UNIVERSITÀ DEGLI STUDI DI PISA

UNIVERSITÀ DI PISA

CORSO DI LAUREA IN INFORMATICA (L-31)

LAUREA TRIENNALE

Ingegneria del Software B – A.A. 2020-2021

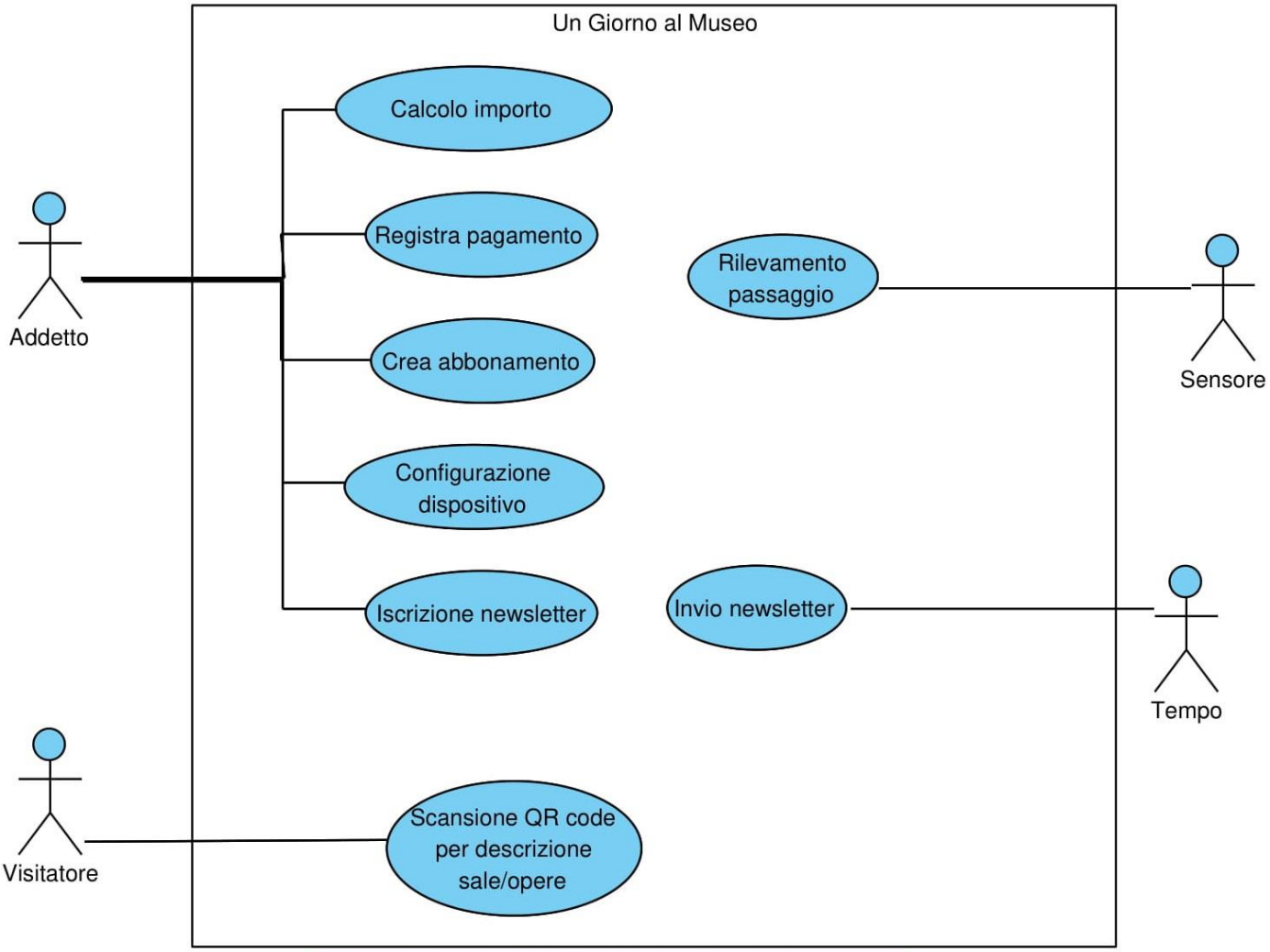
Un giorno al Museo

Membri:
Ambra Manattini - Mat : 601371
Alberto Baldini - Mat : 596681
Davide Borghini – Mat : 596431
Emiliano Sescu - Mat : 596883
Lia Trapanese - Mat : 628153

Data consegna:

31/05/2021

1)

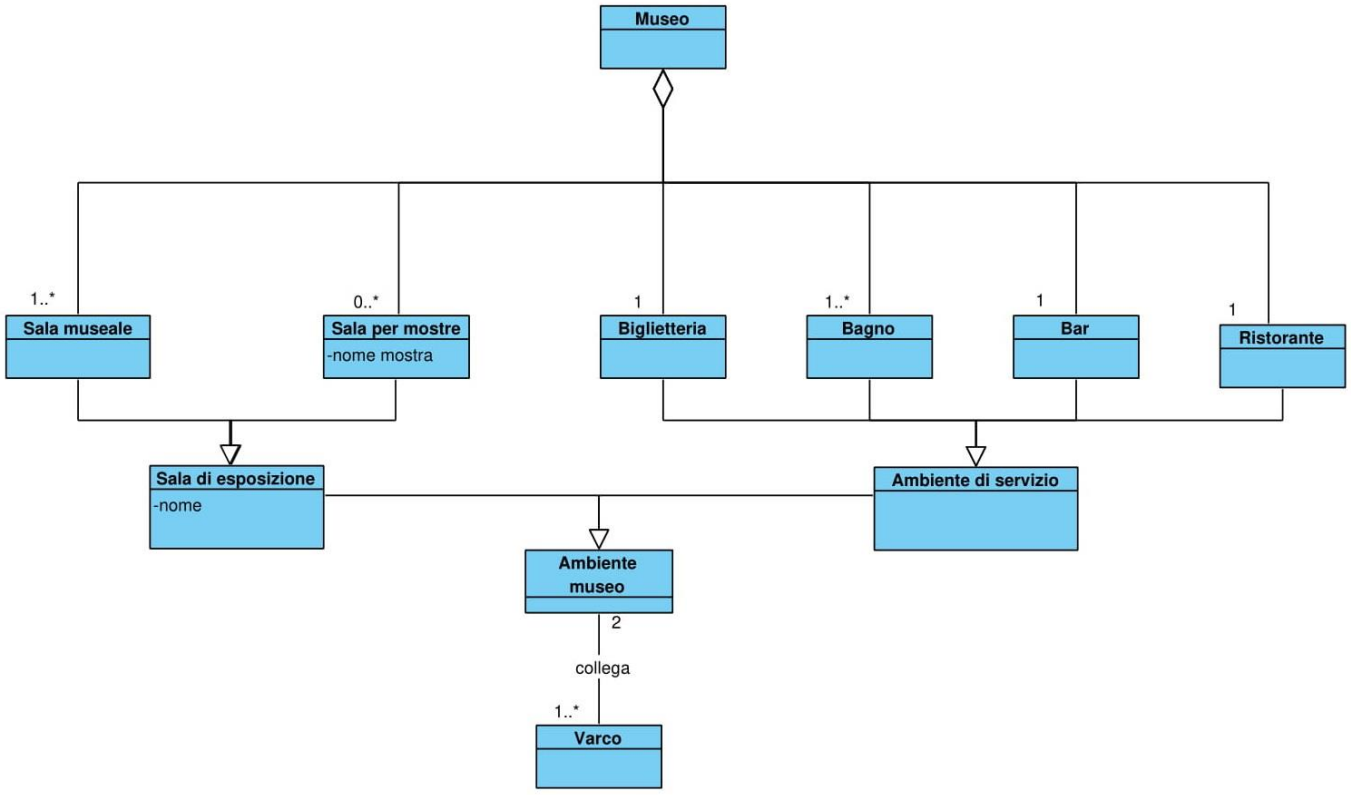


Abbiamo deciso che l'iscrizione alla newsletter può avvenire richiedendo ad uno degli addetti in quanto il cliente ha lasciato libera scelta

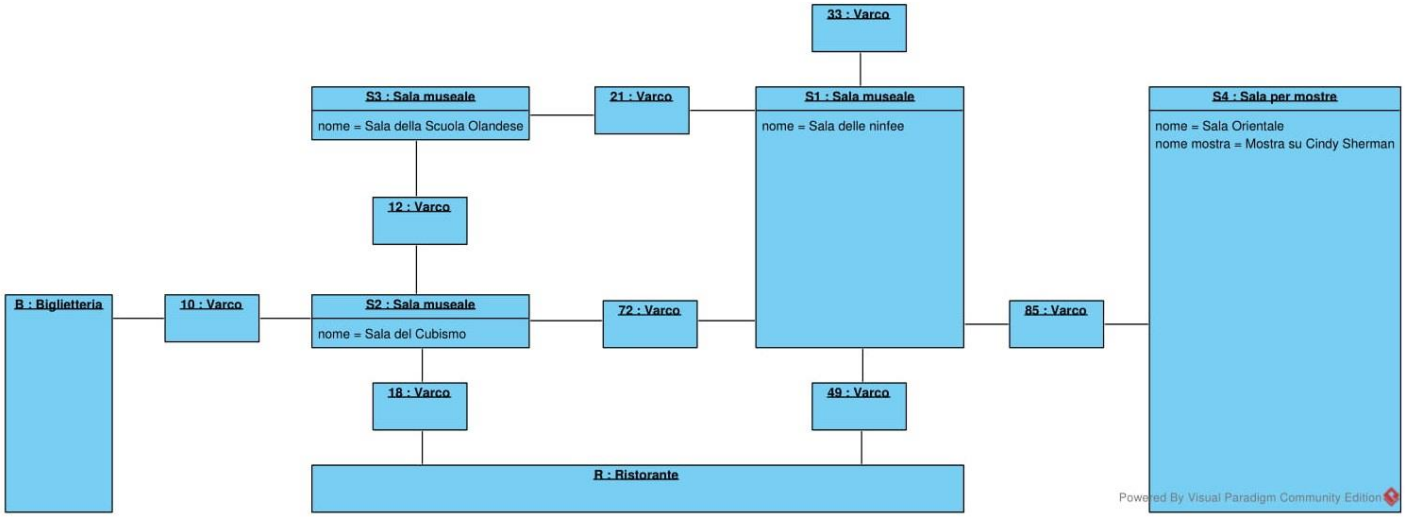
La configurazione del dispositivo comprende la scelta sia della tariffa sia della lingua ed il tipo di descrizione

Nome	Calcolo importo
Descrizione	Il sistema calcola l'importo e aggiorna eventuali abbonamenti
Attore primario	Addetto
Attore secondario	Nessuno
Precondizione	L'utente ha consegnato il dispositivo precedentemente configurato
Sequenza principale	1) L'addetto scansiona il QR code con il dispositivo 2) il sistema calcola i minuti passati nelle sale e il numero di sale visitate 3) if(tariffa == abbonato) 3.1) il sistema scala dall'abbonamento minuti e sale 3.2) if(abbonamento sforato per minuti) 3.2.1) il sistema calcola il prezzo in base ai minuti in eccesso 3.3) elseif(abbomanento sforato per sale) 3.3.1)il sistema calcola il prezzo in base alle sale in eccesso 3.4) if(abbonamento finito) 3.4.1) invalida abbonamento 4) elseif(tariffa == verde) 4.1) il sistema calcola importo in base al tempo passato nel museo 5) elseif(tariffa == bianco) 5.1) il sistema calcola importo in base alle sale visitate 6) il sistema stampa l'importo
Postcondizione	il sistema ha calcolato e stampato l'importo AND l'abbonamento, se usato, è stato aggiornato
Sequenza alternativa	scannerizzazione QR code fallita

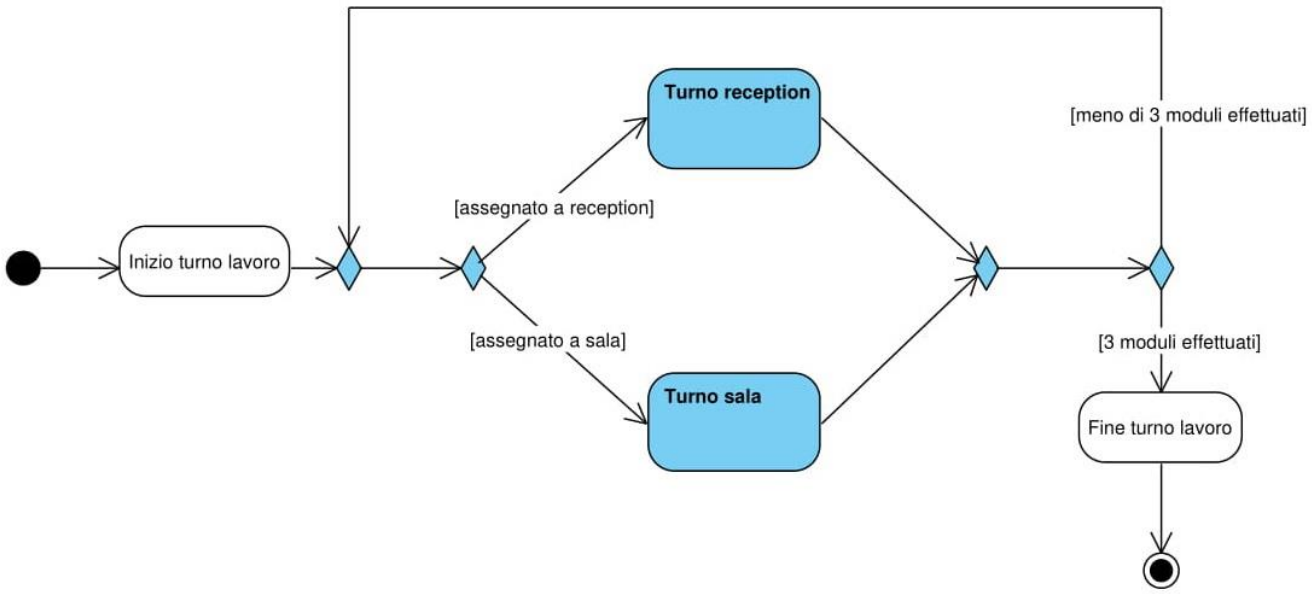
2)

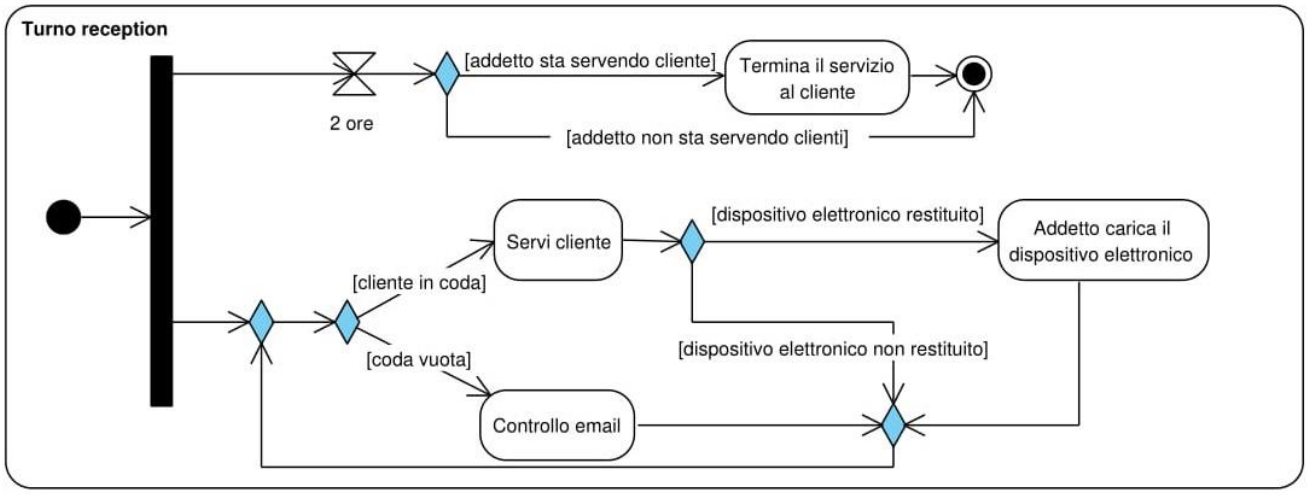


3)

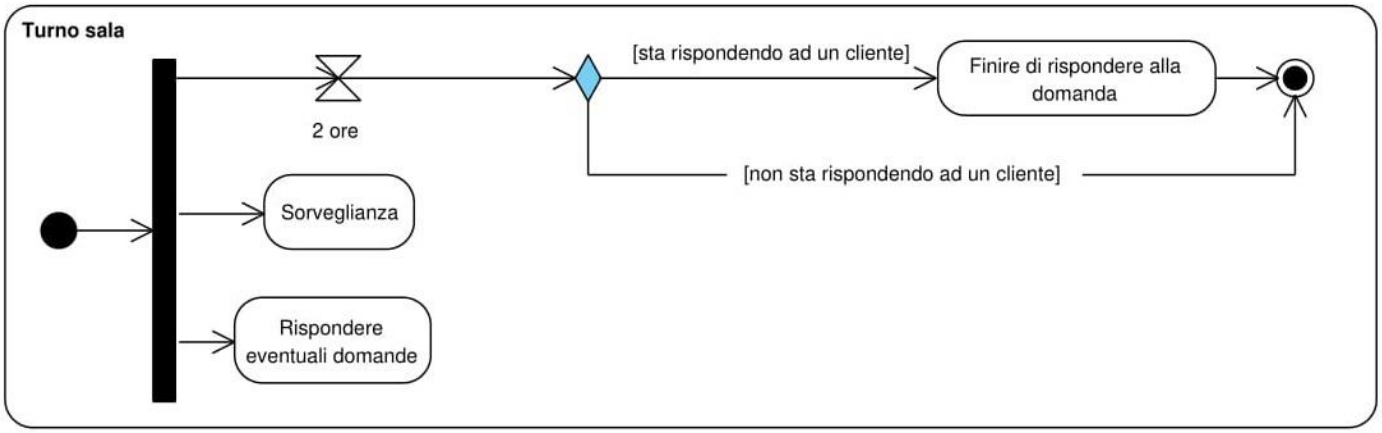


4)

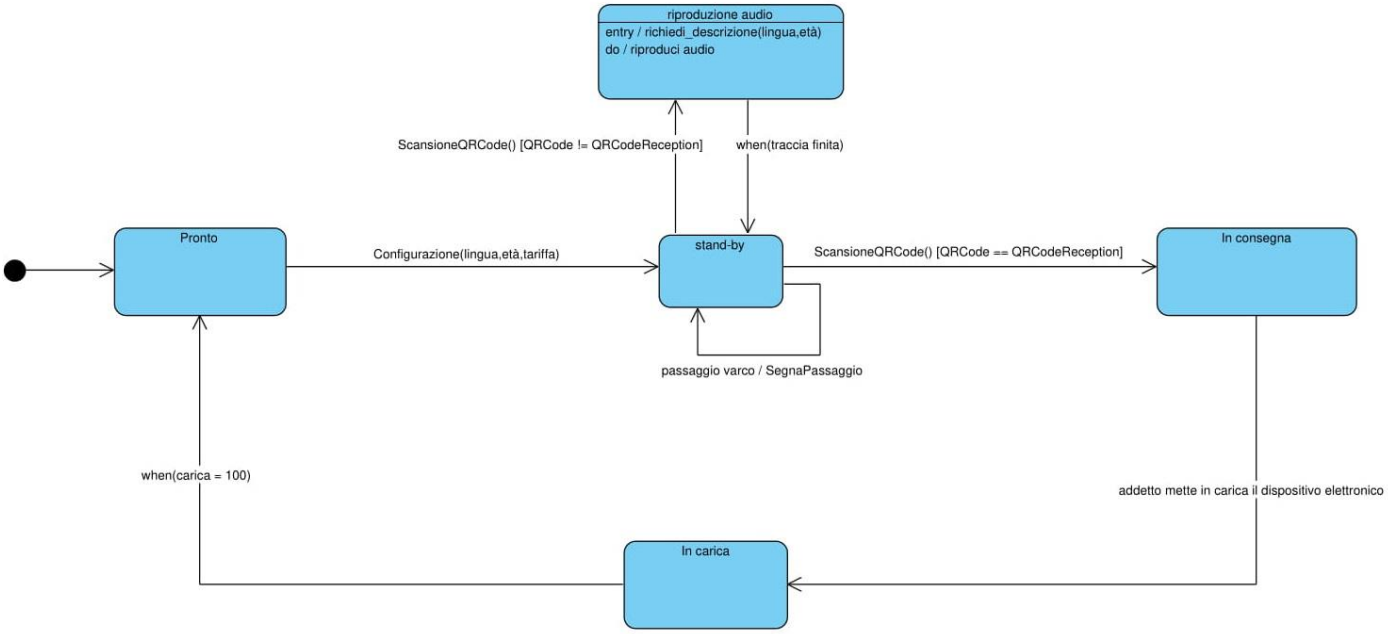




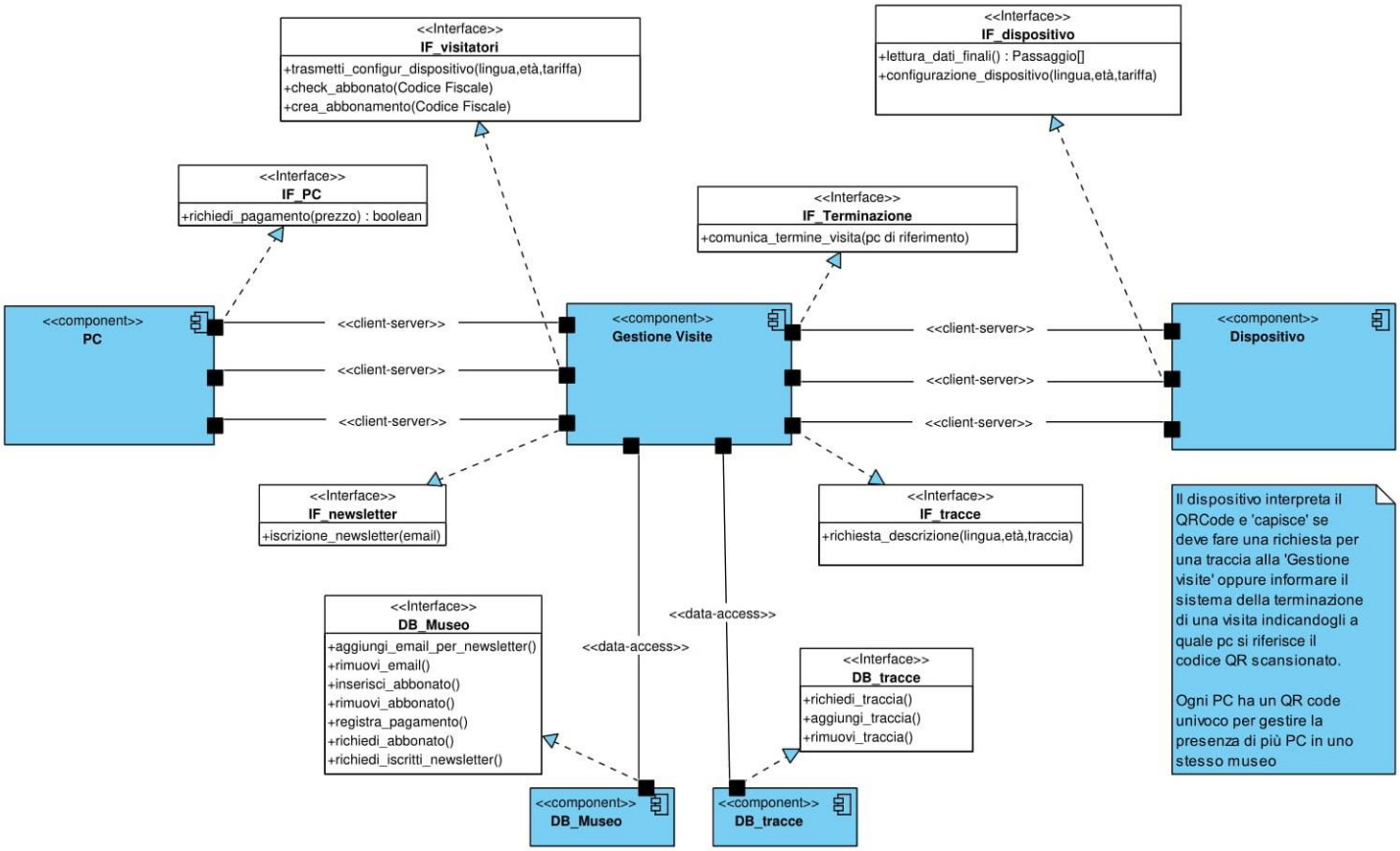
Ipotizziamo che i dispositivi vengono messi in carica mano a mano che vengono riconsegnati


Turno sala
2 ore
[sta rispondendo ad un cliente]
Finire di rispondere alla domanda
[non sta rispondendo ad un cliente]
Sorveglianza
Rispondere eventuali domande

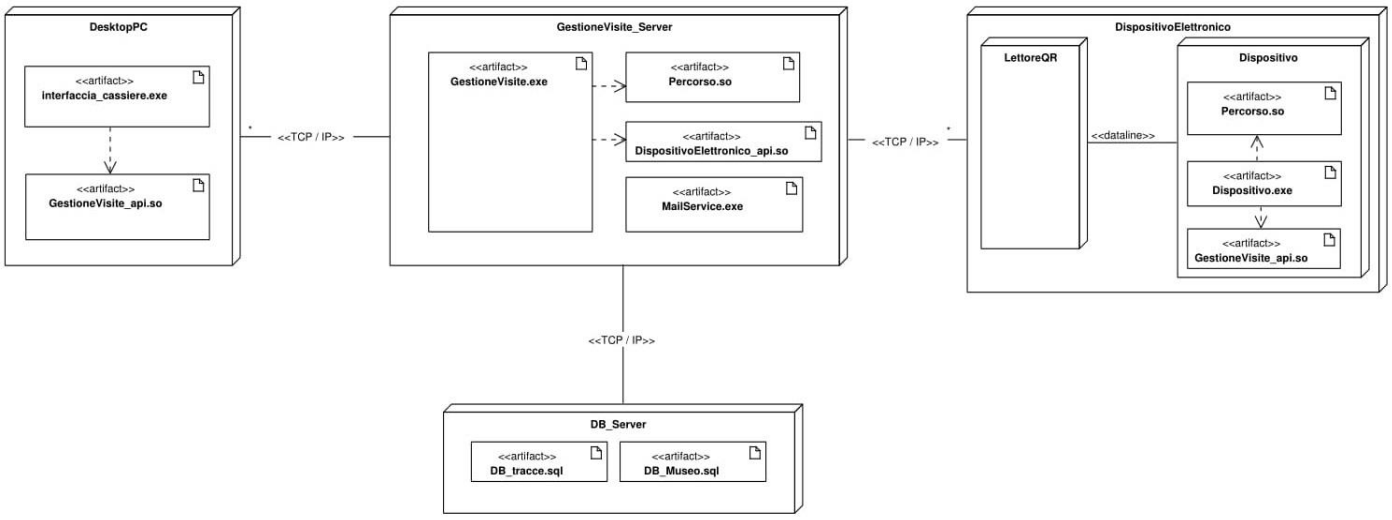
5)


riproduzione audio
entry / richiedi_descrizione(lingua,età)
do / riproduci audio
ScansioneQRCode() [QRCode != QRCodeReception]
when(traccia finita)
Pronto
Configurazione(lingua,età,tariffa)
stand-by
ScansioneQRCode() [QRCode == QRCodeReception]
In consegna
passaggio varco / SegnaPassaggio
when(carica = 100)
addetto mette in carica il dispositivo elettronico
In carica

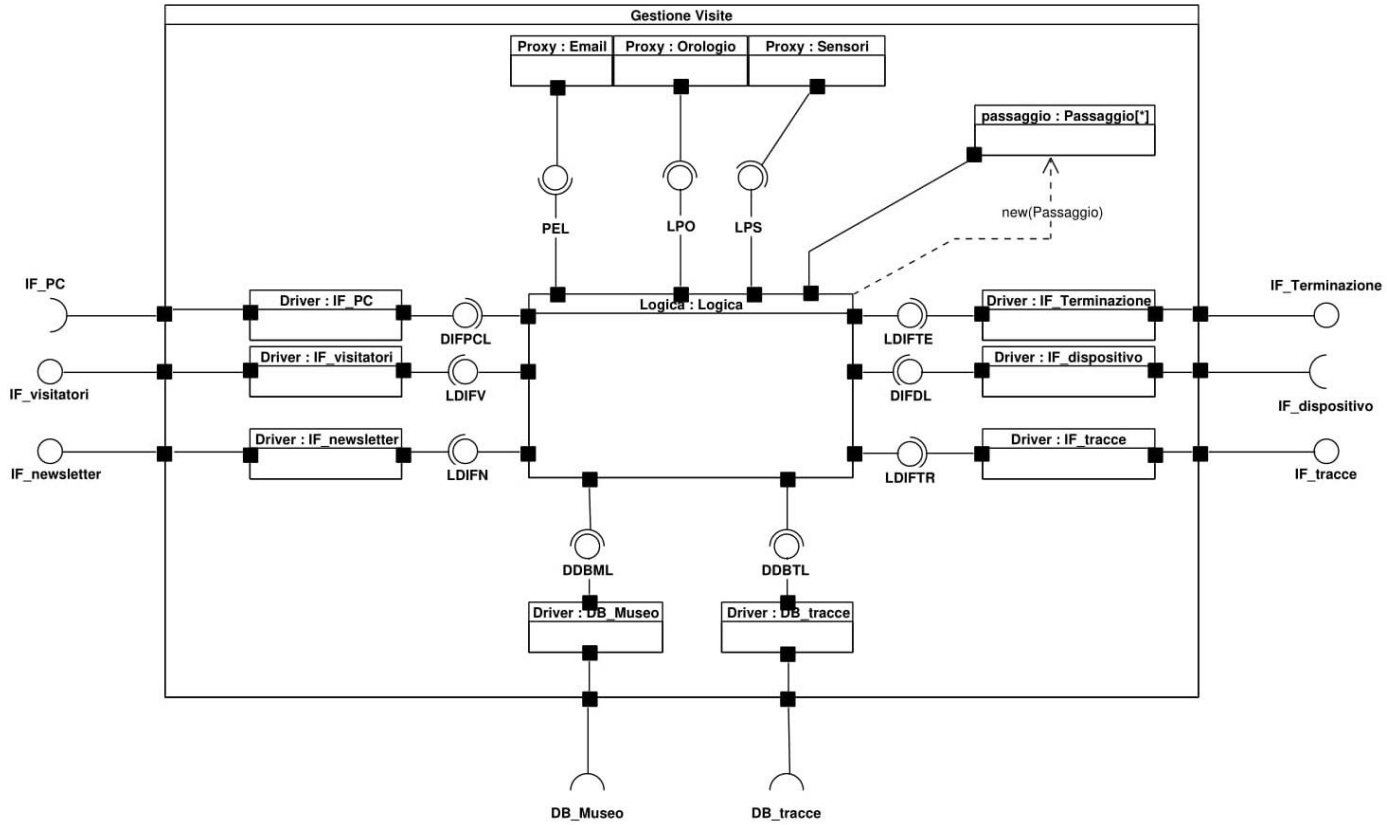
6)


<<Interface>>
IF_visitatori
+trasmetti_configur_dispositivo(lingua,età,tariffa)
+check_abbonato(Codice Fiscale)
+crea_abbonamento(Codice Fiscale)
<<Interface>>
IF_dispositivo
+lettura_dati_finali() : Passaggio[]
+configurazione_dispositivo(lingua,età,tariffa)
<<Interface>>
IF_PC
+richiedi_pagamento(prezzo) : boolean
<<Interface>>
IF_Terminazione
+comunica_termine_visita(pc di riferimento)
<<component>>
PC
<<client-server>>
<<client-server>>
<<client-server>>
<<component>>
Gestione Visite
<<client-server>>
<<client-server>>
<<client-server>>
<<component>>
Dispositivo
<<Interface>>
IF_newsletter
+iscrizione_newsletter(email)
<<Interface>>
IF_tracce
+richiesta_descrizione(lingua,età,traccia)
Il dispositivo interpreta il QRCode e 'capisce' se deve fare una richiesta per una traccia alla 'Gestione visite' oppure informare il sistema della terminazione di una visita indicandogli a quale pc si riferisce il codice QR scansionato.
Ogni PC ha un QR code univoco per gestire la presenza di più PC in uno stesso museo
<<Interface>>
DB_Museo
+aggiungi_email_per_newsletter()
+rimuovi_email()
+inserisci_abbonato()
+rimuovi_abbonato()
+registra_pagamento()
+richiedi_abbonato()
+richiedi_iscritti_newsletter()
<<data-access>>
<<data-access>>
<<Interface>>
DB_tracce
+richiedi_traccia()
+aggiungi_traccia()
+rimuovi_traccia()
<<component>>
DB_Museo
<<component>>
DB_tracce

7)


DesktopPC
<<artifact>>
interfaccia_cassiere.exe
<<artifact>>
GestioneVisite_api.so
<<TCP / IP>>
GestioneVisite_Server
<<artifact>>
GestioneVisite.exe
<<artifact>>
Percorso.so
<<artifact>>
DispositivoElettronico_api.so
<<artifact>>
MailService.exe
<<TCP / IP>>
DispositivoElettronico
LettoreQR
<<dataline>>
Dispositivo
<<artifact>>
Percorso.so
<<artifact>>
Dispositivo.exe
<<artifact>>
GestioneVisite_api.so
<<TCP / IP>>
DB_Server
<<artifact>>
DB_tracce.sql
<<artifact>>
DB_Museo.sql

8)



9)

a)

Partizione valori in ingresso

t è definito come la differenza tra l'orario di entrata nella stanza i + 1 e di entrata dalla stanza i.
(es. t' = o'' - o')

T : (t < 0, 0 <= t < 30, t >= 30) , dove orario = [o, o', o'', ...]

Stanze: (Temporanea, Permanente, Servizio, non definite)

Combinazioni

Le combinazioni possibili sono 3 * 4 = 12, rimuovendo le classi di errore si ottengono 2+2*3= 8.
Effettuando un solo test per i casi in cui un visitatore passa meno di 30 secondi in un ambiente qualsiasi otteniamo un totale di 6 test.
Aggiungendo i casi soglia (t = 30, t = 29, t = 0) otteniamo 9 casi da testare.

Dobbiamo effettuare test con array di dimensioni diverse: 0, 2, 3, 4,…
Si hanno 0 sale visitate se non si lascia la biglietteria.
Non si può avere una sola sala visitata in quanto una volta entrati nel museo bisogna uscire, pertanto i test più significativi avranno almeno 2 valori.

Il totale dei test sarà: 3(errori) + 1(array vuoto) + 4(combinazioni stanze/orario) + 3(valori soglia).

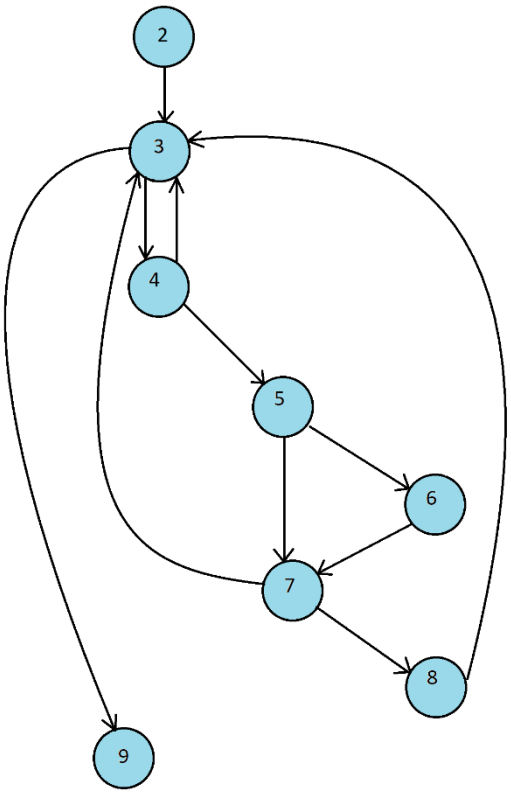
Input <[(secondi, sala)]>	Output	Commenti
[(10, Realismo), (45, Biglietteria)]	Errore	Stanza non definita
NULL	Errore	Array non fornito
[(10, Cubismo), (9, Biglietteria)]	Errore	Valore di tempo passato in una stanza negativo
{}	0	Il visitatore non ha lasciato la biglietteria
[(10, Cubismo), (60, Ristorante), (95, Cubismo), (150, Biglietteria)]	6	Il caso testa due delle combinazioni: (t > 30, sala permanente) (t > 30, ambiente servizio)
[(10, Cubismo), (12, Ninfee), (17, SalaTemp), (140, Ninfee), (141, Cubismo), (142, Biglietteria)]	5	Il caso testa due delle combinazioni: (t < 30, stanza generica) (t > 30, sala temporanea)
[(10, Cubismo), (10, Ninfee), (39, Cubismo), (68, Biglietteria)]	0	Il caso testa i valori soglia di: (t=0, sala) (t=29, sala)
[(10, Cubismo), (40, Biglietteria)]	3	Il caso testa il valore soglia di: (t=30, sala)

STUB

```
public class Passaggio {
	public Passaggio(String stanza, int orario) {
		this.stanza = stanza;
		this.orario = orario;
	}
	public String stanza;
	public int orario;
}
public boolean salaPermanente(String stanza) {
	String[] saleperm = {"Ninfee", "Cubismo", "Olandesi"};
	return saleperm.contains(stanza);
}
public boolean salaTemporanea(String stanza) {
	String[] saletemp = {"SalaTemp"};
	return saletemp.contains(stanza);
}
```

b)

GRAFO DI FLUSSO




```
1 int calcolaTariffaBianca (Passaggio[] lp){
2         int tariffa = 0;
3         for (i = 1; i < lp.length; i++){
4             if (lp[i].orario - lp[i-1].orario >= 30){
5                 if salaPermanente(lp[i-1].sala)
6                     tariffa  += 3;
7                 if salaTemporanea(lp[i-1].sala)
8                     tariffa += 5;
                  }
    }
9         return tariffa;
  }
```

c)

I test definiti nel punto A garantiscono il 100% della copertura delle decisioni.

Input <[(secondi, sala)]>	**Archi attraversati**
[(10, Cubismo), (60, Ristorante), (95, Cubismo), (150, Biglietteria)]	(2,3), (3,4), (4,5), (5,6), (6,7), (7,3) … (3,9)
[(10, Cubismo), (12, Ninfee), (17, SalaTemp), (140, Ninfee), (141, Cubismo), (142, Biglietteria)]	(2,3), (3,4), (4,3), (3,4), (4,3), (3,4), (4,5), (5,7), (7,8), (8,3) … (3,9)

d)

Supponiamo che, i nostri stub restituiscano TRUE sia nei casi corretti, sia nel caso in cui la stanza sia un ambiente di servizio.

Considerando il test case:

Valore di i	**Valori lp[i-1] dell'iterazione del for**	**Risposte stub**	**Incremento tariffa**	**Incremento atteso**
1	(10, Cubismo)	(TRUE, FALSE)	3	3
2	(60, Ristorante)	(TRUE, TRUE)	3 + 5	0
3	(95, Cubismo)	(TRUE, FALSE)	3	3
4	(150, Biglietteria)			