***SOFTWARE LIFE CYCLE***

Si è scelto di procedere seguendo uno dei metodi Agile: Extreme Programming. Questo promuove lo sviluppo iterativo ed incrementale organizzato in brevi cicli di sviluppo, detti interazioni.

Lo sviluppo è accompagnato dalla stesura di un piano di lavoro che viene continuamente aggiornato a intervalli brevi e regolari.

Dal punto di vista dell’implementazione, Extreme Programming promuove la scrittura di soluzioni semplici che possono essere adattate e migliorate in un secondo momento attraverso refactoring o scrittura di componenti aggiuntive. Nell’Extreme Programming lo sviluppo prevede che tutti lavorino su tutto alternandosi durante alcune fasi, favorendo così il rendimento e mantenendo alto il livello di concentrazione.

Cinque principi:

* **Feedback rapido**: ogni volta che vengono aggiornate parti del sistema queste vengono controllate e testate dagli sviluppatori, in quanto si pensa di consegnare sempre una versione funzionante al cliente.
* **Semplicità**: si vuole evitare di crearsi ulteriore complessità, per cui si cerca di mantenere un design semplice e di aggiungere funzionalità solo quando necessario ed il codice sarà 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 qualità:** si cerca di offrire sempre un prodotto qualitativo

***CONFIGURATION MANAGEMENT***

Per la Configuration Management abbiamo scelto di utilizzare Github. Questo permette attraverso una serie di funzioni di gestire al meglio il progetto.

Per la realizzazione del progetto abbiamo cercato di sfruttare la possibilità di lavorare su branch locali in modo da caricare solo le versioni definitive e funzionanti sul branch main.

Quando sono state effettuate delle modifiche di cui era necessario il giudizio degli altri membri del team abbiamo utilizzato le pull request (come nei casi di alcuni dubbi sui diagrammi per esempio) oppure sono state create delle issues per simili motivi.

***PEOPLE MANAGEMENT AND TEAM ORGANIZATION***

Il team è composto da tre persone:

* **Alexander Rubino** *(MAT. 1064467)*
* **Ibrahima Sarr** *(MAT. 1046446)*
* **Beniamino Infante** *(MAT.1063452)*

La struttura del team è semplice: tutti e tre i membri sono al centro del progetto e la coordinazione avviene mediante diretta supervisione.

Il lavoro viene suddiviso in maniera proporzionale, assegnando uno o due diagrammi UML per ogni singolo membro. Per la stesura del codice invece si è scelto di lavorare in coppia, alternandoci di volta in volta per favorire il rendimento e quindi il risultato finale.

La comunicazione è alla base del progetto, infatti sono fissate riunioni periodiche per riepilogare il lavoro svolto, discutere dei problemi incontrati e fissare nuovi obiettivi da raggiungere nel breve periodo.

***SOFTWARE QUALITY***

Il sistema deve essere conforme allo standard IEEE 9126.

Questo prevede i seguenti attributi:

* **Funzionalità:** il sistema deve essere sicuro e deve contenere le funzioni base descritte nei requisiti non funzionali.
* **Affidabilità:** il sistema deve avere una buona tolleranza agli errori e quindi non causare problemi gravi all’utente finale.
* **Usabilità:** il sistema deve essere facilmente utilizzabile dall’utente finale.
* Efficienza: il sistema non deve usare risorse eccessive. Deve essere ottimizzato per l’uso immediato.
* **Manutenzione:** il sistema deve essere regolarmente aggiornato risolvendo failure o bug incontrati e aggiungendo nuove funzioni richieste dall’utente.
* **Portabilità:** il sistema deve poter girare su diversi OS.

***REQUIREMENTS ENGINEERING***

Si vuole realizzare un applicativo che gestisca i noleggi di auto. Gli utenti saranno in grado di accedere ad un’interfaccia dove è possibile selezionare il tipo di noleggio ed il tipo di auto desiderato. Viene offerta la possibilità di scegliere tra 3 tipi di noleggi: car sharing (durata variabile da qualche giorno ad una settimana), breve peiodo (fino a qualche mese), lungo termine (fino ad un anno). Gli utenti poi potranno scegliere tra diversi tipi di auto in base alle proprie esigenze. Verranno proposte auto di tipo: utilitarie, business, luxury.

Al fine di confermare ed inizializzare il noleggio verrà richiesta l’autenticazione presso il servizio e quindi verrà offerto un servizio di registrazione e login.

Al termine del noleggio agli utenti verranno accreditati punti in base ai km percorsi ed al tipo di noleggio effettuato.

***TABELLA MOSCOW***

* ***MUST HAVE:*** un sistema che permetta la creazione di un noleggio e che permetta di scegliere un’auto
* ***SHOULD HAVE:*** un sistema di registrazione ed autenticazione
* ***COULD HAVE:*** una lista che mostra lo storico dei noleggi per singolo utente
* ***WON’T HAVE:***

***MODELING***

***Tutti i diagrammi UML sono stati generati attraverso il tool StarUML e sono disponibili nel repository all’interno della cartella “Diagrammi UML”.***

***SOFTWARE ARCHITECTURE***

Per l’architettura del nostro software si è scelto di adottare 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 tre 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 dall’interfaccia grafica (View) e li attua modificando lo stato delle altre due componenti

***SOFTWARE 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 insieme all’utilizzo di alcuni casi di test creati con JUnit.