UNIVERSITÀ DI PISA

Un giorno al Museo

Progetto di Ingegneria del Software 2020-2021

Luca Rizzo (598992),
Elia Menoni (598375),
Davide Marchi (602476),
Giordano Scerra (596363),
Stefano Passanante (597415),
Giuseppe Di Stefano (599065).

31 mag 2021

Indice

1 | Abstract

Il progetto “Un giorno al museo” consiste nella realizzazione di un sistema che possa aiutare la regione toscana a gestire e coordinare le visite nei vari musei presenti sul territorio.

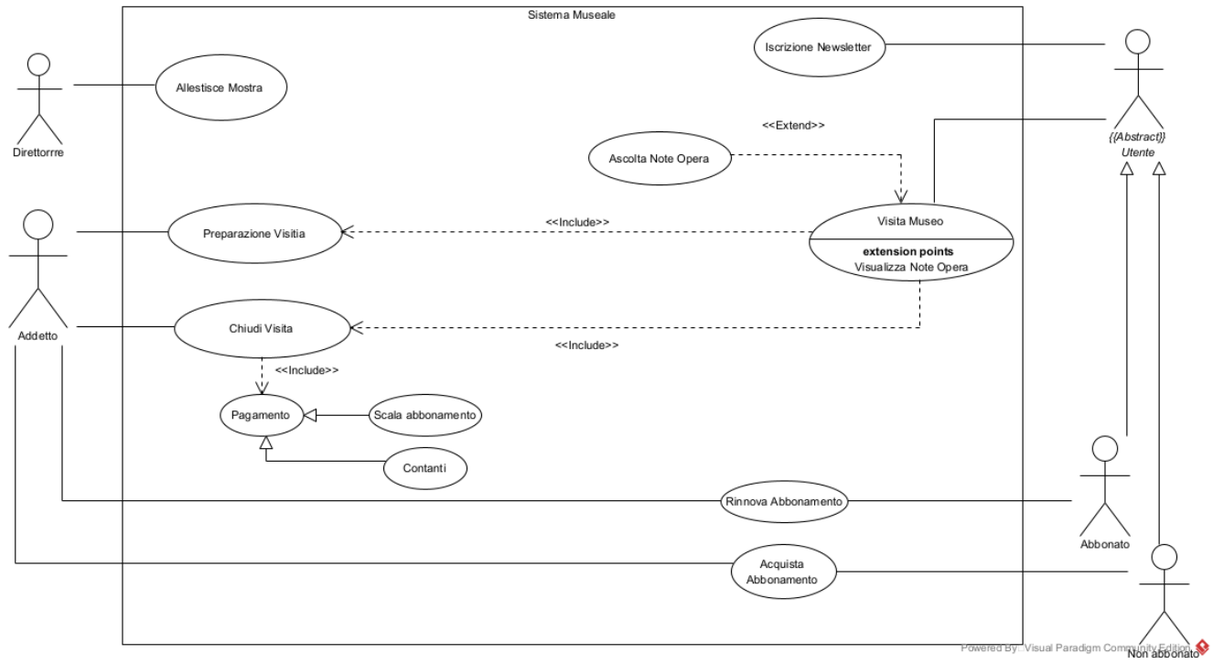
Tra le richieste vi era anche quella di gestire un servizio di newsletter: abbiamo dunque ritenuto necessaria l’ideazione di siti web che consentano agli utenti di farlo in autonomia.

Abbiamo inoltre fissato delle assunzioni per la realizzazione del progetto.

Il sistema offre benefici sia al singolo museo, che gestirà più rapidamente pagamenti, iscrizioni alla newsletter e gestioni delle sale, sia a tutto il sistema museale con una migliore coordinazione per il trattamento degli abbonamenti e delle opere.

2 | Casi d'uso

Si fornisce di seguito lo schema dei casi d'uso del sistema SW.



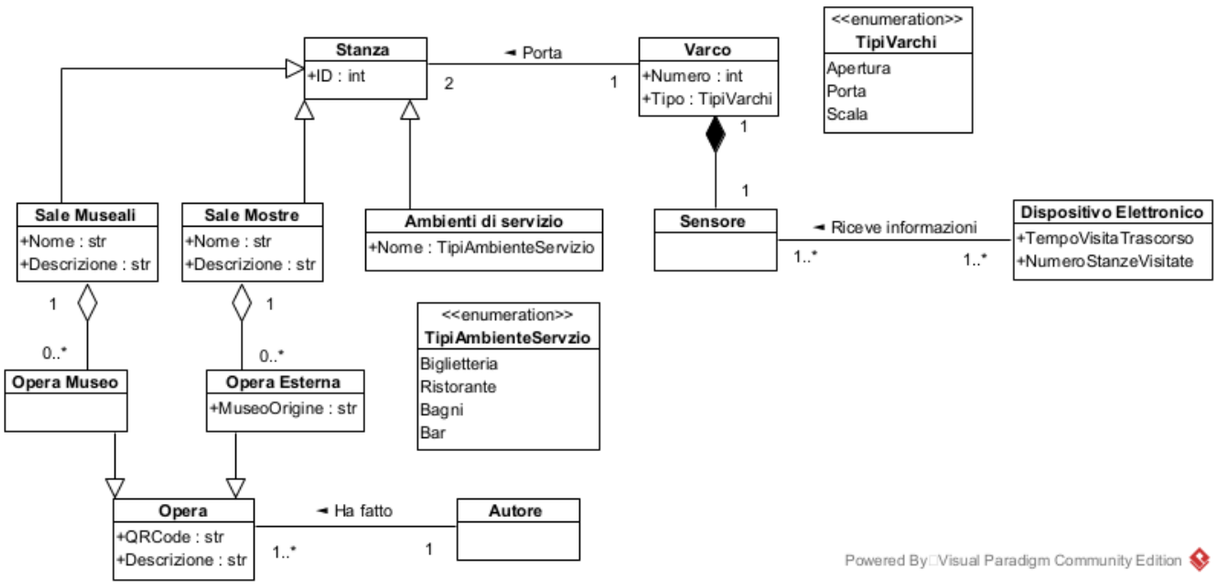
2.1 Narrativa

Di seguito è riportata la narrativa del caso d'uso **Preparazione Visita**

PREPARAZIONE VISITA	
Breve descrizione	L'addetto configura il dispositivo in base alle esigenze dell'utente.
Attori primari	Addetto
Attori secondari	Utente
Pre-Condizioni	Ci sono dispositivi disponibili `AND` un utente è in biglietteria e vuole visitare il museo
Sequenza eventi principale	1. L'addetto richiede i documenti all'utente, 2. l'addetto controlla dal PC se l'utente è abbonato, 3. `SE(utente non abbonato)` (a) l'utente sceglie una tariffa tra quelle disponibili, 4. il cliente sceglie la lingua e il tipo di contenuto (per adulti, per bambini) 5. l'addetto configura il dispositivo interfacciandosi con il PC e lo consegna all'utente.
Post-Condizioni	Il dispositivo è stato configurato e viene consegnato all'utente.
Sequenza alternativa degli eventi	Nessuna.

3 | Diagramma delle Classi

Come richiesto sul testo, di seguito, viene riportato il diagramma delle Classi atto alla descrizione della struttura museale e le assunzioni che hanno portato al relativo schema.

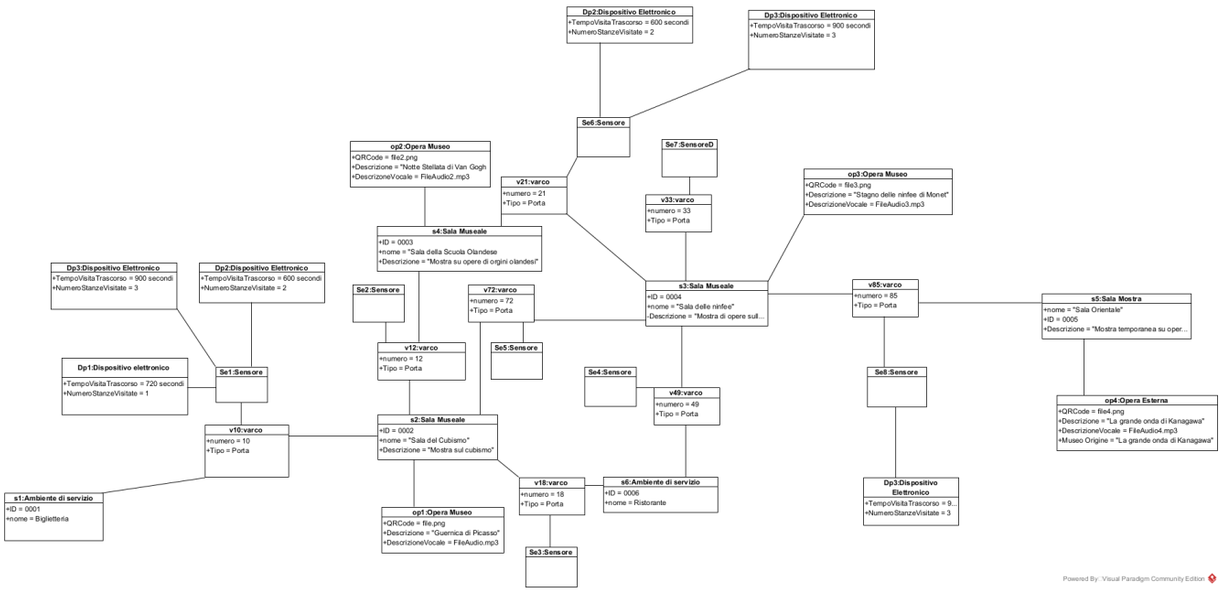


3.1 Assunzioni

- Ogni **Stanza** assume un tipo in relazione a cosa ospiterà,
- viene deciso una sola volta se una **Stanza** ospiterà opere interne o esterne al museo,
- ogni **Varco** è composto da un **Sensore**,
- una **Sala Museale** ospita opere in possesso al museo (**Opera Museo**),
- una **Sala Mostre** ospita opere provenienti da altri musei (**Opera Esterna**),
- un'**Opera**, a prescindere che sia interna o esterna al museo, possiede un **Autore**;
- ogni **Dispositivo Elettronico** riceve informazioni da 1 o più **Sensori** nel corso della visita,
- gli **Ambienti di Servizio** possono assumere una funzione tra quelle prestabilite,
- ogni **Varco** è categorizzato in base a un tipo derivante dalla struttura del museo.

4 | Diagramma degli Oggetti

Di seguito è riportato il diagramma degli oggetti che descrive una possibile istanza del diagramma delle classi. I dati inseriti sono stati estrapolati dalla mappa presente sul testo e alcune informazioni aggiuntive.



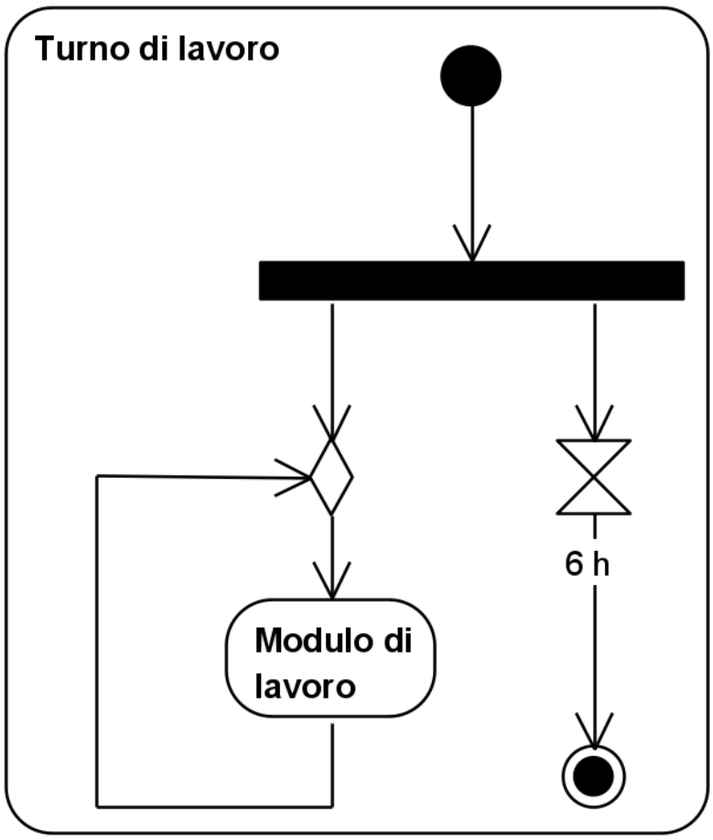
4.1 Assunzioni

- All'interno del museo, al momento, sono presenti 3 visitatori (e relativi dispositivi elettronici associati),
- il dispositivo 1 ha visitato una sola stanza (passando dal sensore 1) e sono trascorsi 720 secondi dall'inizio della visita,
- il dispositivo 2 ha visitato 2 stanze (passando dai sensori 1 e 6) e sono trascorsi 600 secondi dall'inizio della visita,
- il dispositivo 3 ha visitato 2 stanze (passando dai sensori 1, 6 e 8) e sono trascorsi 900 secondi dall'inizio della visita.

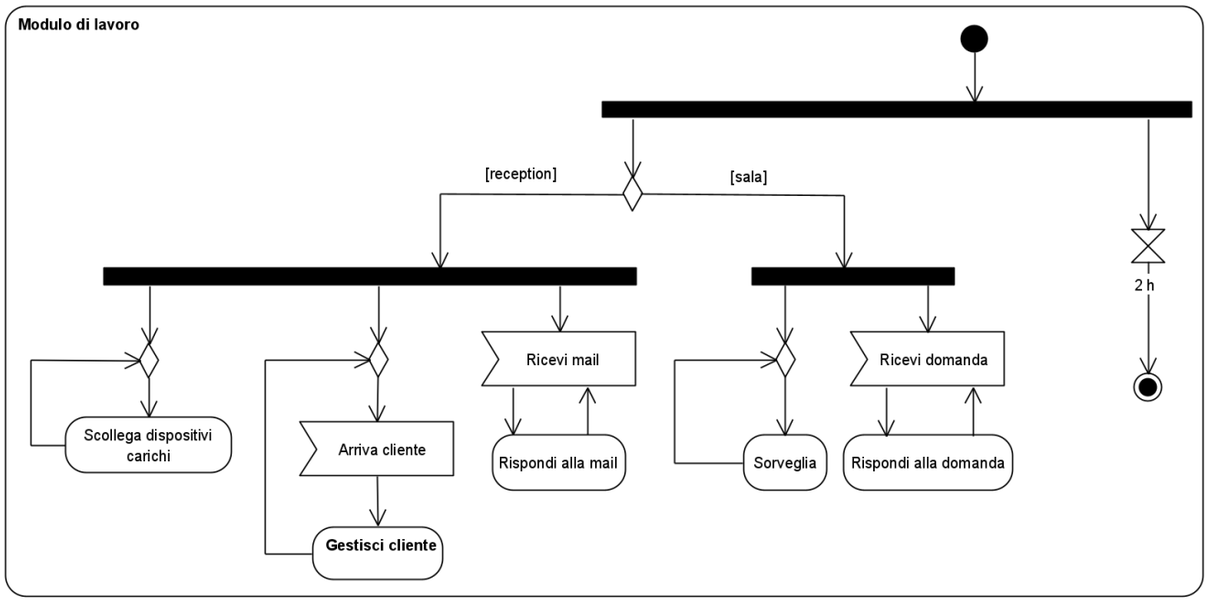
5 | Diagramma di Attività

Si fornisce il diagramma di attività che modella l'attività di un addetto al museo.

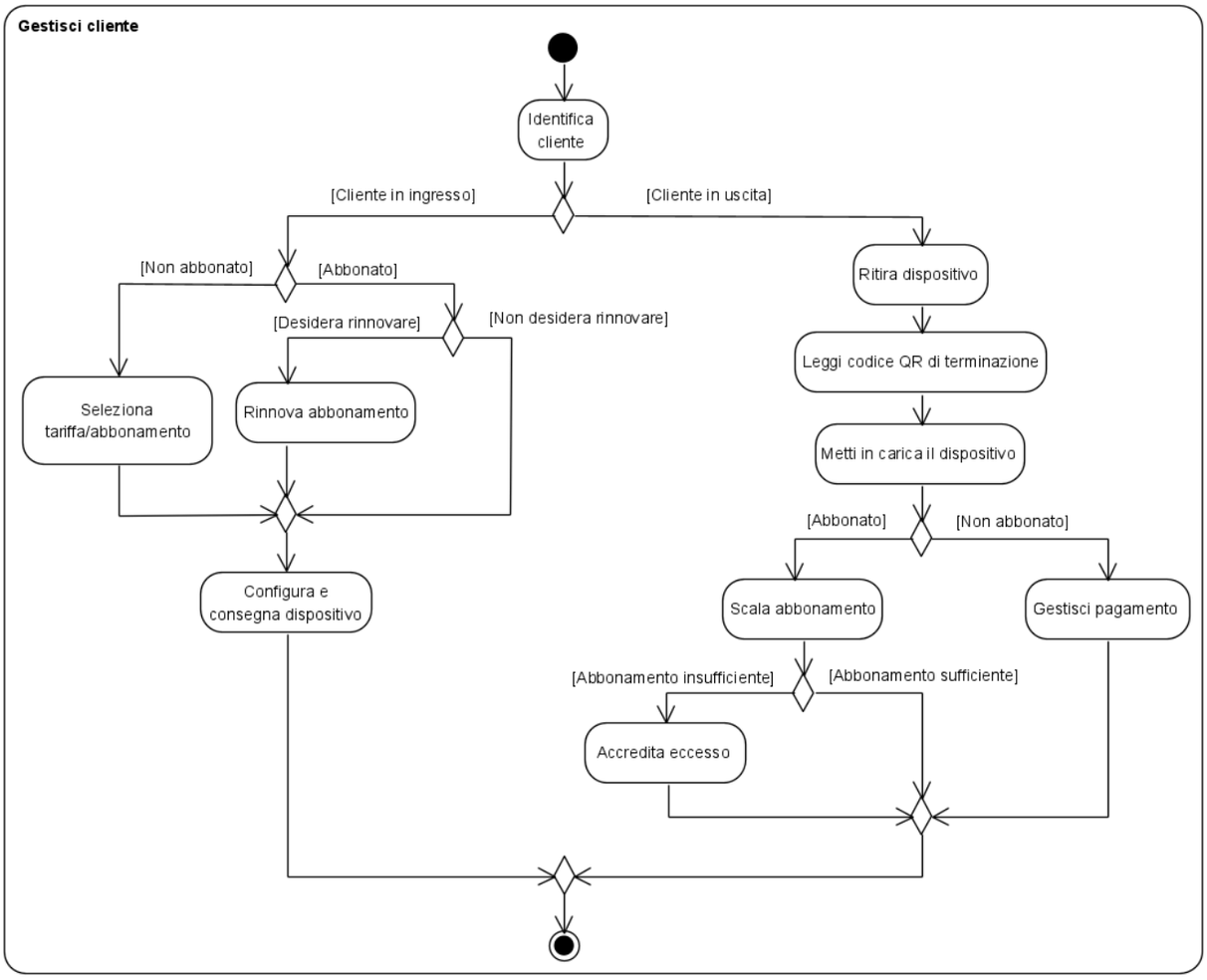
5.0.1 Schema Turno di Lavoro



5.0.2 Schema Modulo di Lavoro



5.0.3 Schema Gestione Clienti



5.1 Punti cardine

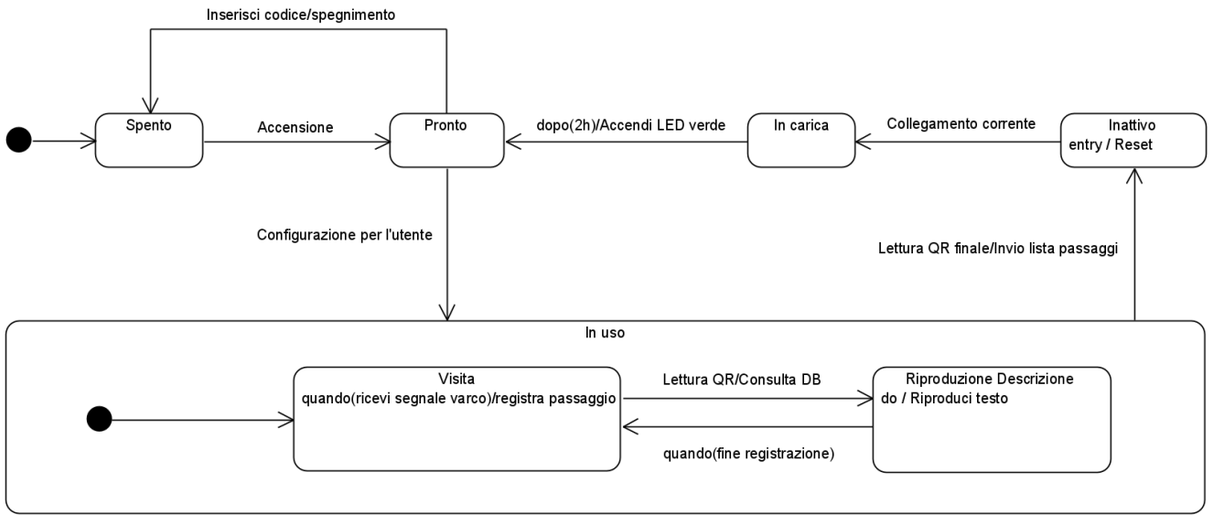
- Un addetto ha un **turno** di lavoro di 6 ore consecutive, diviso in **moduli** di 2 ore,
- all'inizio di ogni modulo viene assegnato alla reception oppure ad una sala,
- alla **reception** deve:
  - mettere in carica i dispositivi scarichi utilizzati (che si ricaricano in 2 ore),
  - servire i clienti all'entrata e all'uscita del museo,
  - controllare le email per eventuali richieste di info;
- nella **sala** deve:
  - fare sorveglianza,
  - rispondere ad eventuali domande.

5.2 Assunzioni

- L'addetto riconosce quali sono i dispositivi carichi da scollegare dalla corrente tramite l'accensione di un **LED verde** sul dispositivo. In questo modo i dispositivi potranno essere collegati in un turno e scollegati in quello successivo.
- Ogni dispositivo viene **sempre** messo in carica alla fine di una visita, per assicurarsi che non si scarichi durante la successiva.

6 | Diagramma di Macchina a Stati

Si presenta un diagramma di macchina a stati che modella gli stati attraversati da un dispositivo elettronico all'interno del museo.

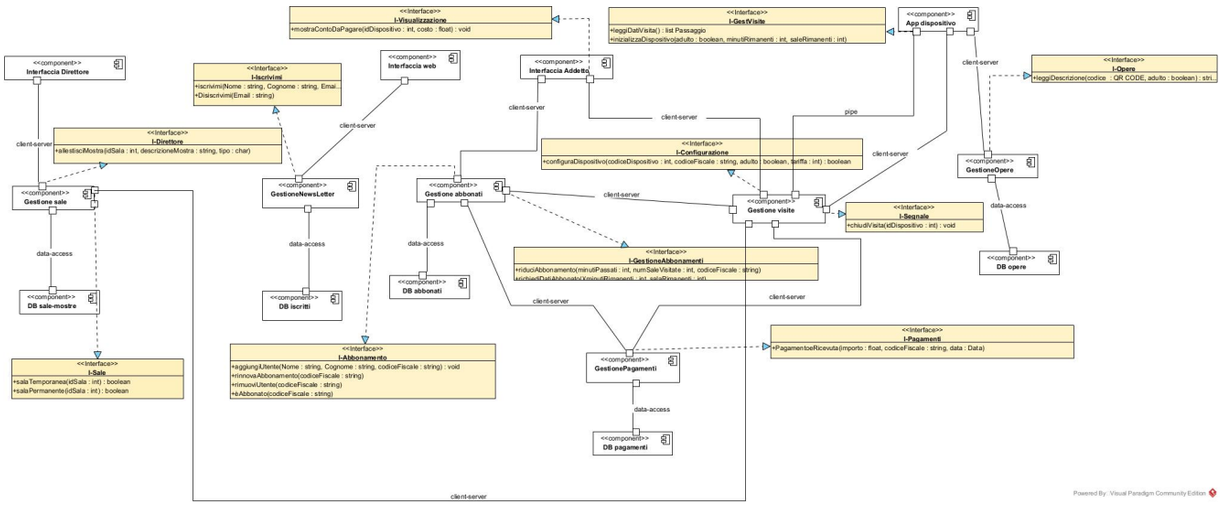


6.1 Assunzioni

- Gli utenti **non** si muovono mentre ascoltano le spiegazioni,
- i dispositivi **non** si possono scaricare durante la visita,
- il dispositivo, alla sua prima accensione mattutina, è **carico** (poiché è stato caricato il giorno precedente);
- il dispositivo può essere spento **solo** dal dipendente in reception, tramite l'inserimento di un codice per evitare che il cliente spenga il dispositivo durante la visita;
- ogni dispositivo ha un suo **caricatore** (idealmente: si ha un dock di carica con n posti per n dispositivi i.e. dispositivi salvatempo del supermercato),
- l'evento "Configurazione per l'utente" comprende la scelta della **lingua** e del **contenuto** (adulti/bambini),
- il **Reset** consiste nell'azzeramento della lista dei passaggi, per essere sicuri che un visitatore non paghi anche le sale visitate da un altro cliente.

7 | Diagramma C&C

Si fornisce di seguito una vista C&C che realizza le funzionalità descritte dal diagramma dei casi d'uso.



7.1 Assunzioni

Per individuare le componenti nella vista C&C abbiamo seguito i principi **GRASP**, ovvero siamo partiti dalle **funzionalità offerte** dal diagramma dei casi d'uso e abbiamo assegnato le responsabilità a ciascuna componente.

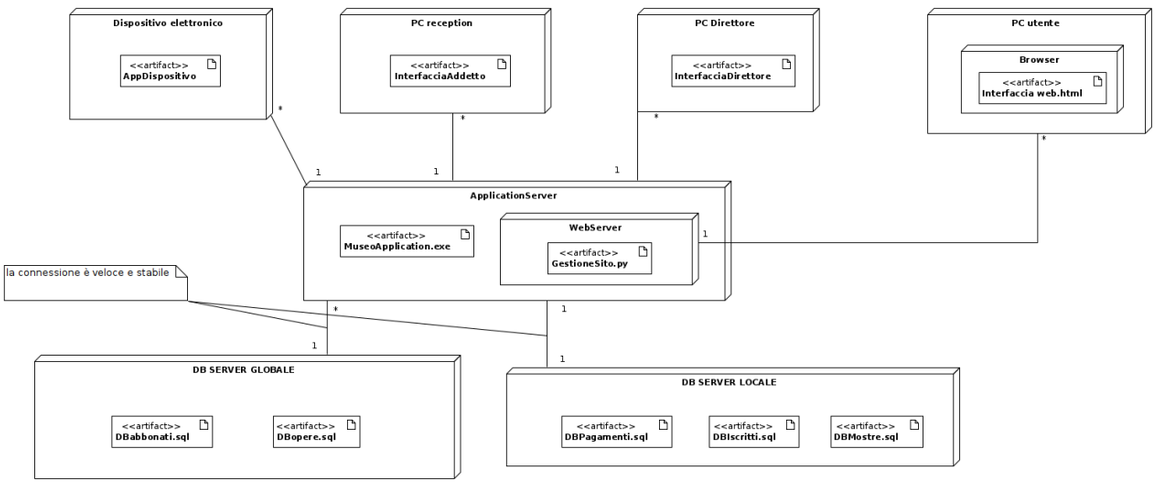
La progettazione è stata quindi guidata dalla realizzazione dei casi d'uso, cercando di mantenere più saldo possibile anche il principio **SOLID** di "Single Responsibility": ciascuna componente di gestione ha una singola responsabilità, ovvero gestisce un solo aspetto del dominio di interesse (ha un solo motivo per cambiare).

La componente **Gestione Visite**, ad esempio, si occupa di gestire le varie fasi relative ad una visita, a partire dall' inizializzazione di un dispositivo fino al pagamento o alla riduzione di un abbonamento, ma richiede ad altre componenti metodi per completare operazioni che non fanno parte della sua responsabilità: per registrare pagamenti e stampare ricevute si interfaccia al gestore dei pagamenti, per avere informazioni sui tipi della sala (temporanea, permanente o di servizio) si interfaccia al gestore delle sale, per ridurre un abbonamento si interfaccia a Gestione abbonamenti etc.

Si possono così individuare, grazie alle **interfacce** che ciascuna componente implementa e richiede, le varie interazione fra le componenti che portano alla realizzazione delle funzionalità descritte nel diagramma dei casi d'uso.

8 | Diagramma di Dislocazione

Si fornisce di seguito una possibile dislocazione del software sui nodi hardware.

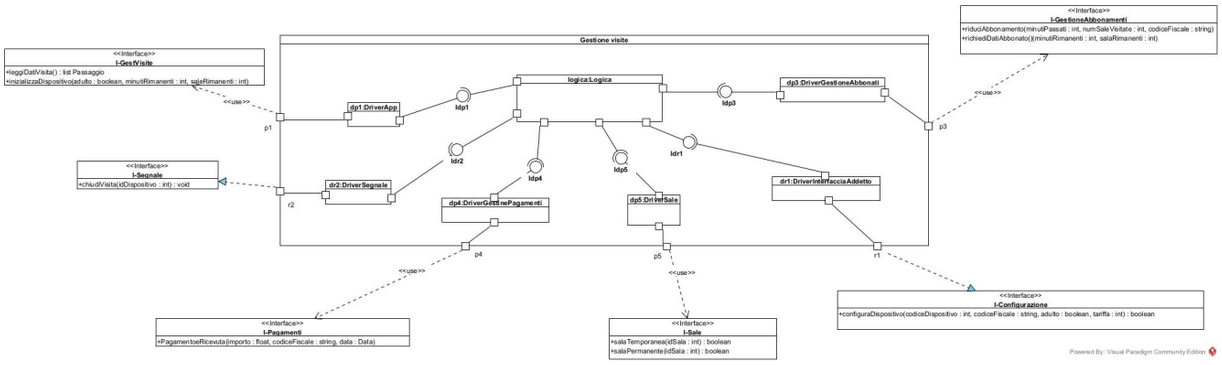


8.1 Assunzioni

- Una visita ad un museo scala l'abbonamento all'intero sistema museale,
- tutte le opere sono associate un QR code,
- ogni museo avrà a disposizione un application server **personale** su cui installare gli artefatti che permetteranno la gestione del software e la gestione del sito web associato,
- ogni museo avrà a disposizione un **database locale** in cui memorizzerà:
  - le informazioni relative ad ogni stanza del museo (tipo, descrizione),
  - i pagamenti relativi ad ogni visita o ogni abbonamento,
  - tutti gli iscritti alla newsletter;
- l'insieme dei musei **condividono** un database contente:
  - tutti gli abbonati: un utente si abbona al sistema museale e non al singolo museo. Ciò garantisce una migliore coordinazione degli abbonati al sistema museale. (una visita al singolo museo scala l'abbonamento relativo all'insieme del sistema museale),
  - tutte le opere del sistema museale: questo permette ai vari musei di scambiarsi le opere presenti nel sistema in maniera agevole, trasferendo opera e codice QR associato. Tutti i singoli application server si collegheranno al database globale per ottenere informazioni riguardo ad un'opera.

9 | Diagramma di Struttura Composita

Si fornisce di seguito un diagramma di struttura composita della componente gestione visite.



9.1 Assunzioni

- E' stato applicato il pattern di strutturazione visto a lezione, introducendo per ogni porto un **driver** che permette la comunicazione con le altre componenti e una **logica** che si occupa della realizzazione delle funzionalità richieste dalla componente,
- i porti di tipo "**p**" sono porti con un'interfaccia "richiesta" da altre componenti,
- i porti di tipo "**r**" sono porti con un'interfaccia "fornita" ad altre componenti.

10 | Testing

10.1 Punto A

10.1.1 Assunzioni

- Consideriamo le sale con indice uguale a 0 sale di **servizio** (stanze in cui la sosta non va pagata),
- consideriamo le sale di indice pari sale **permanenti**,
- consideriamo le sale di indice dispari sale **teporanee**.

Secondi gli stub qui definiti:

```
//effects: ritorna true se si tratta di una sala permanente
salaPermanente(int i){
  if(i==0)
    return false; // una stanza di servizio non va pagata
  if(i%2==0)
    return true;
  return false;
}
```

```
//effects: ritorna true se si tratta di una sala temporaneaSa
salaTemporanea(int i){
  if(i%2==0)
    return false;
  return true;
}
```

Date le assunzioni precedenti, possiamo supporre che: **la prima e l'ultima** sala segnata dal dispositivo elettronico saranno la biglietteria (sala di servizio), ovvero sala in cui ritiro e consegno quest'ultimo.

Casi Di Prova		**Giustificazione**
Input	**Output**	
$[(0,0),(2,40),(4,80),(6,120),(0,160)]$	$3+3+3=9$	Classe equivalenza: passaggi tutti tra stanze permanenti in cui sto più di 30 secondi
$[(0,0),(1,40),(3,80),(5,120),(0,160)]$	$5+5+5=15$	Classe equivalenza: passaggi tutti tra stanze temporanee in cui sto più di 30 secondi
$[(0,0),(2,20),(4,40),(6,60),(0,80)]$	0	Classe equivalenza: passaggi tutti tra stanze permanenti in cui sto meno di 30 secondi
$[(0,0),(1,20),(3,40),(5,60),(0,80)]$	0	Classe equivalenza: passaggi tutti tra stanze temporanee in cui sto meno di 30 secondi
$[(0,0),(0,30),(0,60),(0,80),(0,120)]$	0	Classe di equivalenza: passaggi tutti tra stanze di servizio
$[(0,0),(1,20),(2,60),(0,100),(2,110),(0,120)]$	$5+3=8$	Classe di equivalenza: passaggio fra stanze miste (temporanee,esposizione, di servizio) con tempi variabili
$[(0,0),(2,30),(4,60),(6,90),(0,120)]$	$3+3+3=9$	Caso limite: passaggi tutti tra stanze permanenti in cui sto esattamente 30 secondi
$[(0,0),(1,30),(3,60),(5,90),(0,120)]$	$5+5+5=15$	Caso limite: passaggi tutti tra stanze temporanee in cui sto esattamente 30 secondi
$[(0,0),(1,20),(2,60),(1,60),(0,80)]$	5	Caso limite: caso in cui trovo due timestamp uguali consecutivi (faccio rapidamente avanti e indietro in una porta)
$[(0,0)]$	0	Caso limite: nessun passaggio effettuato
$[(0,0),(1,20),(2,10),(0,20)]$	*ERRORE*	Errore/Defensive programming: il dispositivo segna orari non coerenti
$[(1,20),(2,60),(0,100),(2,110)]]$	*ERRORE*	Errore/Defensive programming: il dispositivo non segna come prima o ultima sala la biglietteria

10.1.2 Correzione codice

Eseguendo il Walkthrough del codice, ci siamo subito accorti di alcuni errori di sintassi, che sarebbe comunque stati segnalati dal compilatore:

- manca la definizione della variabile `i` (`riga 4`),
- mancano le parentesi per la guardia dell'`if` (`riga 7 e 10`),
- manca una parentesi graffa prima del `return` (`riga 15`) per chiudere o il `for` (`riga 4`) o l'`if` (`riga 5`),
- l'operatore di incremento `=+` è invertito (`riga 8`)

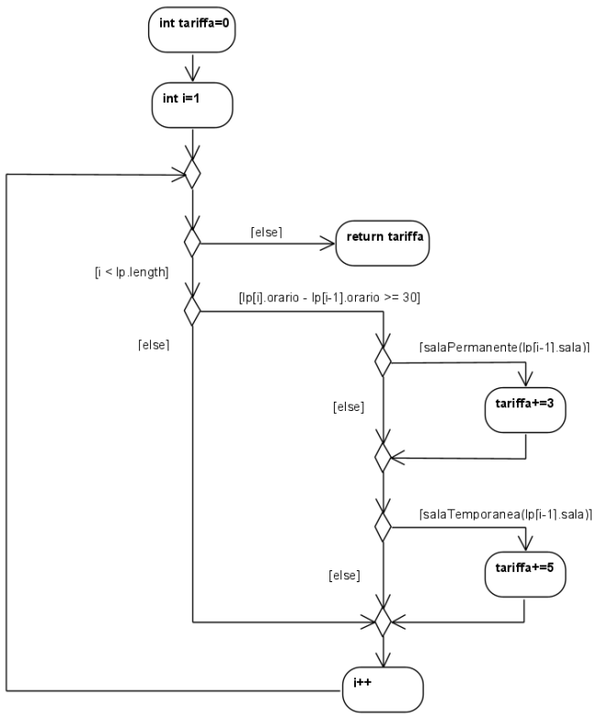
Una prima analisi statica ci ha dunque portati al seguente codice:

```
int calcolaTariffaBianca ([Passaggio] lp){
  int tariffa = 0;
  // INT i --> Corretto
  for (int i = 1; i < lp.length; i++){
    if (lp[i].orario - lp[i-1].orario >= 30){
      // (<if guard>) --> Corretto
      if (salaPermanente(lp[i-1].sala))
        tariffa += 3; // tariffa += 3 --> Corretto
      // (<if guard>) --> Corretto
      if (salaTemporanea(lp[i-1].sala))
        tariffa += 5;
    }
  // } --> Corretto
  }
  return tariffa;
}
```

10.2 Punto B

10.2.1 Assunzioni

Per evitare ambiguità relative al flusso di esecuzione dei comandi abbiamo utilizzato **il diagramma delle attività** per descrivere il grafo di flusso del metodo `int calcolaTariffaBianca([Passaggio] lp)`. Con questa rappresentazione ogni nodo del grafo ha una semantica ben definita.



10.3 Punto C

Il seguente caso di test garantisce una copertura totale (100%) del decisioni sul grafo di flusso.
Il grafo è costruito sulla base del codice corretto del punto precedente (10.1.2).

Casi Di Prova		**Cammino seguito (Numeri indicanti le righe del codice 10.1.2)**
Input	**Output**	
$[(0,0),(1,20),(2,60),(0,100)]$	$5+3=8$	$2, 4(T)\rightarrow5(F)\rightarrow4(T)\rightarrow5(T)\rightarrow7(F)\rightarrow10(T)\rightarrow11, 4(T)\rightarrow5(T)\rightarrow7(T)\rightarrow8, 10(F)\rightarrow4(F)\rightarrow15$

Vengono esercitati entrambi i rami di tutte le condizioni dei comandi decisionali $(4, 5, 7, 10)$.

10.4 Punto D

Nel caso in cui gli stub, `bool salaPermamente()` e `bool salaTemporanea()`, non siano conformi alla specifica i metodi potrebbero non ritornare `True` in maniera mutualmente esclusiva. Potrebbero ritornare entrambi `True` per uno stesso record di passaggio, ed in tal caso la stanza verrebbe contata sia come temporanea che come permanente.

Casi Di Prova		**Possibile Output con stub non conforme alla specifica**
Input	**Output Corretto**	
$[(0,0),(1,20),(2,60),(0,100)]$	$5+3=8$	$5+3+5+3=16$