Ingegneria del Software A.A. 2018/2019
Esame 2019-04-19

Esercizio 1 (6 punti)

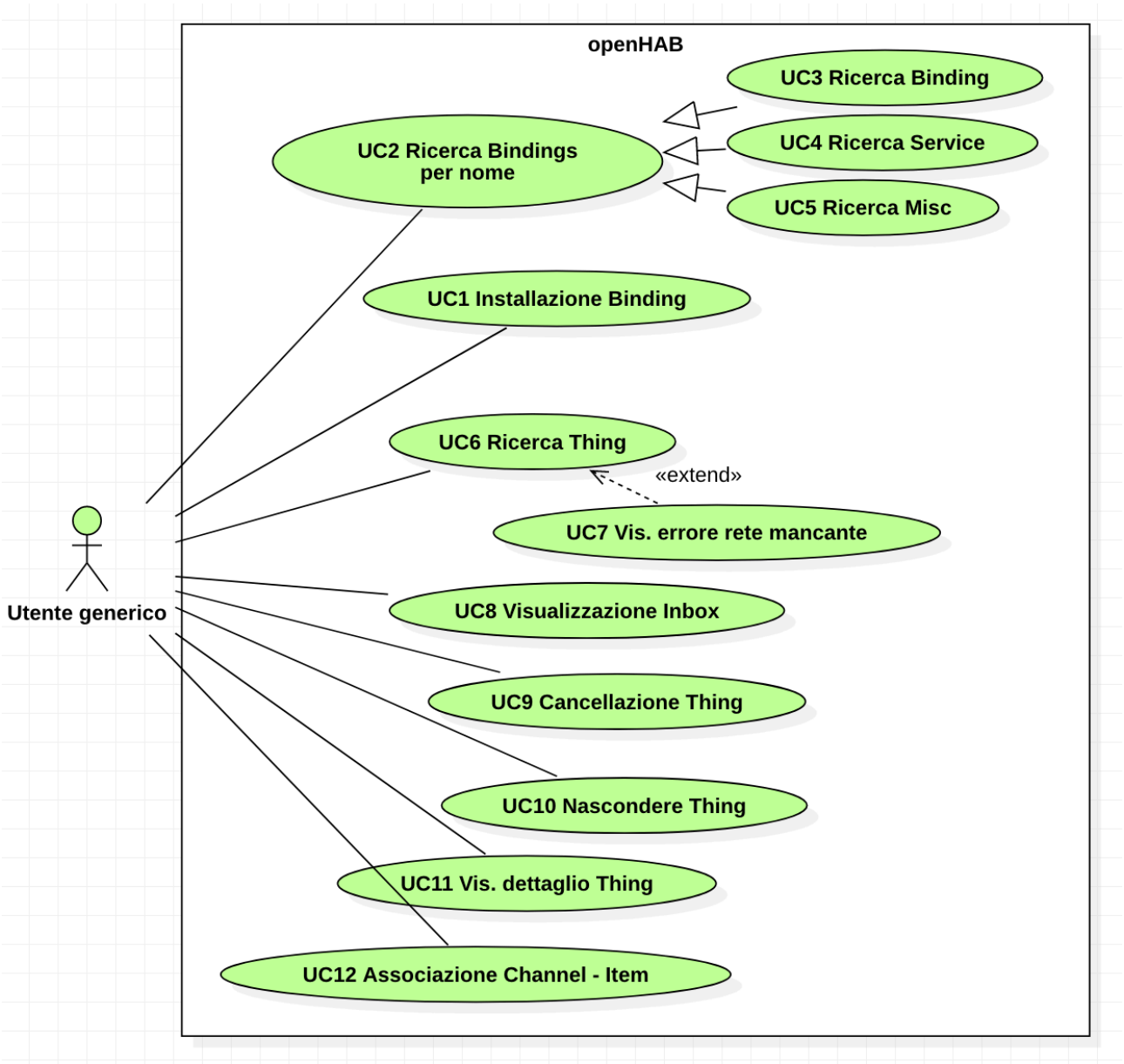
Descrizione

OpenHAB è un software open utilizzato per gestire in modo non dipendente dai *vendor* la domotica casalinga e non. Si base sul concetto di astrazione. I dispositivi fisici vengono astratti all'interno del software come *Thing*. Le funzionalità esposte da ogni *Thing* sono dette *Channel*. Ad un *Channel* è possibile collegare un *Item*, che è la rappresentazione software della funzionalità associata ad un *Thing*. Il software di openHAB lavora a moduli. Attraverso un'interfaccia web è possibile aggiungere (installare) i cosiddetti *Binding*, ossia il software specifico per dialogare con i dispositivi di un particolare *vendor*. I *Binding* possono essere ricercati utilizzando un nome e sono suddivisi per categoria (*Bindings*, *Services*, *Misc*, …). Una volta installato un *Binding* è possibile iniziare una ricerca di tutti i dispositivi collegati alla medesima rete locale del server openHAB. La ricerca visualizza tutti i *Thing* trovati in una lista, detta *Inbox,* oppure visualizza un errore se non è presente alcun collegamento alla rete locale. Nella lista è possibile visualizzare il nome del *Thing* ed una breve descrizione. Inoltre, è possibile cancellare l'oggetto oppure semplicemente nasconderlo. Nella pagina di dettaglio di un *Thing* è possibile visualizzare il suo stato (online/offline), una sua descrizione e la lista dei *Channel* ad esso associati. In questa pagina avviene l'associazione fra un *Channel* e un rispettivo nuovo *Item*.

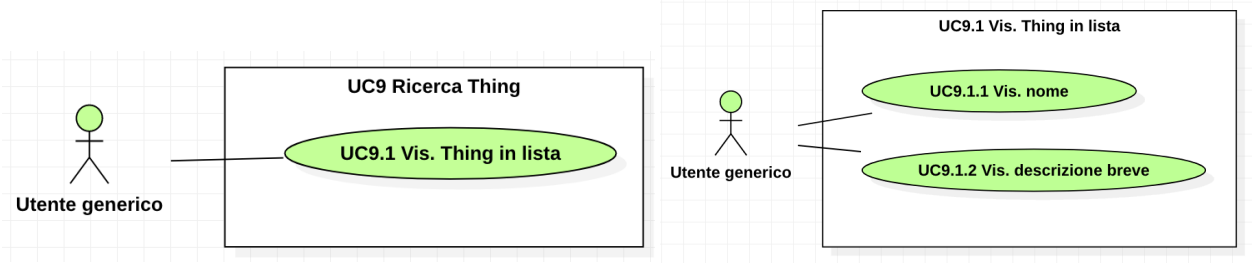
Si utilizzino i diagrammi dei casi d'uso per modellare gli scenari sopra descritti. Non ne è richiesta la descrizione testuale.

Soluzione

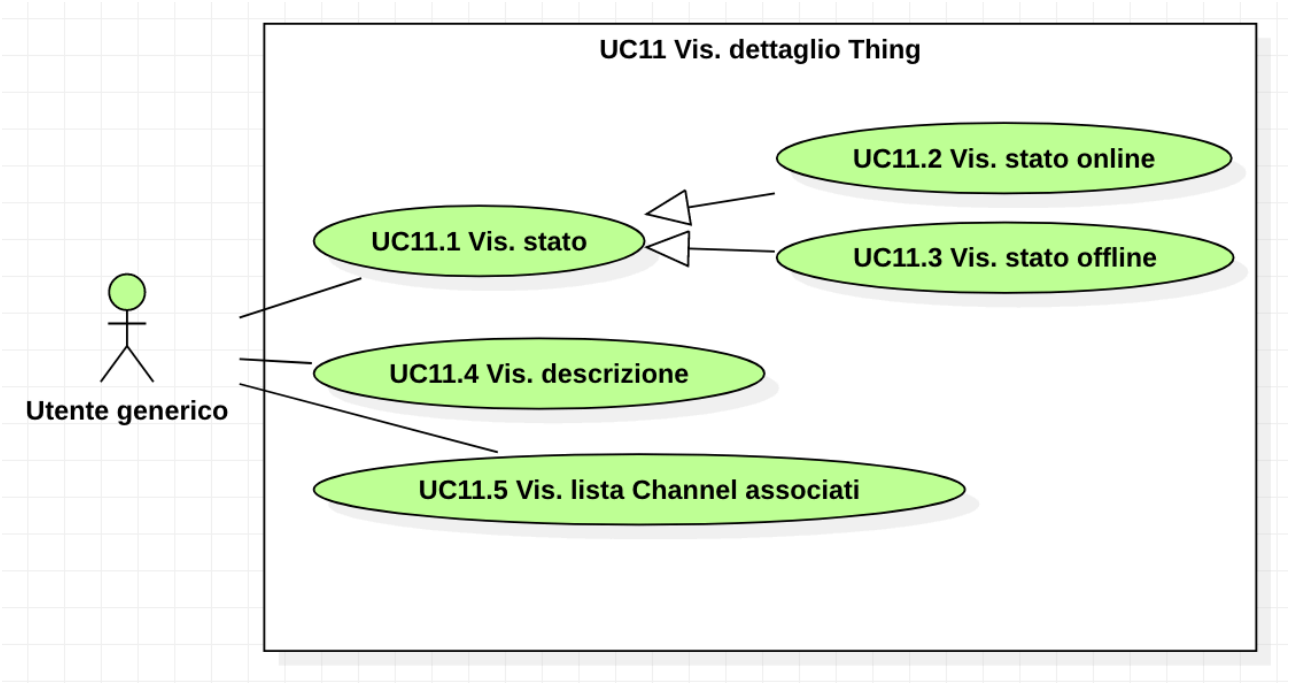
Il diagramma dei casi d'uso principale corrispondente è il seguente.



La visualizzazione dei Thing in lista è modellata come segue.



Infine, la visualizzazione del dettaglio di un Thing può essere modellata come segue.



Esercizio 2 (7 punti)

Descrizione

La rilevazione (*discovery*) di nuovi *Thing* in openHAB avviene con modalità *publish*/*subscribe*. Ogniqualvolta un nuovo dispositivo *hardware* per il quale è installato un *Binding* si registra nella medesima rete locale del *server* openHAB, esso lo rileva e crea un corrispondente *Thing* al suo interno. Il processo di rilevazione opera per *Binding*. Ogni *Thing* si interfaccia con una specifica istanza del *driver* messo a disposizione dal *Binding*. Il potere espressivo di openHAB sta proprio nell'uso del medesimo oggetto, *Thing*, per governare l'interazione con dispositivi diversi di *vendor* diversi, adattando di volta in volta solo lo specifico protocollo. Ogni *Thing* può essere di tipo *switch*, *roller shutter*, e *thermostat*. Ogni tipologia di *Thing* ha poi il proprio modo standard di adattare il *driver* fornito dai *Binding*.

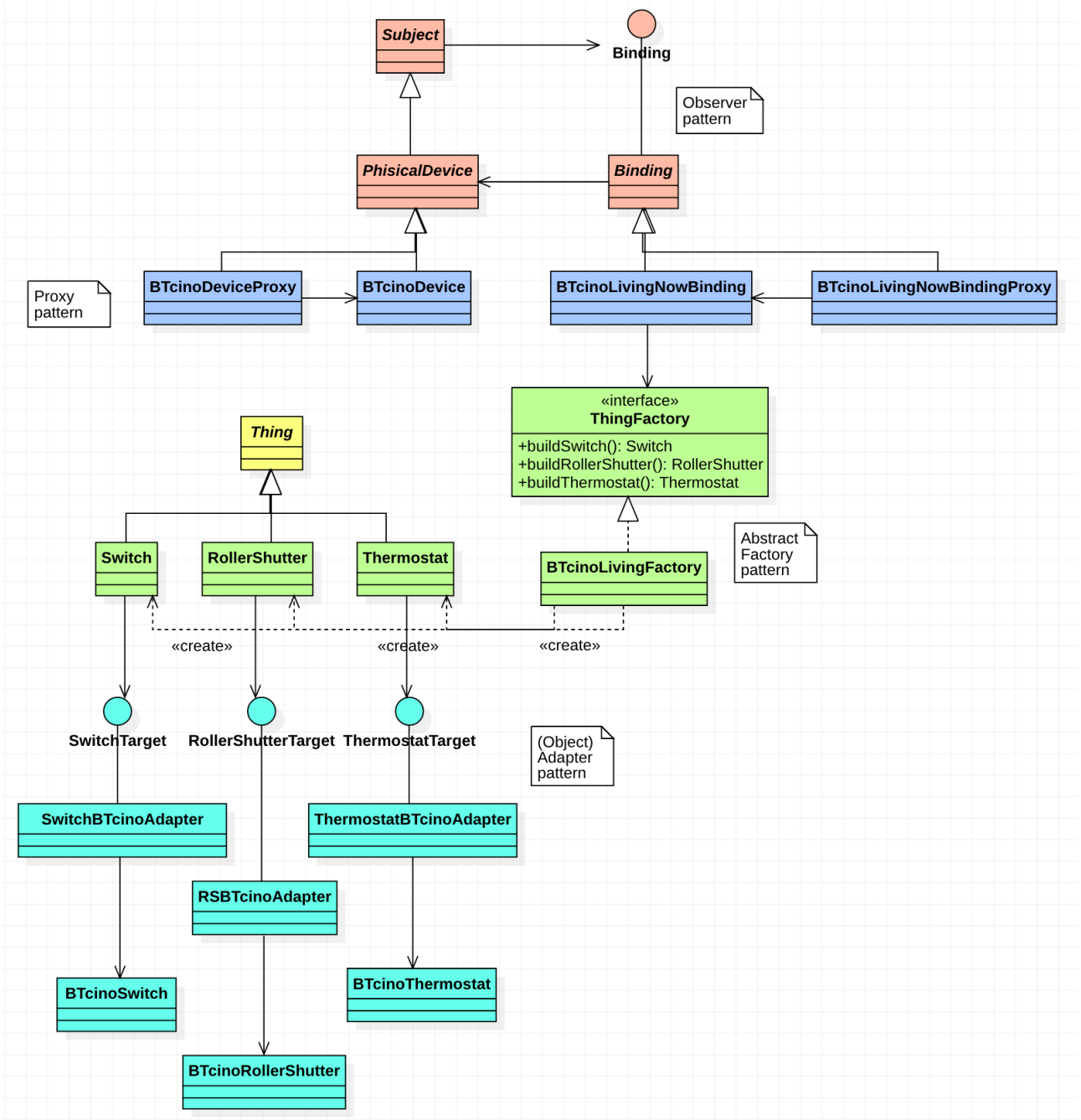
Si modelli tale sistema mediante un diagramma delle classi e i *design pattern* a esso pertinenti. Utilizzando un diagramma di sequenza, si descriva poi l'aggiunta di un nuovo *Thing* di tipo *switch* del *vendor* BTicino Living Now.

Soluzione

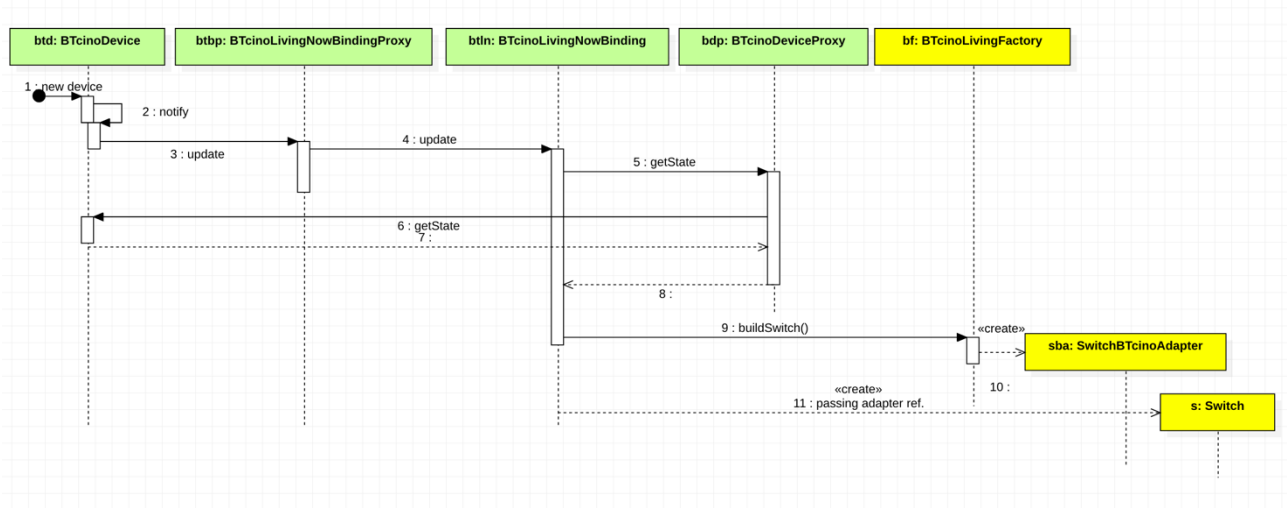
La soluzione prevede l'utilizzo dei seguenti design pattern:

- Observer
- Proxy
- Abstract factory
- Adapter (object)

Un possibile diagramma delle classi è il seguente.



Il diagramma di sequenza richiesto è invece il seguente.



Esercizio 3 (3 punti)

Descrizione

L'installazione di una lista di nuovi *Binding* all'interno del sistema opanHAB avviene utilizzando il seguente codice Java.

```
public class Binding {
    private final String bindingType;
    public static void install(Binding[] bindings) {
        for (Binding b: bindings) {
            switch (b. bindingType) {
                case "Bticino":
                    ((BTicinoBinding) b).installBTicino();
                    break;
                case "Hue":
                    ((PhilipsHueBinding b)).installHue();
                    break;

            }
        }
    }
}
public class BTicinoBinding {
    // Code that initializes super bindingType
    public void installBTicino() {
```

```
        System.out.println("Installing BTicino binding");
    }
}
public class PhilipsHueBinding {
    // Code that initializes super bindingType
    public void installHue() {
        System.out.println("Installing Philips Hue binding");
    }
}
```

Si modifichi il suddetto codice al fine di fare aderire il metodo install al principio *Open-Closed*.

Soluzione

È necessario uniformare i metodi nelle classi concrete in una unica interfaccia `install`, dichiarata in `Binding`. In questo modo, è possibile estendere il sistema aggiungendo nuove tipologie di *Binding*, senza tuttavia modificare il codice esistente.

```
public abstract class Binding {
    public abstract void install();
    public static void install(Binding[] bindings) {
        for (Binding b: bindings) {
            b.install();
        }
    }
}
public class BTicinoBinding extends Binding {
    @Override
    public void install() {
        System.out.println("Installing BTicino binding");
    }
}
public class PhilipsHueBinding extends Binding {
    @Override
    public void install() {
        System.out.println("Installing Philips Hue binding"); }
    }
```

```
}
```