PROJECT PLAN  
(Ingegneria del software 2021/22, Parcheggio a pagamento)

A cura di Raffaele Giacomo Giovanni Di Maio e Giorgio Chirico

1. INTRODUCTION

La decisione riguardante la realizzazione del progetto “Parcheggio a pagamento” è stata presa di comune accordo a metà ottobre 2021.

Il parcheggio, specie per eventi importanti, diventa una risorsa preziosa che va pertanto controllata e tutelata attraverso dei meccanismi.

L’applicativo realizzato vuole gestire il flusso di entrata e di uscita dei veicoli dal parcheggio stesso, comprendendo le attività di:

-generazione ed erogazione ticket in entrata;

-riconoscimento e gestione pagamento ticket in uscita;

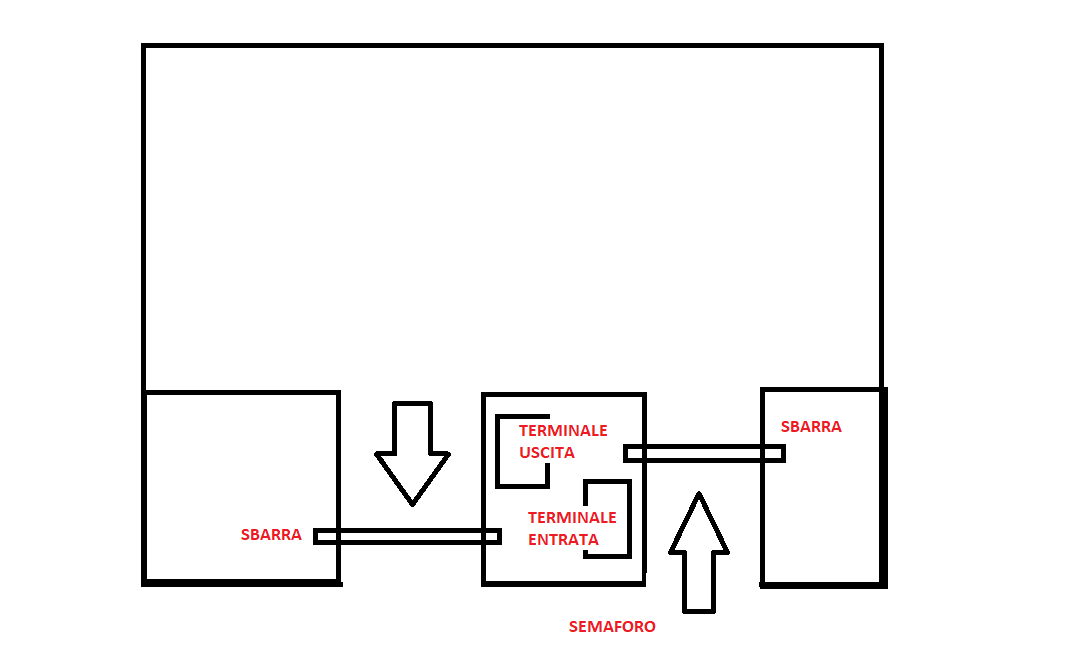
-gestione concorrenzialità e limite d’entrata;

-reliability dei dati per il tempo utile.

Il progetto si pone come obiettivo la realizzazione di un parcheggio dotato di sbarra per la gestione del flusso entrante e uscente delle macchine, garantendo l’indipendenza dei traffici nell’applicativo, e dotato di semaforo che notificasse all’esterno se ci fossero parcheggi liberi o meno.

Inoltre, l’uscita dal parcheggio ha, come condizione, il pagamento del ticket anch’esso effettuato in maniera asincrona.

Il servizio offerto viene erogato per mezzo di due terminali dotati di GUI, posti rispettivamente sul canale d’entrata e d’uscita del parcheggio, mentre la gestione dei ticket e dei pagamenti viene assegnata a un server locale centrale: va pertanto garantita semplicità d’uso per l’utente, una stratificazione dell’applicativo e una gestione della concorrenzialità dei flussi tale che l’attività del server appaia trasparente agli utenti e, il più possibile, ai terminali.



I responsabili del progetto sono due studenti del Polo Ingegneristico dell’Università di Bergamo:

* Giorgio Chirico;
* Raffaele Giacomo Giovanni Di Maio.

Nel corso del progetto sono stati ricoperti più ruoli all’interno del team da entrambi i membri, perciò risulta difficile farne un’assegnazione specifica.

Basandosi tuttavia sulle attività maggiormente ricoperte nel corso del progetto, si può fare la seguente distinzione:

-Giorgio Chirico: Software Architect e DB Specialist;

-Raffaele Giacomo Giovanni Di Maio, Project Manager e Programmatore.

[aggiunta del 2/12]

1. PROCESS MODEL

XP: Xtreme Programming.

È stata adottata una filosofia di tipo agile caratterizzata da uno sviluppo del lavoro “a piccoli passi” con:

* implementazione veloce;
* prototipizzazione sempre accompagnata da testing;
* focus sugli stakeholder con aggiunta di features nel tempo in base alle loro esigenze risultate dalle interviste fatte.

I punti fondamentali del nostro processo di lavoro sono:

* Coesione nel team e costante comunicazione sugli sviluppi;
* Piani dinamici secondo un Planning Game: si intende che, durante lo sviluppo, ogni piano verrà creato in un breve periodo di tempo e dotato, eventualmente, di una futura estensione poi realizzata;
* Coinvolgimento del cliente nel testing;
* Orientamento al sustainable pace;
* Sviluppo test-driven continuo;
* Costante refactoring.

Per quanto riguarda le milestones, è stato realizzato ad inizio progetto il file “IdeeProgetto.txt” in cui, all’interno, sono state definite tutte le deadline riguardanti lo sviluppo.  
Qui sotto viene riportata la parte del file relativa alle delivery:

“ […]

SVOLGIMENTO DEL LAVORO

- FASE 1 [DA TERMINARE ENTRO FINE OTTOBRE] --> IMPLEMENTARE CONGIUNTAMENTE ED IN MODO FUNZIONANTE IL PROGRAMMA UTILIZZANDO IL MECCANISMO DEI SEMAFORI CON PROGRAMMAZIONE CONCORRENZIALE (MUTEX PER GESTIRE IL PARCHEGGIO, ENTRATA PER GESTIRE LE MACCHINE ENTRANTI, USCITA PER GESTIRE LE MACCHINE USCENTI --> MECCANISMO DI TURNAZIONE ENTRATE ED USCITE PRIVILEGIANDO LE USCITE).

EVENTUALE IMPLEMENTAZIONE/SCELTA DEL DATABASE DA UTILIZZARE PER IMMAGAZZINARE LE INFORMAZIONI RELATIVE ALLE MACCHINE COME IL TICKET E LA TARGA [ rimosso il 13/11 ].

GITHUB.

- FASE 2 [DA TERMINARE ENTRO FINE NOVEMBRE] --> IMPLEMENTAZIONE TOTALE DEL DATABASE CON L'APPLICATIVO ED IMPLEMENTAZIONE DEL PAGAMENTO DEL TICKET IN MODO FUNZIONANTE.

- FASE 3 [DA TERMINARE ENTRO META' DICEMBRE] --> AVER REDATTO TUTTA LA DOCUMENTAZIONE NECESSARIA DA FORNIRE.

[…] ”

I punti critici del progetto sono:

* La gestione della concorrenzialità;
* Definire la struttura del server, dei terminali e le modalità di comunicazione tra questi;
* La gestione del database;
* La ricerca delle librerie per gli obiettivi sopra citati.

Si vorrebbe inoltre implementare la possibilità di avere anche una diversificazione dei pacchetti software che si possono offrire:

* Parcheggio con sbarra d’entrata e d’uscita per il controllo: necessita di hardware dedicato quali sbarre e semaforo, gestione multi-thread

(gestione sincrona terminale d’entrata e d’uscita)

* Parcheggio barrierless (senza sbarra): variante versatile, costi contenuti, server procedurale (singolo terminale) o multi-thread (gestione sincrona terminale d’entrata e d’uscita). Maggiore sforzo computazionale: bisogna memorizzare le targhe

//aggiunta del 13/11

Data una maggiore priorità nel parcheggio con sbarra, il team ha deciso di inserire il parcheggio di tipo “barrierless” tra gli sviluppi futuri

1. ORGANIZATION OF THE PROJECT

Il progetto è sviluppato interamente dal team in maniera verticale senza outsourcing del lavoro o affidamento di task ad organizzazioni esterne.

Gli stakeholders verranno coinvolti nello sviluppo del progetto, presentando le loro idee in consultazioni periodiche.

Il team sarà composto da due sviluppatori che ricopriranno simultaneamente diversi ruoli, talvolta analoghi, utilizzando anche il paradigma del pair-programming.

1. STANDARDS, GUIDELINES, PROCEDURES

Per quanto riguarda la programmazione, si è scelto come standard il linguaggio di programmazione Java su ambiente Eclipse.

Una regola importante che è stata imposta è quella di rimanere sempre sulle librerie proposte da java in modo che ci sia omogeneità di sviluppo.

La coordinazione nel team sarà regolata da periodici incontri di due ore, tre volte a settimana, durante i quali sarà discusso e redatto il codice per l’implementazione del progetto; si discuterà inoltre di problemi e prossime versioni.

//aggiunta del 09/11

A livello database è stato utilizzato MySQL con la libreria Connectors per Java.

La comunicazione su MySQL è di tipo socket TCP con autenticazione (porta 3300).

L’utilizzo di MySQL è stato dettato dalla semplicità di implementazione ed è stato astratto ulteriormente dalla classe BackupDB in java.

1. MANAGEMENT ACTIVITIES

Si svolgeranno due riunioni a settimana, che risulteranno con la produzione di due file denominati “meeting log” in cui si appuntano gli obiettivi conseguiti e le future intenzioni, sviluppi, argomenti dei prossimi meeting.

Si prevede di utilizzare GitHub per l’organizzazione di una baseline dove verranno conservati i seguenti items:

-codici: produzioni di file sorgente completi, funzionanti, utili (quindi testati!)

-documentazione

-materiali di setup per i framework utilizzati (ad es. il database) [ aggiunta del 09/11 ]

//aggiunta del 19/10

Implementato GitHub, le attività di log del lavoro conseguito saranno documentate per mezzo dei commit sulla baseline e, per le più importanti, anche per mezzo di file di “meeting log”. Il resto sarà affidato alla comunicazione diretta tra i partecipanti al progetto.

//aggiunta del 13/11

La cartella sulla baseline relativa ai codici sarà composta da:

-una cartella contenente prototipi;

-una con le versioni complete del progetto;

-l’ultima versione del progetto.

Ogni prototipo o versione precedente di progetto dovrà essere:

-testato

-caratterizzato da un nome che

* + sia affine al tipo di prototipo che si sta sviluppando (se il prototipo sviluppato e testato è la raffinazione ulteriore di un prototipo già esistente, si riprende lo stesso nome o ci si ispira il più possibile a questo)
  + espliciti la versione rispetto al progetto globale
  + se focalizzato su un componente, espliciti la versione di questo

1. RISKS

I rischi temporali riguardanti il nostro progetto sono:

* Non trovare un database di facile implementazione;
* Non rispettare le deadline;
* Difficoltà varie nella concezione dell’architettura dei terminali e del server;
* Setup porte database.

I rischi progettuali sono:

* Fragilità del database;
* Comunicazione IPC non stabile;
* Possibile Starvation/Deadlock derivanti dalla concorrenzialità;
* Installazione e setup della JDK.

1. STAFFING

Il personale è rimasto sempre lo stesso nel corso del tempo alternando i ruoli all’interno della costruzione dell’architettura e dell’intero software applicativo.

1. METHODS AND TECHINQUES

* **Requirement Engineering**

1. *Elicitation* sulla linea delle seguenti tecniche:

-*metodo Tayloriano* (soprattutto);

-*Scenario-based Analysis* (importante per la determinazione di scenari alternativi);

-*Prototyping* quando abbiamo avuto la possibilità di offrire interfaccia GUI. [aggiunta del 13/11] .

1. *Requirements Specification* eseguito seguendo lo standard IEEE 830;
2. *Verification&Validation* per mezzo di prototipo e test plan.

* ***Requirement Analysys***

in scala MoSCoW su documento MoSCoW.txt.

* ***Design e architettura***

In sintesi, il design segue il modello client-server, con server modularizzato, multi-thread, e due client, anch’essi multi-thread, con struttura a layers; sotto al server, una banca dati. Le comunicazioni terminale-server e server-base dati sono di tipo TCP Socket.

**L’architettura** generale vuol essere in repository-style: un componente centrale (chiamato GestoreParcheggio) gestisce la concorrenzialità dei client (ovvero i terminali denominati TerminaleEntrante e TerminaleUscente) che ne usufruiscono per l’accesso dati e aggiornamento dati.

Per dar trasparenza delle funzionalità offerte dalle GUI dei terminali agli utenti, i terminali saranno realizzati in layered-style, seguendo le linee del paradigma model-control-view per la suddivisione tra GUI, gestore degli eventi della GUI (controller), richiesta e estrazione dati dal gestore centrale e invio pagamenti in maniera trasparente.

Per quanto riguarda i **connettori**,il programma si basa sostanzialmente sul tipo message passing per la comunicazione IPC tra terminali-server e server-db.

Altri connettori sono: data flow per il passaggio dati relativi a ticket e pagamento in maniera ordinata e controllata, implicit invokation per l’attivazione di funzioni a seguito di pulsanti premuti sulla GUI. [ aggiunta del 20/11 ]

Particolare attenzione al data flow è stata messa per “rifiutare” le richieste di erogazione ticket qualora il parcheggio fosse pieno, oltre che per la gestione del comportamento del semaforo, con meccanismo di “permesso” (anche questo gestito in message passing). [ aggiunta del 2/12 ]

1. QUALITY ASSURANCE

Per quanto riguarda la quality assurance, in una prima fase essa è stata realizzata attraverso continui test realizzati da noi così da verificare che tutte le richieste fossero state rispettate.

In una seconda fase invece, essa è stata garantita attraverso costanti contatti con gli stakeholders del progetto nonché test vari fatti con loro in maniera tale da capire se tutte le esigenze chieste erano state soddisfatte nella maniera corretta.

(Infine, nel caso in cui ci fossero problemi non notati o non risolti, essi devono essere redatti direttamente a noi che procederemo con la risoluzione dei bug e/o con l’implementazione delle funzioni aggiuntive richieste).

1. WORK PACKAGES

I pacchetti di lavoro sono stati suddivisi tra noi due programmatori in maniera tale che Giorgio Chirico si occupasse dall’architettura e della realizzazione della comunicazione Client-Server tra i terminali con conseguente implementazione di classi e metodi java vari mentre Raffaele Di Maio si occupasse della parte legata alle interfacce ed a quella di setup dello schema iniziale e del lavoro da cui partire con conseguente implementazione di metodi e classi java di vario tipo.

1. RESOURCES

Le risorse utilizzate per la realizzazione di questo progetto sono state le seguenti:

* + Ambiente di sviluppo Eclipse (versione del 2019 e versione 2021);
  + Libreria jre contenuta in javaSE – 1.8;
  + Vari package per realizzare le classi, in particolare:
    - AWT, SWING per la realizzazione delle GUI;
    - UTIL per la gestione concorrenzialità e time-out (per la GUI);
    - IO per la gestione dei flussi socket; in particolare, IO.Serializable per la serializzazione dell’oggetto Ticket;
    - TIME per l’utilizzo di oggetti timestamp;
    - NET per la gestione della comunicazione IP;
    - Vari package da “Prog-4ed” per la gestione degli I/O su terminale;
  + Un database MySQL locale (da configurare a ogni nuova installazione) con connessione TCP.

1. BUDGET AND SCHEDULE

Entrambi i programmatori useranno i propri Personal Computer per l’implementazione dell’applicativo andando ad attingere da Internet le varie ed eventuali librerie Java (e non), utili per il raggiungimento dello scopo prefissato.  
Di conseguenza, non sono previste spese dal punto di vista economico mentre, per quando riguarda la schedule del lavoro, ci si può rifare al Punto 2 “Process Model”, in particolare ai file delivery pubblicati.

1. CHANGES

I cambiamenti sono stati realizzati per piccoli passi come redatto dal manifesto agile.  
Ad ogni meeting sono state realizzate nuove classi e/o metodi che portassero il software al risultato finale tenendo controllato Il numero di quest’ultimi così da non trovarsi mai in una situazione in cui essi fossero radicali come previsto dallo stile agile da noi utilizzato.

1. DELIVERY

Il progetto verrà consegnato agli stakeholders con tutto quello che serve per il suo

funzionamento tra cui troviamo:

* L’applicativo contenente tutte le classi e i metodi;
* Le istruzioni per installare l’estensione relativa a MySQL per Java Eclipse per l’utilizzo del DB.