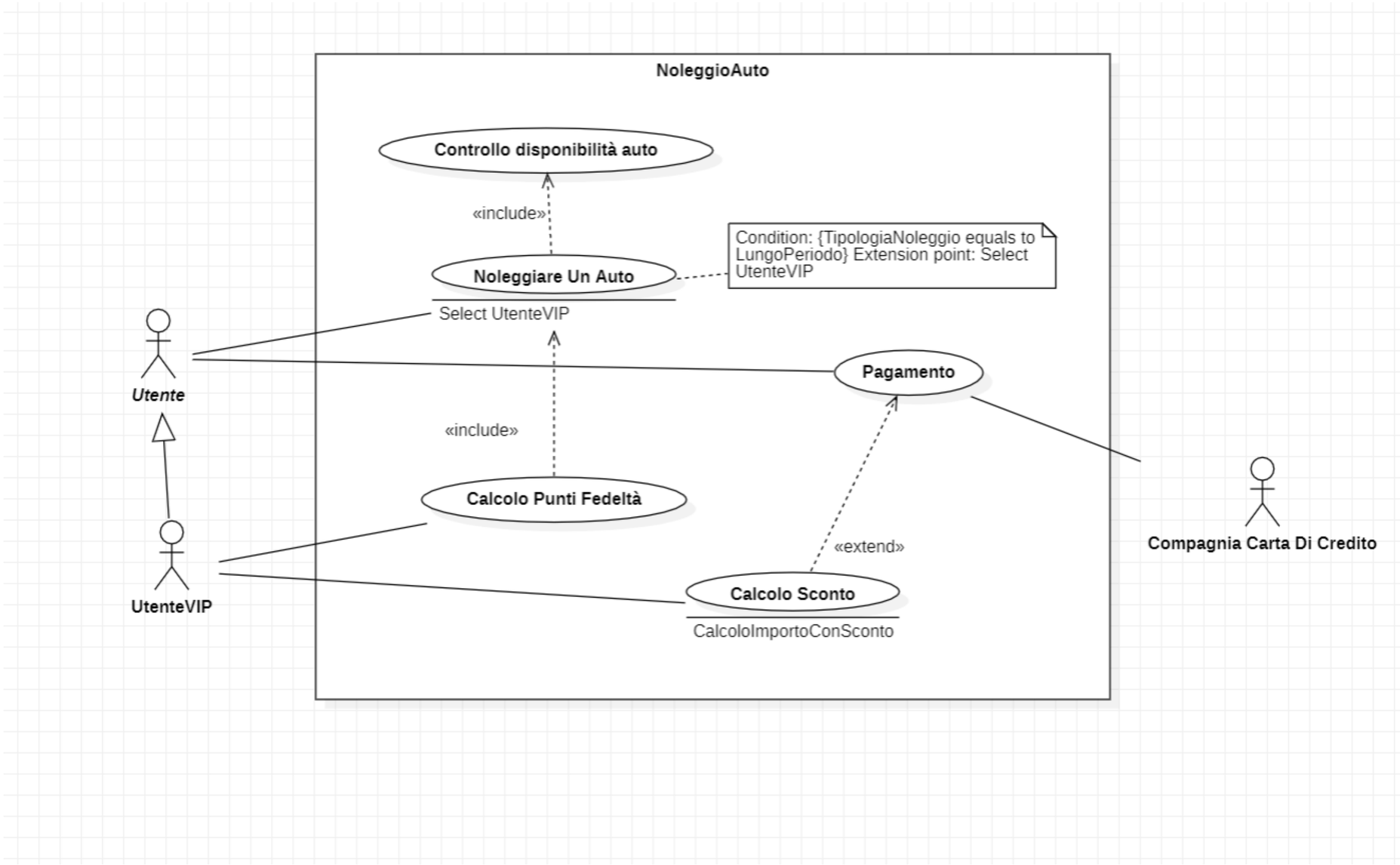
Visualizza i dati contenuti nel Model e si occupa dell'interazione con utenti e agenti.

CONTROLLER

Riceve i comandi dell'utente, generalmente dalla interfaccia grafica (View) e li attua modificando lo stato delle altre due componenti.

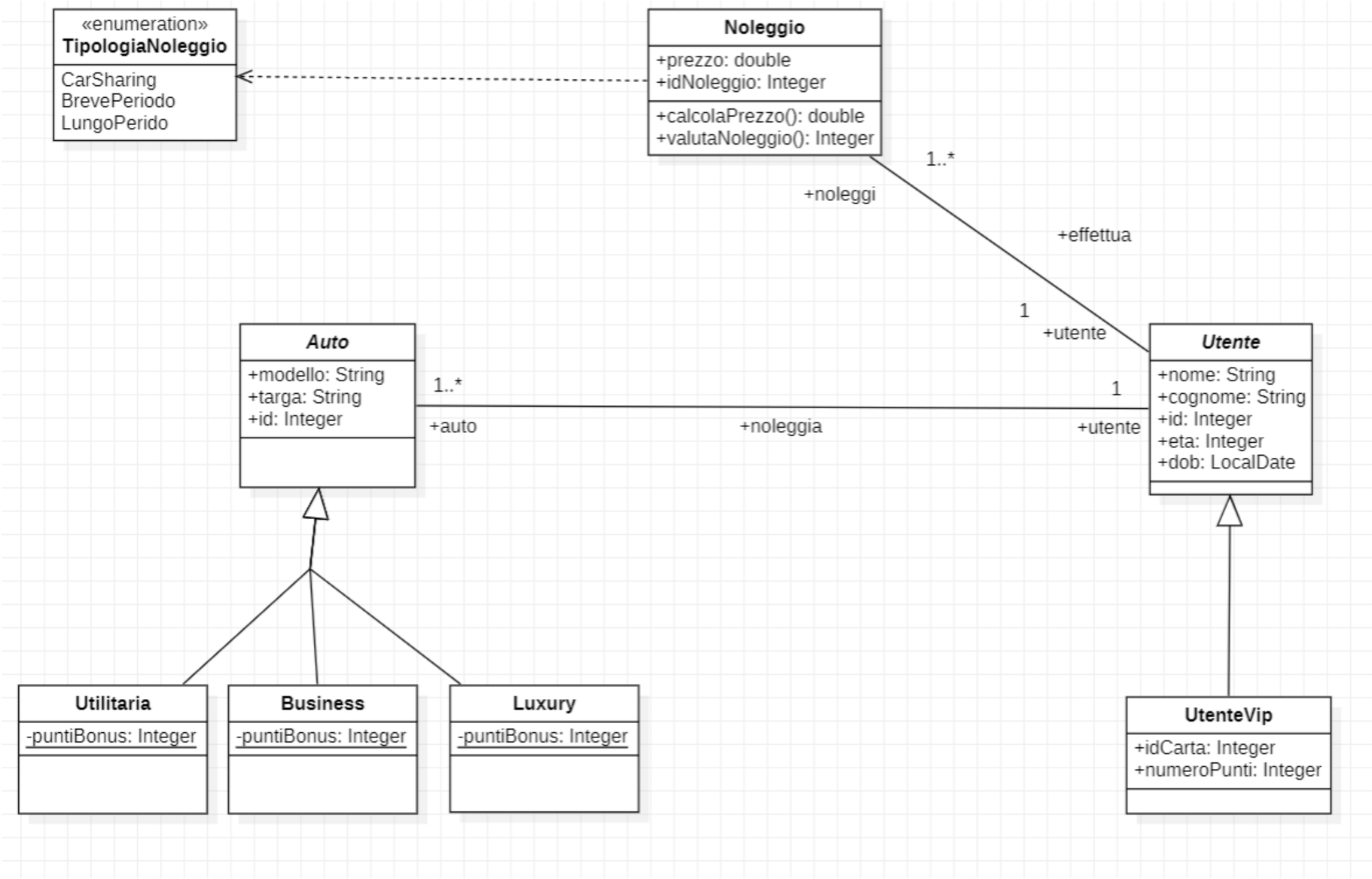
I DIAGRAMMI UML

Use Case Diagram



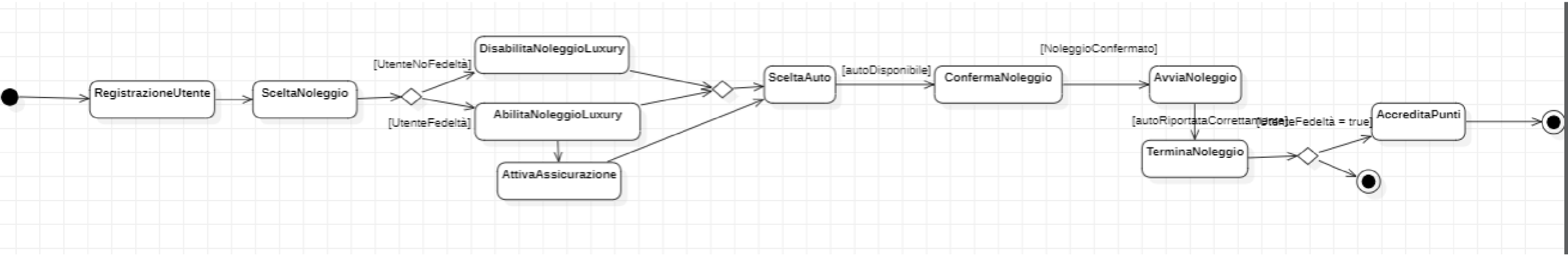
I DIAGRAMMI UML

Class Diagram



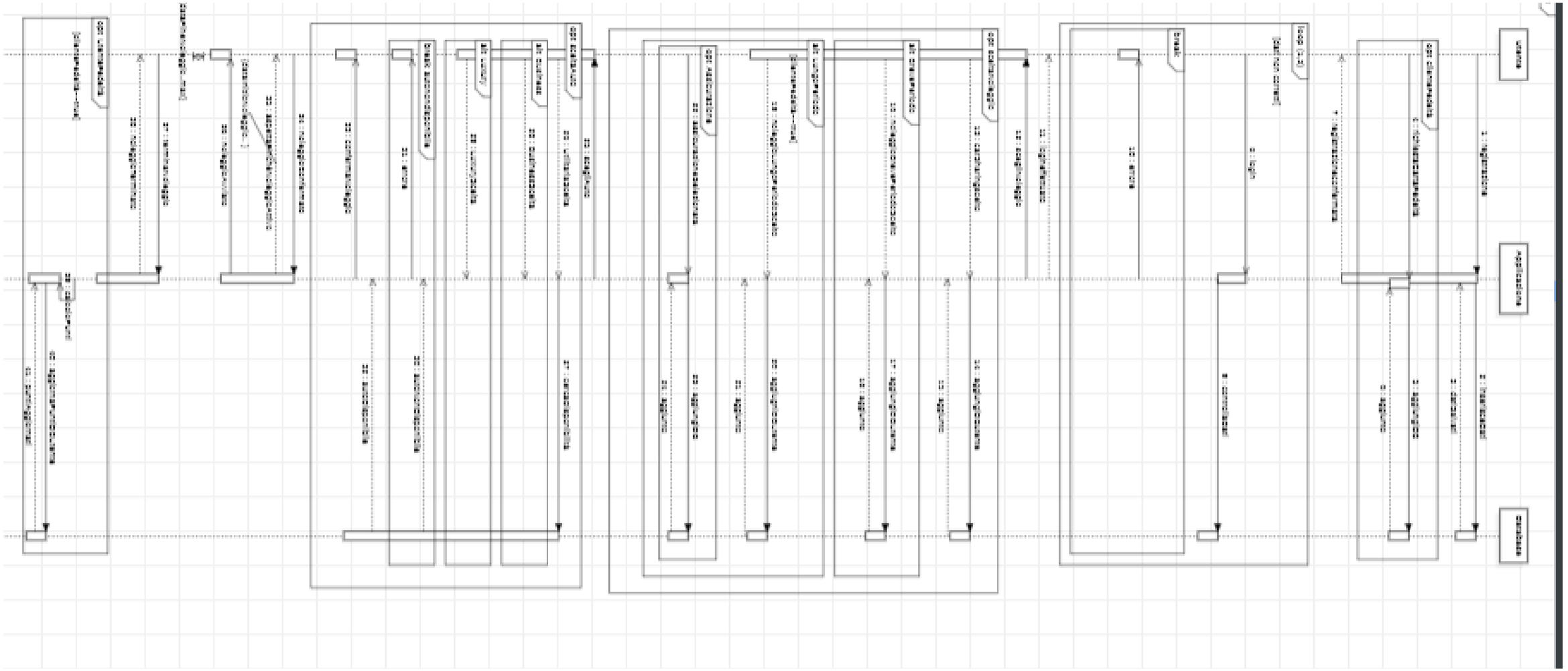
I DIAGRAMMI UML

Activity Diagram



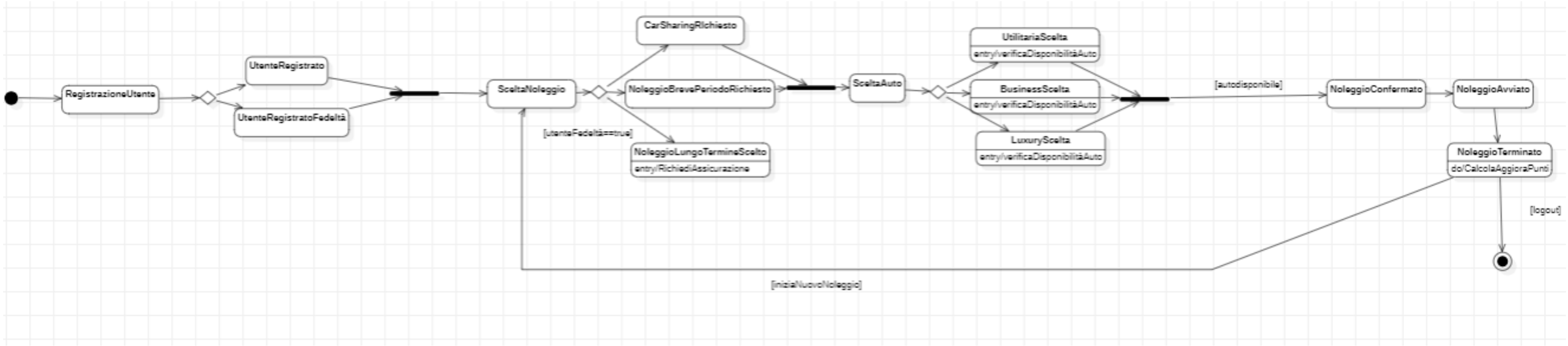
I DIAGRAMMI UML

Sequence Diagram



I DIAGRAMMI UML

State Machine Diagram



LA FASE DELL'IMPLEMENTAZIONE

Siamo partiti dallo **scheletro del codice** generato dal plugin **Rebel** dal Class Diagram.
Da qui abbiamo poi implementato i diversi metodi di ciascuna classe.
Abbiamo utilizzato **Maven** per la gestione del progetto e la compilazione del codice. Per inizializzare il progetto abbiamo usato **SpringBoot**.

Concluso il **backend** abbiamo pensato di realizzare il **frontend** sfruttando **Vaadin** o **Angular** che anche attraverso le sue componenti ci ha permesso di introdurre un'interfaccia di login ed in generale la parte **View** della nostra app

LA FASE DI TESTING

L' Extreme Programming viene riconosciuto come un **Test Driven Developement** ed appunto per questo il nostro è stato un approccio orientato al testing.

Ogni modifica che è stata effettuata veniva testata **manualmente**, e quindi senza la costruzione di casi di test specifici.
Tuttavia quando lo sviluppo dell'applicazione è giunto alle fasi finali abbiamo realizzato dei **casi di test con JUnit**.

JUnit