

GPMS
General Purpose Monitoring System
Specifica dei requisiti di progetto

Gabriele Masini #108456

2019-12-10 rev.2020-02-13

Indice

1	Introduzione	1
1.1	Obiettivo	1
1.2	Campo d'applicazione	1
1.3	Definizioni, acronimi, abbreviazioni	1
1.4	Fonti	2
1.5	Struttura del documento	2
2	Descrizione generale	3
2.1	Inquadramento	3
2.2	Macro funzionalità	4
2.3	Caratteristiche degli utenti	5
2.4	Vincoli e limiti	6
2.5	Presupposti e dipendenze	6
2.6	Requisiti da analizzare in futuro	6
3	Specificazione dei requisiti	7
3.1	Requisiti dell'interfaccia esterna	7
3.2	Requisiti funzionali	8
3.3	Requisiti non funzionali	9
4	Appendice	10
4.1	Tabelle dimensione stream registrati	10
4.2	Esempio di sintassi da riga di comando	10
4.3	Design	12
4.3.1	Use Case	12
4.3.2	Design pattern	12
4.3.3	Activity Diagram	17

1 Introduzione

1.1 Obiettivo

Il presente documento si riferisce alla definizione e descrizione delle informazioni necessarie per lo sviluppo del progetto *GPMS* e interessa tutte le entità coinvolte nella progettazione, sviluppo e utilizzo dello stesso. Il documento viene redatto secondo le direttive espresse nel *IEEE Recommended Practice for Software Requirements Specifications* [1], ovvero secondo lo standard *IEEE Std 830-1998*.

1.2 Campo d'applicazione

Il progetto nominato *GPMS* si inserisce nel contesto odierno di forte sviluppo di tecnologie relative all'*Internet of Things* e si pone come obiettivo la gestione dei dispositivi per scopi di monitoraggio e videosorveglianza, nonché di metodi comuni di registrazione dei sopra menzionati dispositivi, in modo da organizzare gerarchicamente la *pool* di dispositivi installati in un ambiente e di conseguenza semplificare il lavoro di manutenzione e gestione.

1.3 Definizioni, acronimi, abbreviazioni

Nel presente documento vengono spesso utilizzate abbreviazioni e acronimi comuni nel contesto di sviluppo del progetto. Di seguito vengono riportate le interpretazioni delle suddette abbreviazioni:

Amministratore	Persona o ente responsabile del mantenimento del <i>Server</i> . Spesso è anche un <i>Client</i> .
Browser Web	Programma destinato alla visualizzazione e navigazione di pagine web (ipertesti).
Client	Elaboratore che si collega ad un <i>Server</i> per poter usufruire dei servizi da esso offerti.
HTTP	<i>Hypertext Transfer Protocol</i> . Protocollo di trasferimento di documenti ipertestuali tramite web.
IOT	”Internet of Things”, l’aggregazione in internet di dispositivi di poca capacità di calcolo destinati come controllori per determinati elettrodomestici o utilizzati come sensori (telecamere, termometri, microfoni etc).
Java	Linguaggio orientato agli oggetti sviluppato da Sun Microsystems e successivamente acquisito da Oracle.
Java Virtual Machine (JVM)	Programma destinato all’interpretazione del bytecode di un programma Java.
JSON [2]	<i>Javascript object notation</i> . Rappresentazione di dati eterogenei tramite oggetti <i>Javascript</i> .

Server	Elaboratore che gestisce in modo centralizzato un insieme di dati e espone una interfaccia per interagire con più <i>Client</i> tramite uno o più protocolli.
Sistema Operativo	Programma destinato alla gestione delle risorse e sotto-programmi di un elaboratore..
Stream	Flusso di dati di vario tipo (audio, video, audio-video etc.).
Utente	Persona o ente che tramite un <i>Client</i> utilizza l'applicazione.

1.4 Fonti

- [1] IEEE, “IEEE Recommended Practice for Software Requirements Specifications,” Software Engineering Standards Committee of the IEEE Computer Society, rapp. tecn., 1998.
- [2] ISO/IEC, “Information technology — The JSON data interchange syntax,” International Organization for Standardization, International Electrotechnical Commission, rapp. tecn., 2017.
- [3] E. Gamma, R. Helm, R. Johnson e J. Vlissides, *Design Patterns, Elements of Reusable Object-Oriented Software*. 1994.
- [4] World Wide Web Consortium. (2017). HTML5.2 w3c recomendations, indirizzo: <https://www.w3.org/TR/2017/REC-html52-20171214/> (visitato il 09/02/2020).
- [5] European Parliament e European Union Council. (2016). REGULATION (EU) 2016/679 General Data Protection Regulation, indirizzo: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679> (visitato il 13/02/2020).

1.5 Struttura del documento

Il presente documento contiene le specifiche e le relative descrizioni dei requisiti del progetto *GPMS*. Il progetto viene inquadrato secondo quanto citato in questa sezione e si prosegue nella sezione successiva a descriverne la prospettiva e le funzionalità, nonché la caratterizzazione dei potenziali utenti, dei vincoli e delle dipendenze. Nella sezione *Specifiche dei requisiti*, verranno spiegate le interfacce con l'esterno e i requisiti funzionali e non. Alla fine del documento, in *Appendice*, verranno inseriti diagrammi e grafici per integrare quanto detto nelle prossime sezioni, oltre a proporre un possibile utilizzo dei celeberrimi *design pattern* [3] per la programmazione ad oggetti del progetto.

2 Descrizione generale

2.1 Inquadramento

Il progetto *GPMS* è inteso come una applicazione *stand alone* programmata in *Java* che permetta di gestire una aggregazione di dispositivi atti a monitorare lo stato di uno o più ambienti (ad esempio: una abitazione, un ufficio, un parco etc). L'applicazione è un programma che viene eseguito su una macchina *Server* ed essa espone all'esterno una interfaccia web (*http*) per l'interazione con l'utente; inoltre il programma presenta una interfaccia sul server di tipo “a riga di comando” per la gestione da parte dell'amministratore.

Interfaccia sistema/utente L'interfaccia web supporta monitor in 4:3 e 16:9 di almeno 1024x768 pixel (XGA) e presenta la gerarchia dei dispositivi e gruppi, i form di modifica della stessa e le visualizzazioni dei dati, ovvero forma d'onda e suono per dispositivi audio, video per i dispositivi video e grafici per i sensori (es. temperatura). I messaggi di errore nell'interfaccia web sono tenuti ad una lunghezza minima, specificando, quindi, un codice di errore e un messaggio umanamente comprensibile.

L'interfaccia a riga di comando, invece, non presenta nessun tipo di gerarchia ma viene tutto gestito tramite testo e comandi. In questo caso i messaggi di errore sono più lunghi in modo da facilitare il lavoro dell'amministratore.

Interfaccia hardware Il servizio web utilizza la porta 80 per l'accesso da parte dei client web; la comunicazione coi dispositivi avviene tramite socket e la comunicazione viene iniziata dal server (per cui non sono necessarie porte consciute aperte per la comunicazione coi dispositivi). Le macchine supportate sono quelle ad architettura x86 o amd64 con sistema operativo Microsoft Windows, Linux o macOS. L'applicazione necessita di almeno una interfaccia di rete installata e configurata correttamente sulla macchina *Server* per la comunicazione coi dispositivi da monitorare e l'accesso da parte dei *Client*.

Interfaccia software Il software utilizza un database *CouchDB* per la memorizzazione dei dati degli utenti e delle risorse, che può essere installato sulla macchina *Server* stessa o su una macchina remota accessibile. Per quanto concerne l'elaborazione degli *stream* audio e video, l'applicazione si avvale del progetto *ffmpeg*.

Interfaccia di comunicazione Il software fa largo utilizzo dello stack TC-P/IP per quasi tutte le sue funzionalità. In particolare utilizza i protocolli *http*, *udp*, *rtsp*.

Vincoli relativi all'occupazione di memoria Il *Server* deve disporre di sufficiente spazio di archiviazione per la memorizzazione dei dati temporali dei vari dispositivi (ad esempio: registrazioni video, registrazioni audio, andamento

dei sensori etc) per il periodo di mantenimento indicato dall'amministratore. Per completezza è disponibile anche l'archiviazione su server esterno via rete internet, anche se il suo utilizzo è sconsigliato per enti e privati che non dispongono di una banda larghissima in upload, la quale è necessaria per il trasporto di stream di grandi dimensioni come audio e video.

Per aggregazioni di piccole dimensioni (il numero effettivo dei dispositivi audiovisivi dipende dal numero di elaborazioni intermedie e la risoluzione dei singoli dispositivi, in generale 15 è un buon compromesso) non è necessario preoccuparsi della dimensione della memoria centrale (circa 4GB di memoria sono sufficienti). In caso di aggregazioni grandi è necessario scegliere con cura la quantità di memoria centrale (più 4GB di memoria).

Operazioni Al primo avvio è necessario inizializzare la password di amministratore, la quale verrà chiesta dal programma attraverso la riga di comando. Non sono previste operazioni di backup e recovery.

Vincoli per installazione È necessario che il server sia protetto da accessi indesiderati (sia per quanto riguarda la riga di comando sia per quanto riguarda l'accesso al database e al filesystem con le registrazioni).

2.2 Macro funzionalità

L'applicazione dovrà disporre di due interfacce separate, una accessibile tramite *http* e una via riga di comando.

All'interfaccia http avranno accesso gli utenti abilitati a visionare in *real time* lo stato dei vari dispositivi, nonché accedere alle registrazioni. Ogni utente deve essere prima autenticato con apposite credenziali per poterne identificare i permessi di visualizzazione e gestione della gerarchia.

L'interfaccia via riga di comando è riservata al solo amministratore del sistema. Anch'essa richiede l'inserimento di una password d'amministratore per evitare eventuali manomissioni. L'amministratore ha tutti i permessi di gestione e inoltre è l'unico utente che può gestire l'aggiunta, modifica ed eliminazione di utenti.

Gli utenti con i giusti permessi di gestione della gerarchia possono operare spostamenti, rinomine, cancellazioni e aggiunte di dispositivi e gruppi attraverso l'interfaccia web.

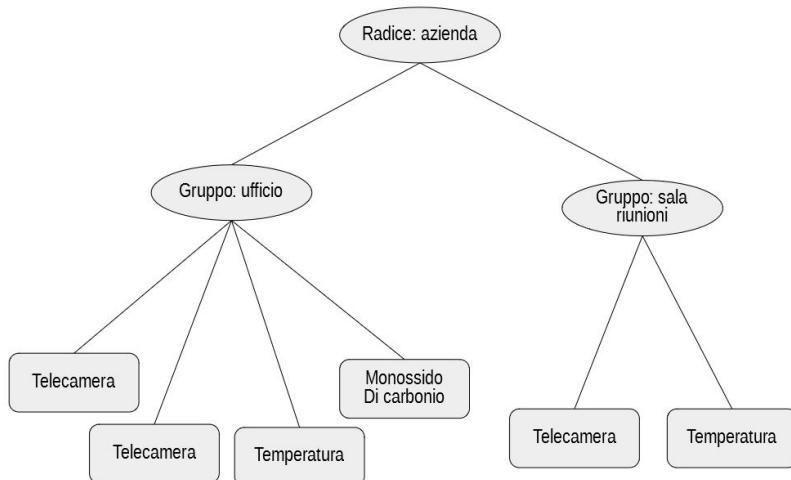
L'amministratore può effettuare tali operazioni anche tramite la riga di comando.

La gerarchia dei dispositivi è gestita sotto forma di albero, ovvero ogni dispositivo può essere aggregato in un gruppo che può a sua volta appartenere ad un'altro gruppo. Non sono ammesse relazioni ricorsive tra gruppi. Tutti i gruppi/dispositivi sono figli (diretti o indiretti) di un gruppo radice. In figura 1 si può osservare un esempio di organizzazione gerarchica di dispositivi e gruppi.

Per ogni gruppo è sufficiente memorizzarne solamente il nome e una descrizione.

Per ogni dispositivo invece è necessario memorizzarne il nome, una descrizione, il percorso dove salvare le registrazioni e il periodo di salvataggio. Inoltre può essere specificata anche una operazione intermedia, che per dispositivi audio-video può essere una operazione di transcodifica (utilizzando quindi ffmpeg) da un formato ad un altro oppure, nel caso di sensori con risultati numerici, una formula matematica (ad esempio il sensore restituisce i gradi su scala Celsius, ma l'utente vuole memorizzare i gradi in scala Kelvin).

Figura 1: Esempio di una possibile gerarchia di dispositivi (nei rettangoli) e gruppi (negli ovali)



2.3 Caratteristiche degli utenti

Gli utenti che si interfacciano con l'applicazione sono l'amministratore e gli utenti generali. L'amministratore ha completo accesso su tutte le parti della applicazione (sia nell'interfaccia web, sia nell'interfaccia a riga di comando). Gli utenti generali avranno dei determinati permessi a loro assegnati per l'utilizzo dell'interfaccia web.

L'utente amministratore deve avere un ottimo grado di dimestichezza con la macchina *Server* su cui configura l'applicazione, sia dal punto di vista del sistema operativo, sia in campo di reti e telecomunicazione, sia nell'ottica di configurazione di programmi via riga di comando, sia, infine, in campo di formati mul-

timediali audio-video. È sufficiente un diploma di tecnico/perito informatico; è consigliata una laurea in campo informatico.

Gli utenti generali sono, invece, di tipo eterogeneo. È comunque necessario che essi sappiano utilizzare un *personal computer* nelle sue funzioni ad altissimo livello (nel senso di comprendere le interfacce grafiche e la loro navigazione) dimodoché siano in grado di utilizzare un *browser web* per l'accesso all'applicazione. Per essi non sono necessarie competenze avanzate in campo di informatica, ma sono consigliate.

2.4 Vincoli e limiti

L'applicazione è limitata e/o vincolata nel suo funzionamento da:

- La memoria massima di archiviazione dei dati.
- Il numero di thread o core del processore e/o la potenza di calcolo della GPU.
- Le funzioni e i metodi che *ffmpeg* mette a disposizione.
- La banda massima supportata dalla connessione ad internet, nonché la banda massima della rete locale.
- Le politiche di privacy necessarie per memorizzare dati non sensibili di utenti (e-mail, username e password).
- Le politiche di privacy necessarie per memorizzare dati audiovisivi di persone.
- La presenza sul server di una *Java Virtual Machine*.
- La presenza sul client di un *browser web*

2.5 Presupposti e dipendenze

Il progetto dipende dai progetti *CouchDB* e *ffmpeg*, utilizza il protocollo ip e i protocolli che si appoggiano ad esso (http, udp, rtsp etc) e supporta solo macchine *x86* e *amd64* con sistema operativo Microsoft Windows, Linux, macOS. I browser supportati dall'interfaccia web sono Microsoft Edge, Apple Safari, Mozilla Firefox, Google Chrome (e suoi derivati, es: Chromium, Opera etc). L'applicazione risente inoltre delle normative di privacy internazionali. Queste dipendenze possono portare al cambiamento del presente documento.

2.6 Requisiti da analizzare in futuro

In futuro può essere necessario supportare più protocolli e più dispositivi. Può essere utile implementare una comunicazione tra l'applicazione e sistemi di *smart home* come Amazon Echo.

3 Specifica dei requisiti

3.1 Requisiti dell'interfaccia esterna

Vengono riportate le principali interfacce di input/output, utilizzando un formato simile a quanto verrà utilizzato per i requisiti funzionali e non. (EI xx = External Interface numero xx)

EI01	Database	Gerarchia dispositivi
Descrizione	L'applicazione riceve in formato JSON la lista di dispositivi, gruppi e la loro organizzazione gerarchica.	
Formato dati	JSON	
Tipo	Input	

EI02	Dispositivi	Stream dati
Descrizione	L'applicazione riceve dai dispositivi attivi uno stream di dati.	
Formato dati	Il formato dati dipende dal protocollo utilizzato dal dispositivo	
Tipo	Input	

EI03	Server	Interfaccia web
Descrizione	L'applicazione invia ai client le pagine HTML tramite http.	
Formato dati	HTML5 [4], via http	
Tipo	Output	

EI04	Client	Interfaccia web
Descrizione	L'applicazione riceve dai client le richieste tramite http.	
Formato dati	JSON, via http	
Tipo	Input	

EI05	Server	Riga di comando
Descrizione	L'applicazione riceve i comandi da riga di comando	
Formato dati	ASCII	
Tipo	Input	

EI06	Server	Archiviazione dati
Descrizione	L'applicazione salva i dati nell'archivio	
Formato dati	Dati binari	
Tipo	Output	

3.2 Requisiti funzionali

RF01	Gestione elementi	Aggiunta
Input	Nome iniziale dell'elemento	
Processo	<p>Da web: l'utente clicca sul bottone “aggiungi”, un popup chiede il nome iniziale, l'utente sceglie se annullare l'operazione o confermarla.</p> <p>Da riga di comando: l'utente scrive il comando apposito per aggiungere un elemento.</p>	
Output	L'elemento viene memorizzato sul database e viene visualizzato sulla interfaccia web o con il comando di elenco degli elementi.	

RF02	Gestione elementi	Modifica
Input	Dati aggiornati dell'elemento	
Processo	<p>Da web: l'utente clicca con il tasto destro sull'elemento da modificare e sceglie “modifica”, un popup chiede i dati, l'utente sceglie se annullare l'operazione o confermarla.</p> <p>Da riga di comando: l'utente scrive il comando apposito per modificare un elemento.</p>	
Output	L'elemento viene aggiornato sul database e viene aggiornato sulla interfaccia web e con il comando di elenco degli elementi.	

RF03	Gestione elementi	Eliminazione
Input	Elemento da eliminare	
Processo	<p>Da web: l'utente clicca con il tasto destro l'elemento da eliminare e sceglie “elimina”; un popup chiede conferma.</p> <p>Da riga di comando: l'utente scrive il comando apposito per eliminare un elemento.</p>	
Output	L'elemento viene eliminato sul database e le sue registrazioni vengono eliminate	

RF04	Gestione elementi	Assegnazione gruppo
Input	Elemento figlio e gruppo padre	
Processo	<p>Da web: l'utente trascina l'<i>handle</i> presente sopra l'elemento figlio e lo rilascia sul gruppo padre.</p> <p>Da riga di comando: l'utente scrive il comando apposito per assegnare un elemento figlio ad un gruppo padre.</p>	
Output	L'elemento padre dell'elemento figlio viene aggiornato e si aggiorna anche la vista sul web.	

RF05	Gestione Elementi	Visualizzazione
Input	Dispositivo da visualizzare	
Processo	Da web: l'utente clicca due volte sull'elemento da visualizzare.	
Output	I dati vengono visualizzati.	

RF06	Utenti	Login
Input	Username e password	
Processo	Da web: l'utente clicca sul bottone "login", un popup chiede username e password, l'utente sceglie se annullare l'operazione o confermarla.	
Output	Se username e password corretti l'utente accede al sistema.	

RF07	Utenti	Aggiunta utente
Input	Username	
Processo	Da riga di comando: l'amministratore aggiunge un utente con un determinato username utilizzando il comando appropriato.	
Output	L'utente viene aggiunto al sistema con una password temporanea che viene mostrata a video all'amministratore.	

3.3 Requisiti non funzionali

RN01	Prestazioni	Tempo di risposta
Descrizione	L'interfaccia web deve rispondere in al massimo 5 secondi nell'80% dei casi. Nel caso la connessione sia lenta deve mostrarne l'avanzamento.	

RN02	Sicurezza	Protezione dati
Descrizione	L'applicazione deve garantire la sicurezza dei dati che gestisce secondo normative internazionali, con particolare riferimento al GDPR [5]	

RN03	User experience	Interfaccia grafica
Descrizione	L'applicazione deve offrire l'interfaccia web in modo elegante e comprensibile anche da parte degli utenti meno esperti	

RN04	User experience	Guida
Descrizione	L'applicazione deve mettere a disposizione una guida nella interfaccia grafica in modo da facilitarne l'uso da parte degli utenti meno esperti	

RN05	Ecosistema	Scalabilità
Descrizione	L'applicazione deve essere in grado di sostenere un numero di elementi sempre crescente e gestire correttamente eventuali upgrade all'hardware del Server	

4 Appendice

4.1 Tabelle dimensione stream registrati

Di seguito vengono riportate alcune tabelle utili per stimare la grandezza totale su disco delle registrazioni dei dispositivi. Si rammenta che le misure sono a solo scopo illustrativo e dipendono da più fattori incalcolabili a priori; inoltre la lista di formati non è da considerarsi esaustiva. Nel caso di dati audiovisivi, è necessario sommare la dimensione al minuto dei rispettivi formati.

Audio		
Wav	PCM 16 bit @44.1kHz stereo	10 MB/min
Flac	@44.1kHz stereo	5 MB/min
mp3	@128kbps, 44.1kHz stereo	1 MB/min
ogg	@128kbps, 44.1kHz stereo	1 MB/min

Video	
h.264	1920x1080
h.264	1280x720
h.264	720x486

Dati	
1 sample al millisecondo, 32 bit floating point	240 kB/min
1 sample al secondo, 32 bit floating point	240 B/min
1 sample al minuto, 32 bit floating point	1 B/min
1 sample all'ora, 32 bit floating point	1 B/ora

4.2 Esempio di sintassi da riga di comando

Aggiunta di un dispositivo generico:

device add <name>

dove *name* identifica il nome del nuovo dispositivo da aggiungere.

Modifica campi di un dispositivo generico:

device set <field> <value>

dove *field* identifica il nome del campo da modificare e *value* il valore da assegnare al campo.

Eliminazione di un dispositivo generico:

device delete <name>

dove *name* identifica il nome del dispositivo da eliminare.

Aggiunta, modifica, eliminazione di un gruppo:

group add <name>

group set <field> <value>

group delete <name>

Aggiunta di un figlio ad un gruppo:

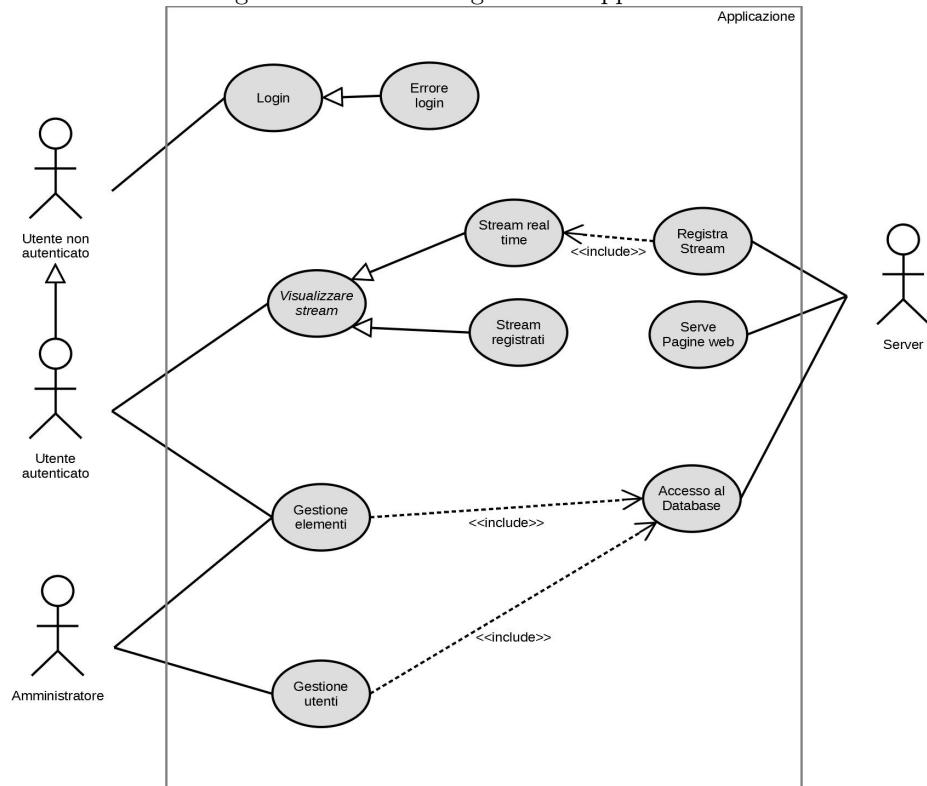
group addchild <parentgroup> <name>

dove *parentgroup* è il nome del gruppo a cui assegnare l'elemento con nome *name*.

4.3 Design

4.3.1 Use Case

Figura 2: Use case diagram dell'applicazione

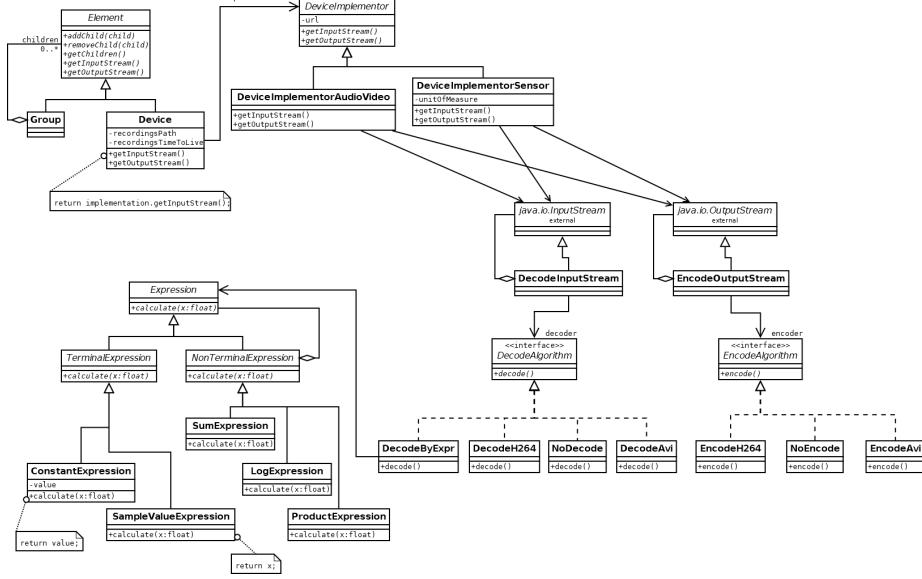


4.3.2 Design pattern

In figura 3 si può osservare una vista di insieme delle relazioni tra le classi di una parte della applicazione. Il diagramma non è da considerarsi esaustivo (ad esempio non sono riportati tutti gli algoritmi di codifica/tutte le espressioni).

In seguito verranno, poi, analizzate parti del diagramma per identificare e spiegare i vari pattern utilizzati.

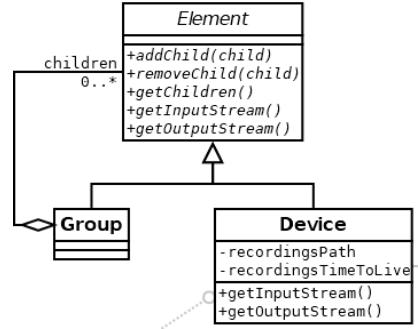
Figura 3: Class diagram non esaustivo di una parte di applicazione



Composite La gerarchia dei gruppi e dei dispositivi è modellata utilizzando il design pattern “Composite”, mostrato in figura 4. Esso prevede che gli elementi della gerarchia vengano trattati allo stesso modo e li rappresenta sotto forma di albero, in linea con quanto richiesto nei requisiti. Si è deciso di tenere l’implementazione dei dispositivi separata dalla gerarchia utilizzando un “Bridge pattern”, la cui discussione è oggetto del paragrafo seguente.

Bridge Per evitare che la gerarchia dei dispositivi cresca in modo incontrollabile e per promuovere la composizione sull’ereditarietà, si è deciso di utilizzare il design pattern “Bridge” (in figura 5) per separare l’astrazione del dispositivo dalla sua effettiva implementazione. Nel diagramma vengono mostrate due possibili implementazioni del dispositivo, ovvero il “sensore” e il dispositivo “audiovisivo”.

Figura 4: Composite



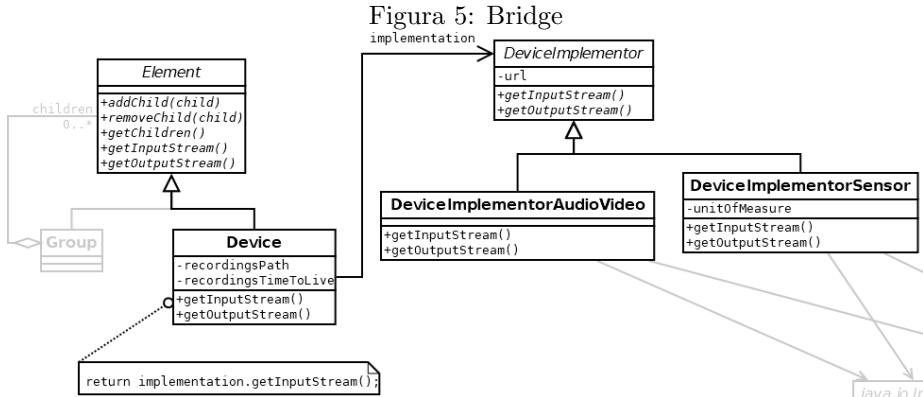


Figura 5: Bridge

implementation

Decorator Per gestire in modo flessibile le operazioni di codifica e decodifica degli stream dei dispositivi si è deciso di implementare il design pattern del “Decorator” (figura 6) in quanto è anche il pattern utilizzato da Java per aggiungere funzionalità agli stream. Per questo stesso motivo il decorator viene utilizzato in giunzione al pattern “Strategy” (la discussione è nel paragrafo seguente), in modo da evitare di dover creare tante classi derivate dai decorator rischiando di dover ripetere del codice; ancora una volta si opta per la composizione invece che l’ereditarietà.

Strategy In figura 7 viene esposto l’utilizzo dello strategy pattern nella definizione degli algoritmi di decodifica (per la codifica si applica lo stesso pattern con le stesse considerazioni). Si noti l’esistenza di un algoritmo chiamato “NoDecode”, il quale semplicemente restituisce lo stream non decodificato, ovvero non ne modifica i byte. Particolare attenzione è da riservare all’algoritmo “DecodeByExpr”, il quale è utile per modellare la richiesta di applicare delle funzioni ai valori dei sensori; esso lavora in sinergia con il pattern “Interpreter” che viene analizzato nel prossimo paragrafo.

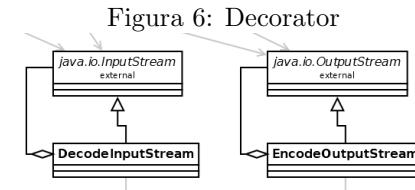


Figura 6: Decorator

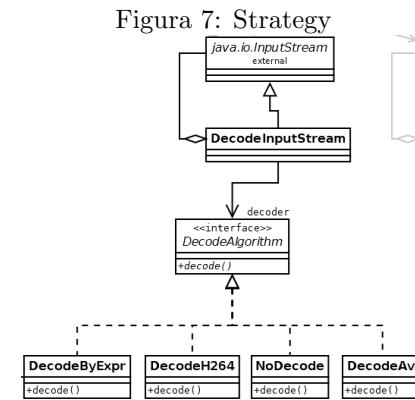


Figura 7: Strategy

Interpreter Come anticipato nel paragrafo precedente, l'applicazione di una funzione al valore di un sensore è implementata tramite il design pattern “Interpreter”. Esso modella una grammatica tramite un albero e permette, quindi, di costruire una espressione matematica in modo ricorsivo. Nel diagramma di figura 8 vengono mostrate alcune operazioni e alcuni operandi d'esempio come la somma, il prodotto, il logaritmo, la costante e il valore del sample stesso (che viene passato come “Contesto” del calcolo).

Di seguito viene mostrato un possibile object diagram e un sequence diagram per la risoluzione di una espressione da parte dell'interprete; inoltre viene proposto in pseudo codice Java una possibile implementazione di alcune classi dello stesso.

Figura 9: Object diagram di una espressione d'esempio

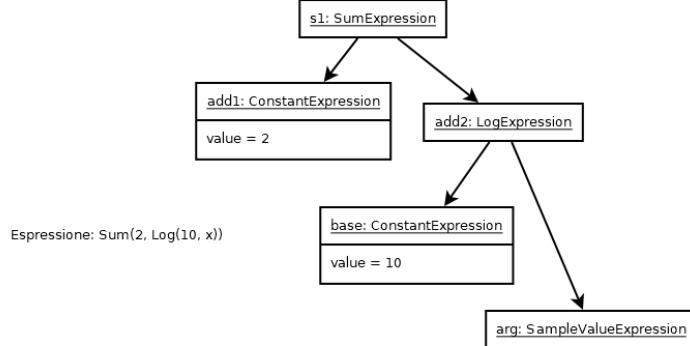


Figura 8: Interpreter

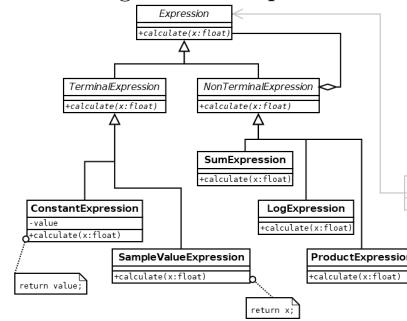
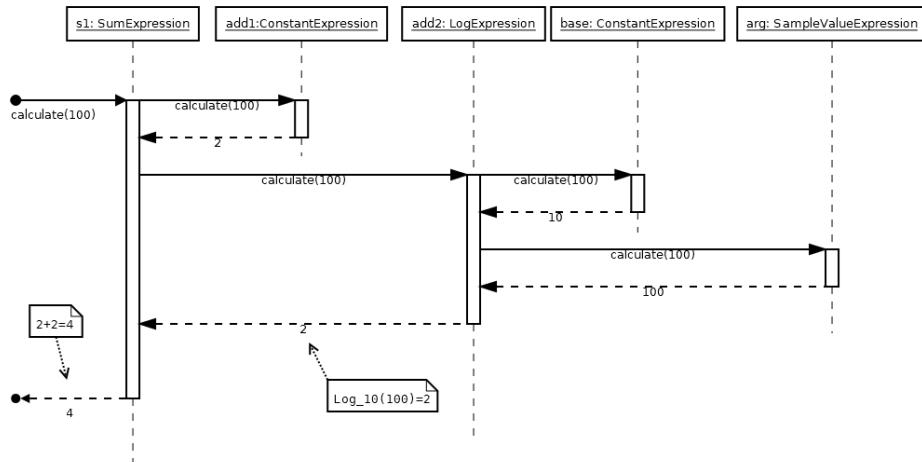


Figura 10: Sequence diagram della risoluzione dell'espressione di figura 9 con valore di $x = 100$



```

public abstract class Expression {
    public abstract float calculate(float x);
}

public final class ConstantExpression extends Expression {
    private final float value;
    public ConstantExpression(float value) {
        this.value = value;
    }
    public final float calculate(float x) {
        return this.value;
    }
}

public final class SampleValueExpression extends Expression {
    public final float calculate(float x) {
        return x;
    }
}

public final class SumExpression extends Expression {
    private final Expression add1;
    private final Expression add2;
    public SumExpression(Expression add1, Expression add2) {
        this.add1 = add1;
        this.add2 = add2;
    }
}
  
```

```

        public final float calculate(float x){
            final float a = this.add1.calculate(x);
            final float b = this.add2.calculate(x);
            return a + b;
        }
    }

    public final class ProductExpression extends Expression {
        private final Expression fac1;
        private final Expression fac2;
        public ProductExpression(Expression fac1, Expression
fac2){
            this.fac1 = fac1;
            this.fac2 = fac2;
        }
        public final float calculate(float x){
            final float a = this.fac1.calculate(x);
            final float b = this.fac2.calculate(x);
            return a * b;
        }
    }

    public final class LogExpression extends Expression {
        private final Expression base;
        private final Expression arg;
        public LogExpression(Expression base, Expression arg){
            this.base = base;
            this.arg = arg;
        }
        public final float calculate(float x){
            final float a = this.arg.calculate(x);
            final float b = this.base.calculate(x);
            return (float)(Math.log(a) / Math.log(b));
        }
    }
}

```

Altri pattern implementabili I pattern sopra proposti sono un esempio dei tanti che si possono utilizzare per questa applicazione; ad esempio si potrebbe pensare di implementare la comunicazione coi dispositivi attraverso un “Proxy” in modo da gestire al meglio il traffico. Un altro pattern implementabile è il “Command” per gestire le operazioni di management della gerarchia. Non sono stati presentati pattern di creazione come “Abstract Factory”, “Singleton” etc, ma hanno anch’essi una applicazione in questo progetto.

4.3.3 Activity Diagram

Viene riportato un activity diagram relativo alla modifica di un elemento nella gerarchia da parte di un utente che sta utilizzando l’interfaccia web.

Figura 11: Activity diagram relativo alla modifica

