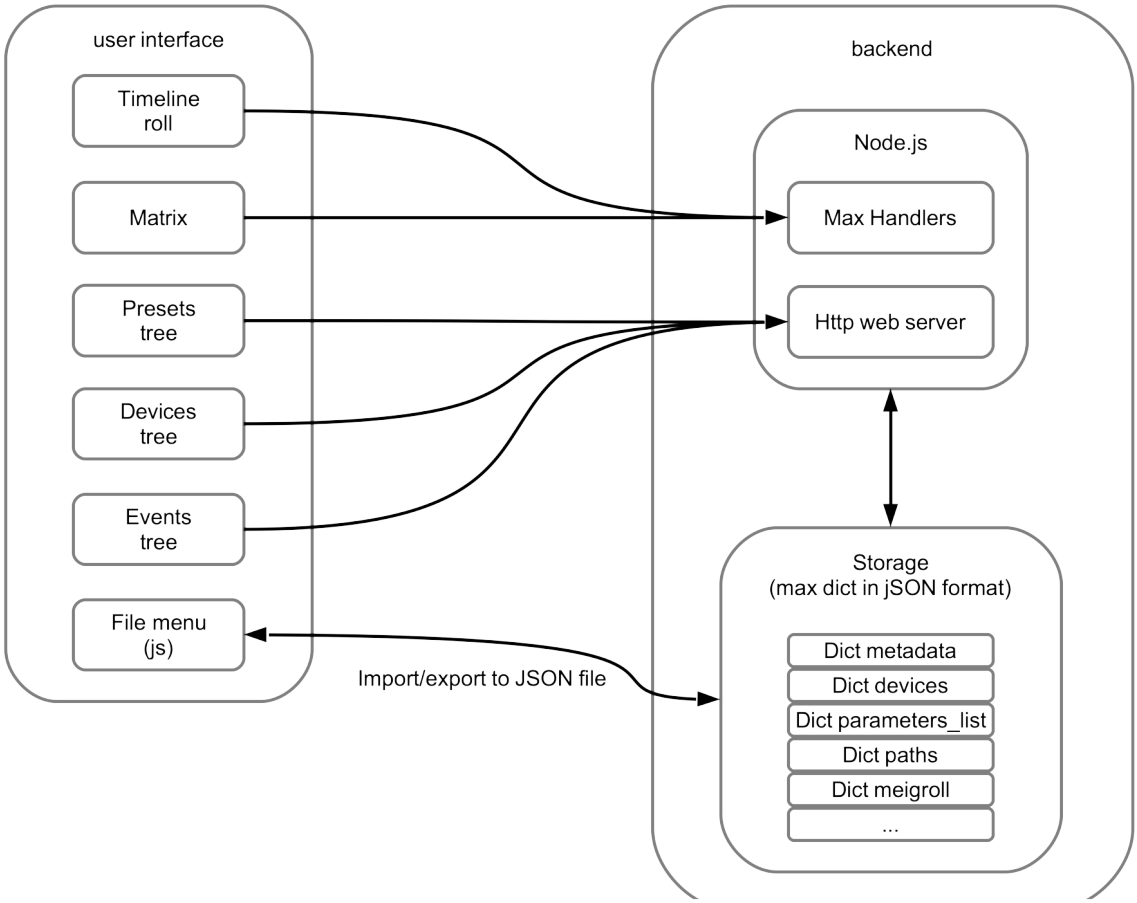
MEIG SYSTEM

Infrastruttura

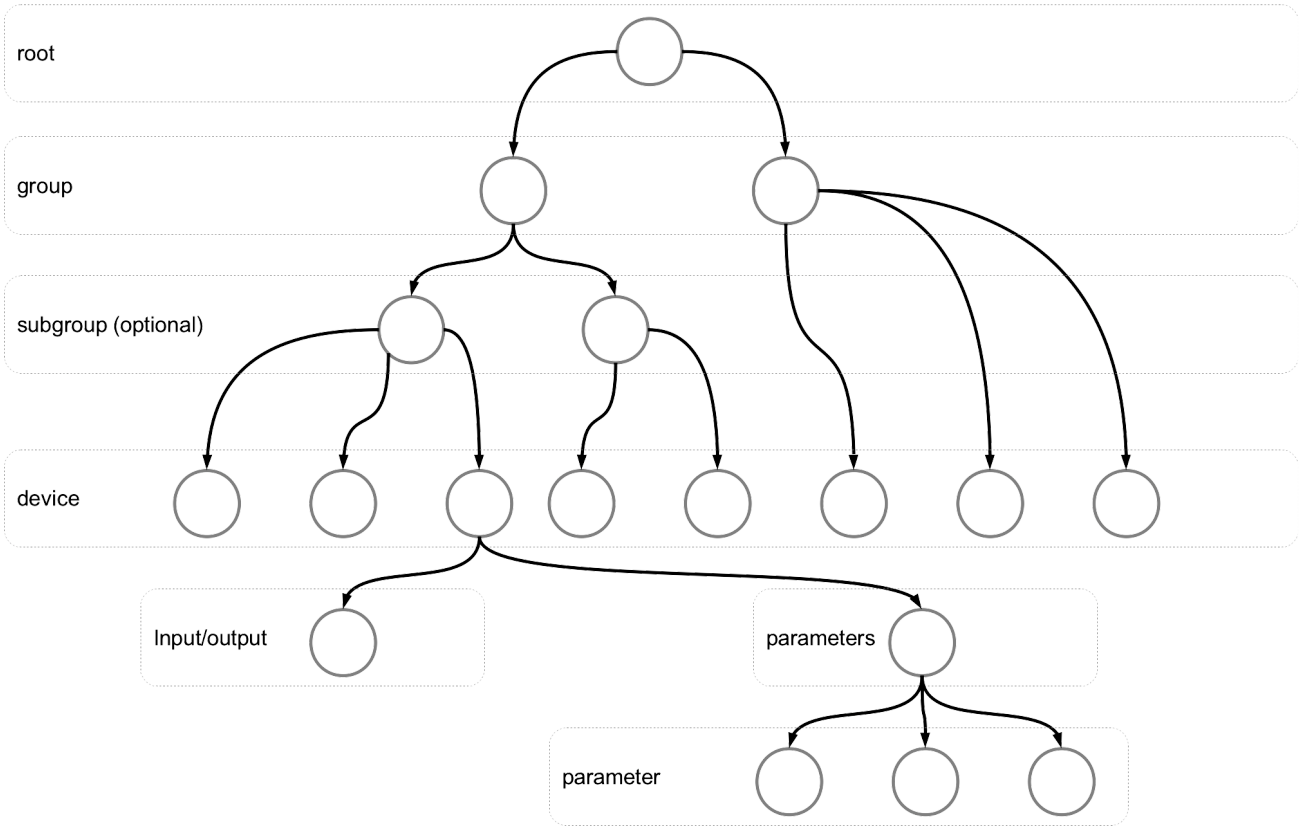
L'app ha una struttura ibrida: una parte delle componenti comunicano tramite il protocollo HTTP, trasmettendo dati in formato JSON; un'altra parte accede direttamente al backend tramite dei manipolatori *ad hoc* (Max handlers). Al centro dell'infrastruttura si trova un server implementato in node.js, che raccoglie le chiamate sia dai client http che dalle componenti ad accesso diretto. I dati vengono mantenuti localmente in oggetti Max di tipo **dict**. Si tratta *de facto* di strutture dati in formato JSON che possono essere aggregate al momento dell'esportazione su file, anch'essi in formato JSON.

In figura si può osservare la struttura generale dell'applicazione.



Per quanto riguarda la persistenza dei dati, è sembrato opportuno utilizzare una struttura dati ad albero, la cui rappresentazione più naturale fosse lo stesso JSON. Come si vede

I dati vengono rappresentati in strutture ad albero con radice. Ogni nodo deriva da uno e un solo genitore, mentre può avere nessuno, uno o più figli. La figura seguente mostra la struttura (parziale) dell'albero dei *devices*.



Ogni nodo è identificato da un *id* univoco nel sistema, quindi può essere modificato senza dover essere duplicato e nuovamente referenziato. L'albero viene rappresentato all'interno dei *dict* di Max in formato JSON: Ogni nodo ha almeno un *id* (l'unico elemento non modificabile), una *label* e una chiave *type*. Se non si tratta di una *foglia* ha la chiave **children**, che contiene un array dei nodi figli:

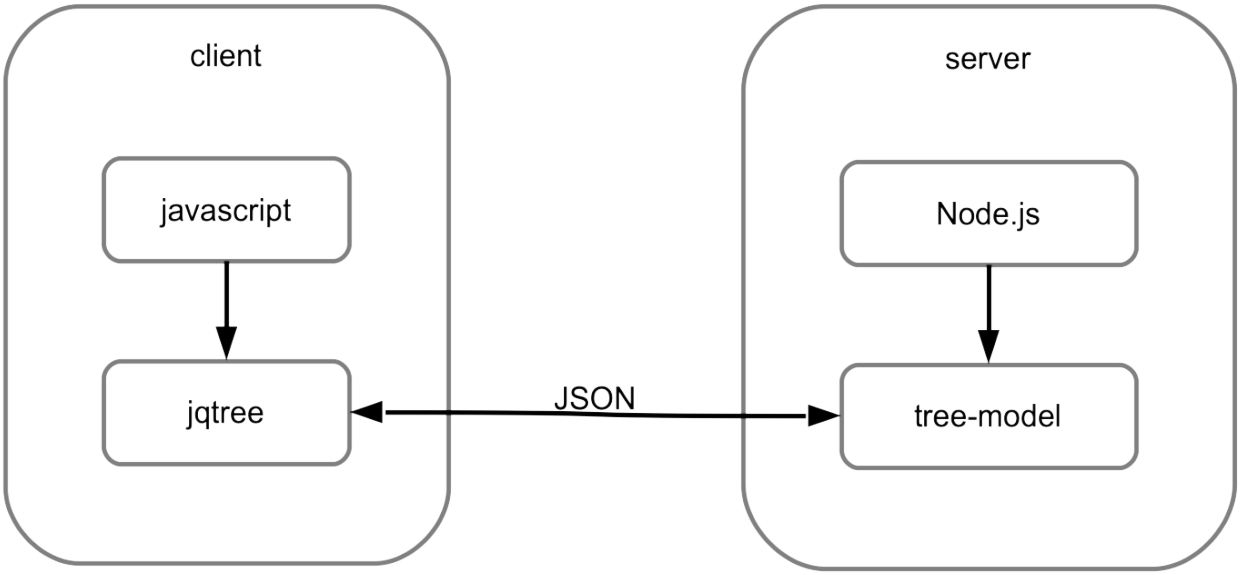
```
{
    "label": "main",
    "id": "1234abcd5678efgh",
    "type": "devices",
    "children": [
        {
            "label": "group_1",
            "id":"id": "0000abcd5678asdf",
            "type": "group",
            "children": [
                {"...":"..."}
            ]
        },
        {
            "label": "group_2",
            "id": "2222sdas12312yrue333",
            "type": "group",
            "children": [
                {"...":"..."}
            ]
        }
```

```
    ]
}
```

La struttura ricorsiva di ogni nodo permette di rendere alcuni *types* innestabili a livelli diversi. L'entità di tipo *group* ad esempio, può avere come *children* altre entità sinonime:

```
{
    "label": "group_1",
    "id":"id": "0000abcd5678asdf",
    "type": "group",
    "children": [
        {
            "label": "subgroup_1",
            "id":"id": "0000abcd5678asdf",
            "type": "group",
            "children": [
                {"...":"..."}
            ]
        }
    ]
}
```

Tali strutture dati vengono gestite sia lato client che lato server con *packages* che permettono di modellizzare le strutture dati ad albero; come si vede in figura il pacchetto javascript **jqtree** si occupa di gestire l'albero lato client, mentre il pachetto **tree-model** gestisce gli alberi lato server.

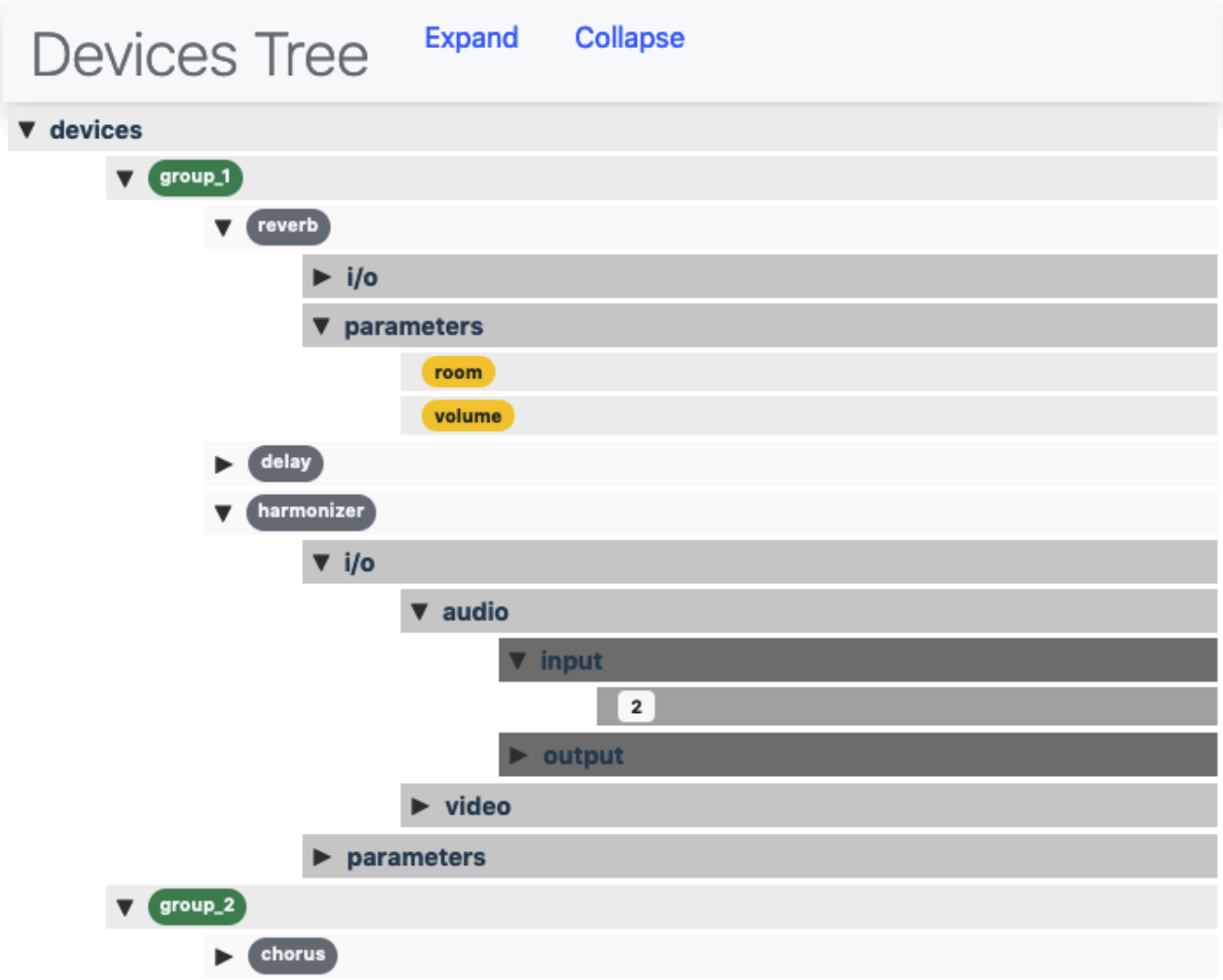


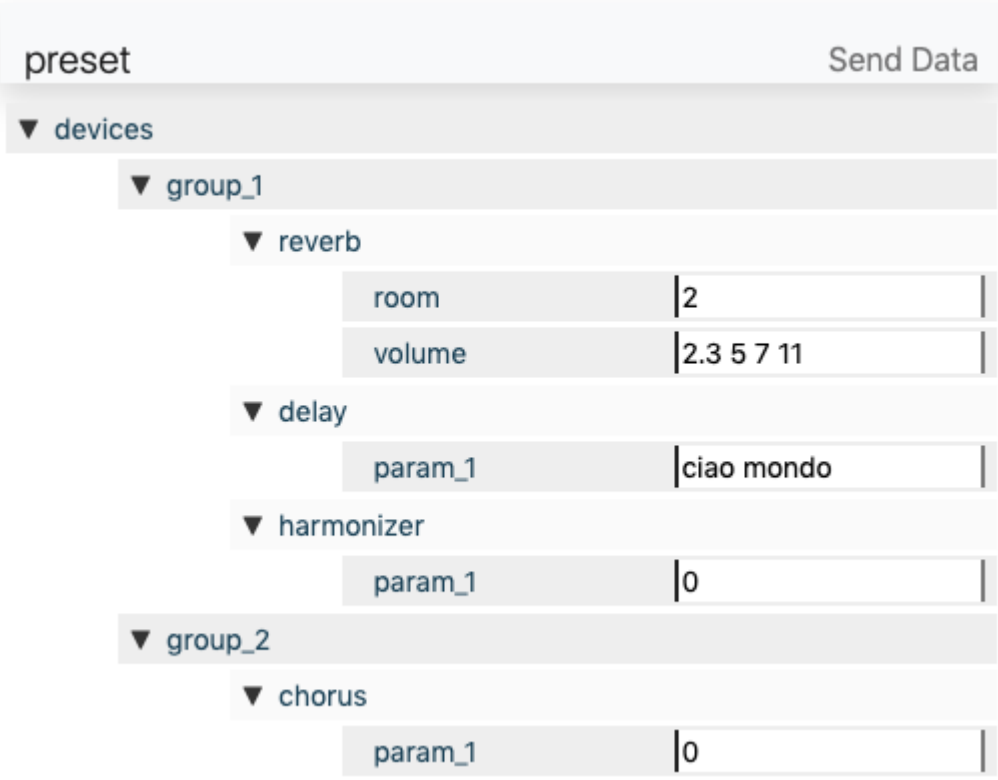
Graphical User Interfaces

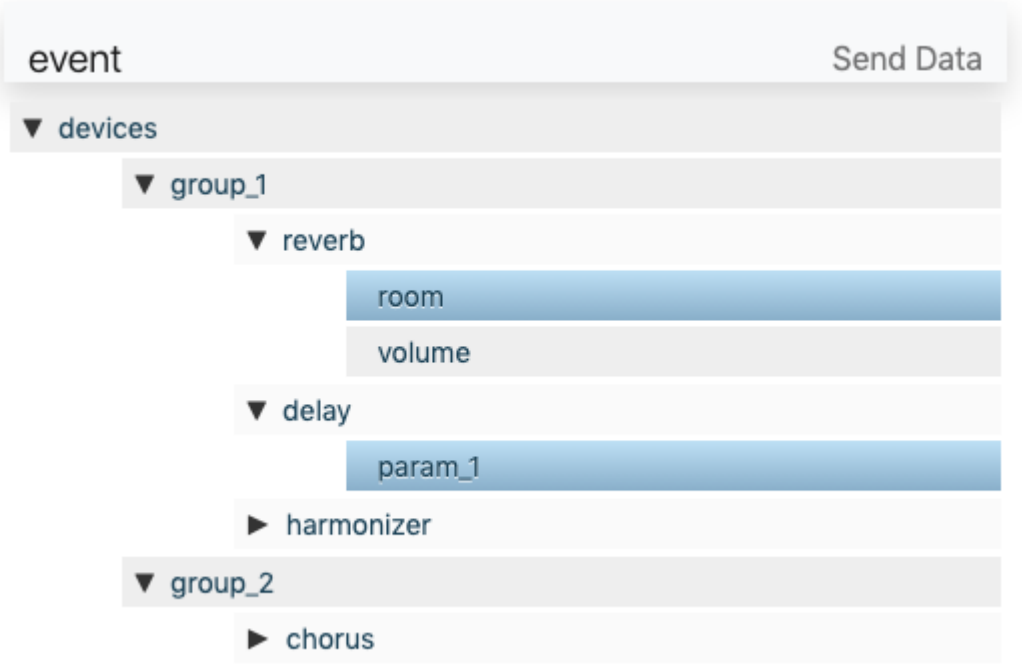
Devices, Preset, Event

Allo stato attuale del progetto, l'entità che rappresenta l'albero dei *Devices* ha una rappresentazione grafica realizzata in html/css/javascript e contenuta all'interno della *patch* di Max nell'oggetto **jweb**. A partire da questa entità vengono creati altri due alberi, *Presets* ed *Events*, con strutture grafiche analoghe. L'albero dei *Devices* viene servito all'utente all'atto dell'apertura di un file esistente o della creazione di un file nuovo e permette di inserire nel sistema nuovi dispositivi. L'entità centrale di questa struttura è il **device** che contiene gli input/output e i parametri, e che può essere a sua volta contenuto in gruppi o sottogruppi. Per aggiungere, rimuovere o rinominare un'entità dall'albero è sufficiente usare i menu contestuali attivabili col destro del mouse.

Dall'albero dei *Devices* derivano le altre due strutture, *Preset* ed *Event*, che permettono, rispettivamente, di impostare i valori di tutti i parametri in un preset e di scegliere quali parametri utilizzare all'interno di un event. Graficamente il *Preset Tree* è composto da una vista ad albero con tutti i parametri provvisti di un *form* in cui inserire il valore (*Number*, *String* o *Array*). L'*Event Tree* invece è un mero elenco di tutti i parametri, con la possiblità di selezionare quelli interessati.


Devices Tree
Expand
Collapse
devices
group_1
reverb
i/o
parameters
room
volume
delay
harmonizer
i/o
audio
input
2
output
video
parameters
group_2
chorus


preset
Send Data
devices
group_1
reverb
room
2
volume
2.3 5 7 11
delay
param_1
ciao mondo
harmonizer
param_1
0
group_2
chorus
param_1
0


event
Send Data
▼ devices
▼ group_1
▼ reverb
room
volume
▼ delay
param_1
▶ harmonizer
▼ group_2
▶ chorus