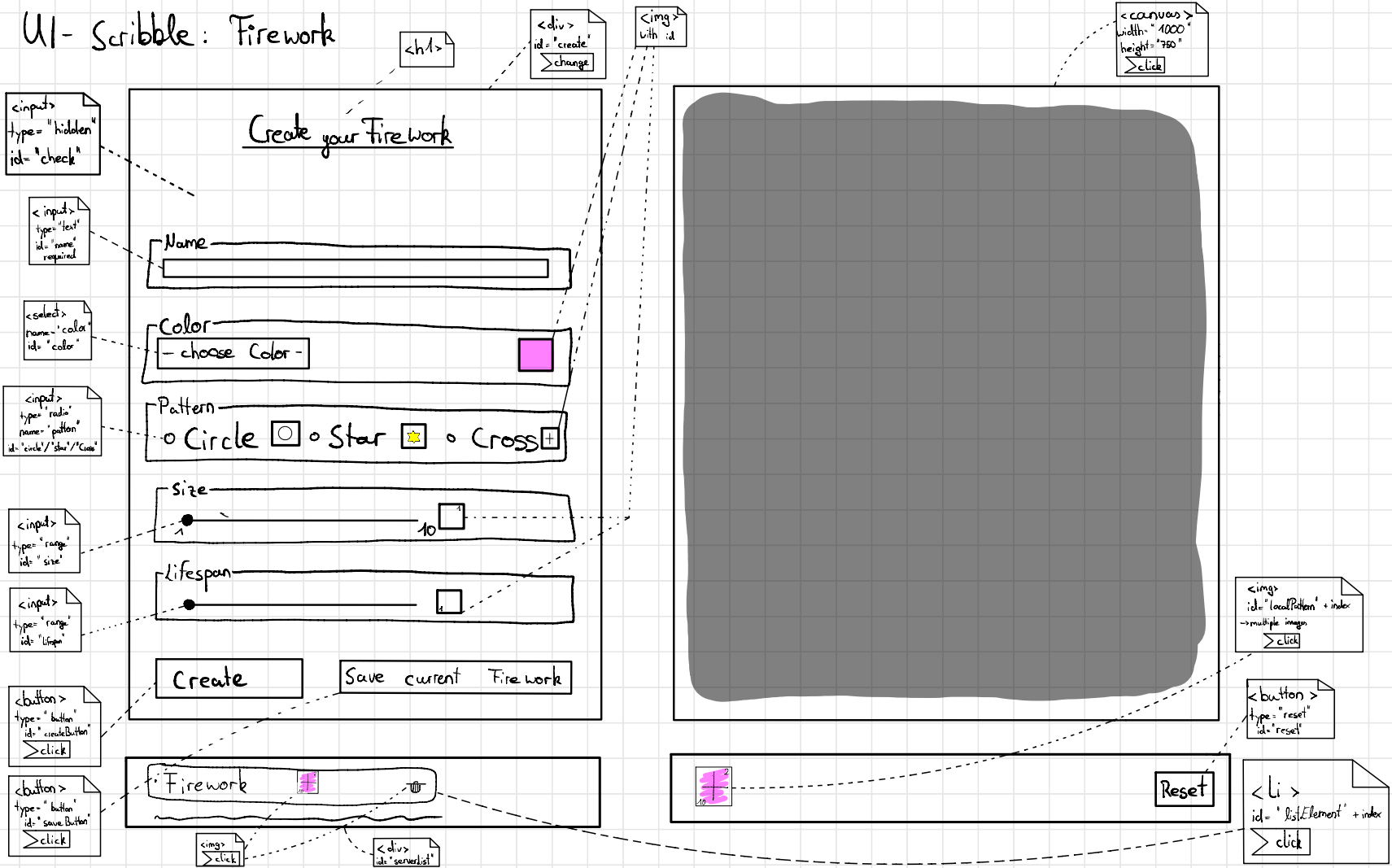
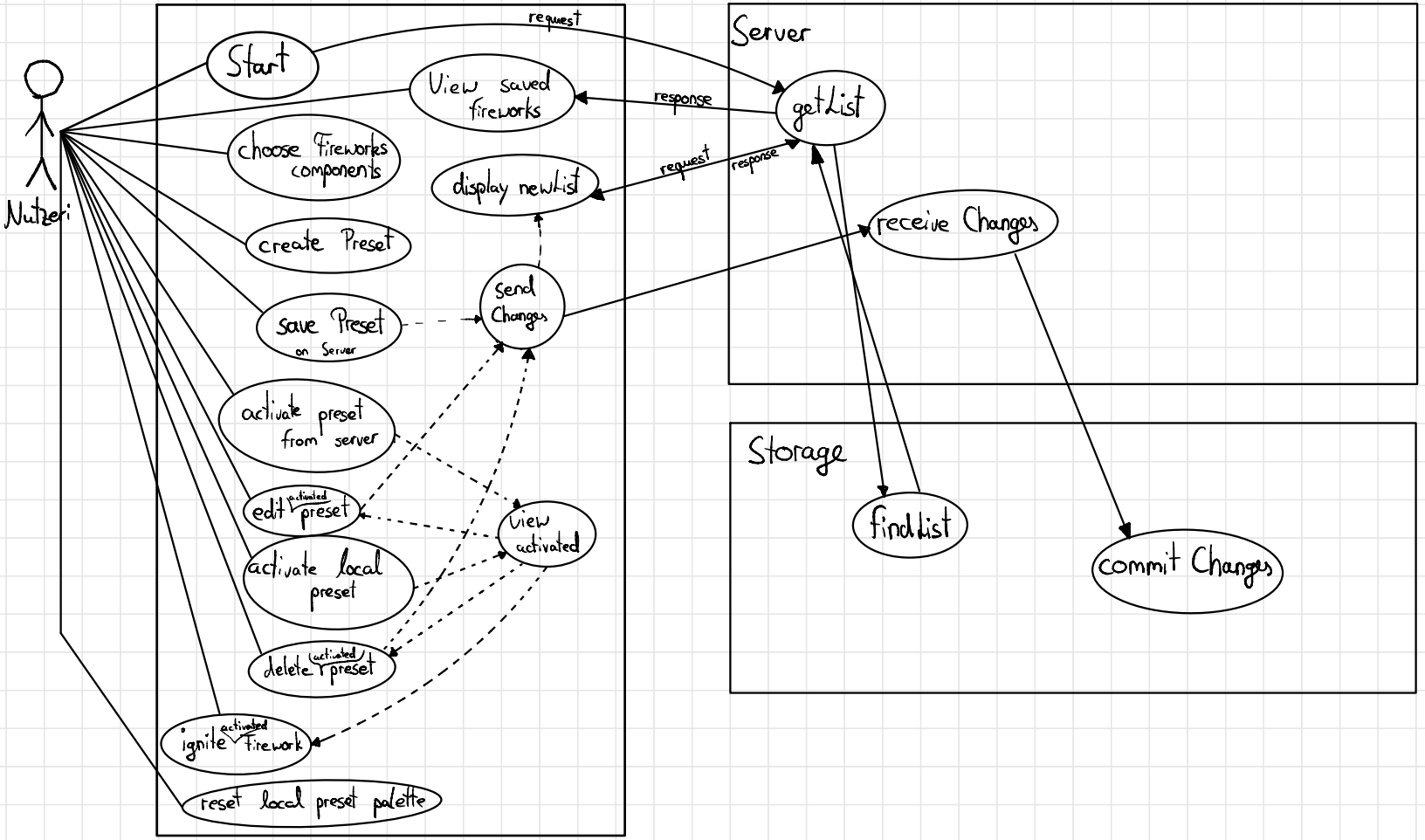
- Farbe ⟶ droop Down
- Größe der Partikel ⟶ Slider
- Muster ⟶ Radio - Button
- Lebensdauer der Partikel -> Slider

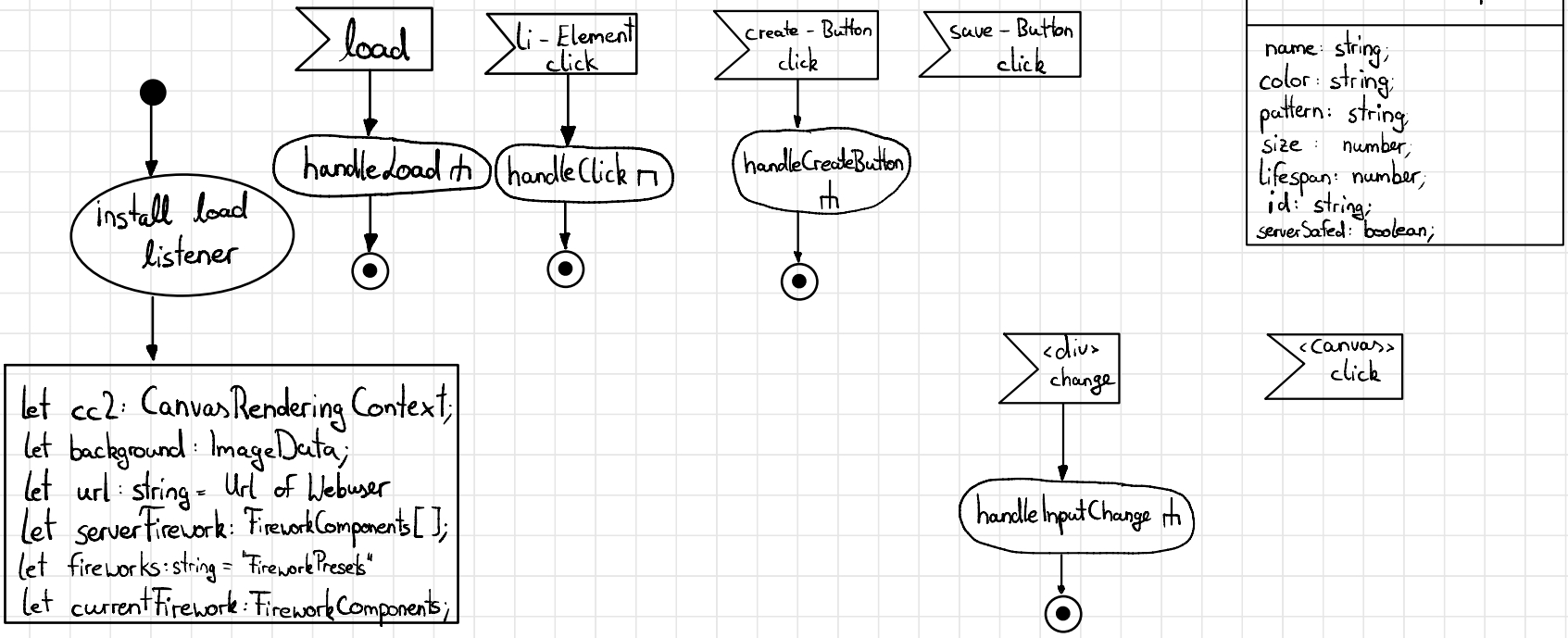
UI- Scribble: Firework



Use - Case - Diagram

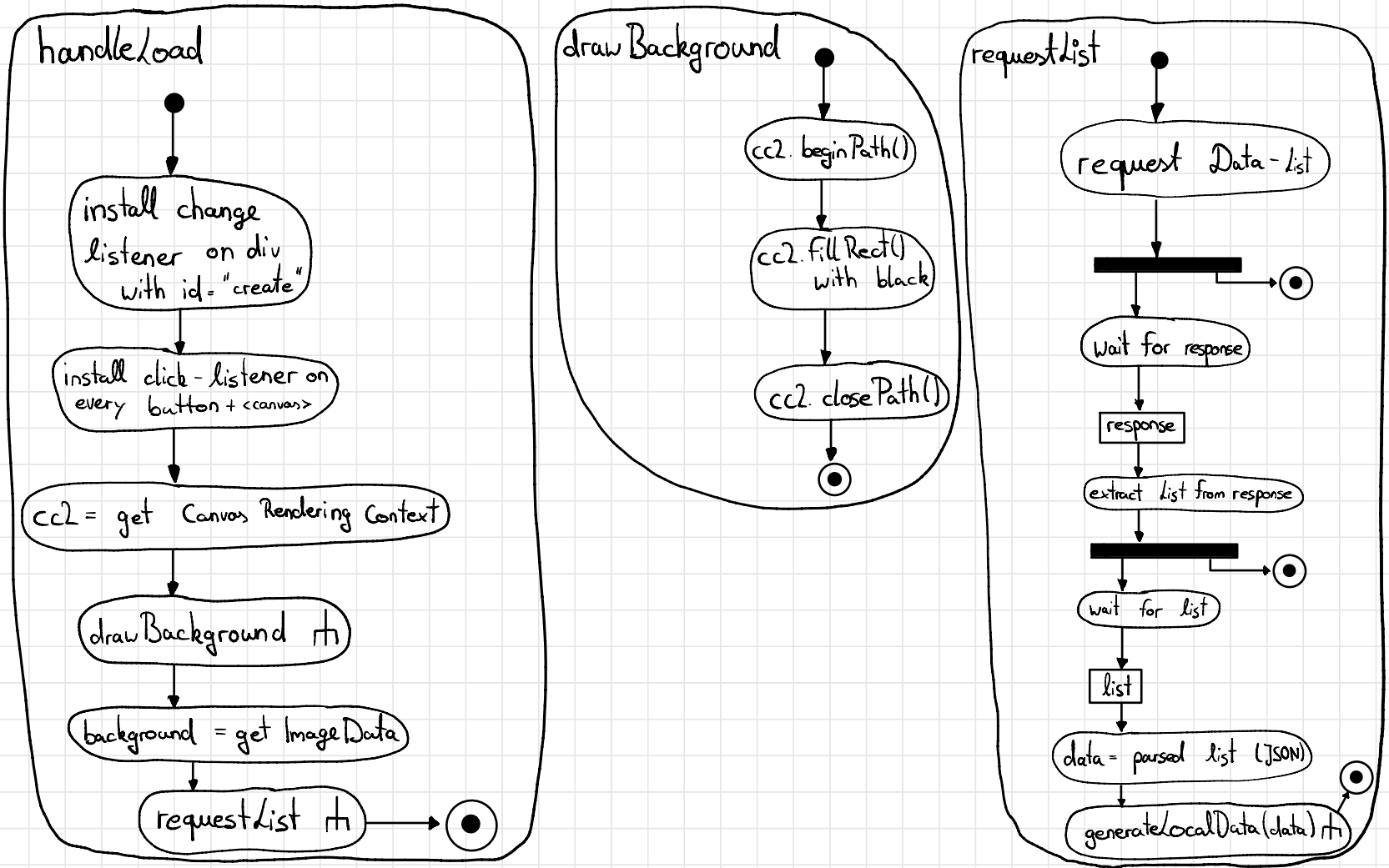


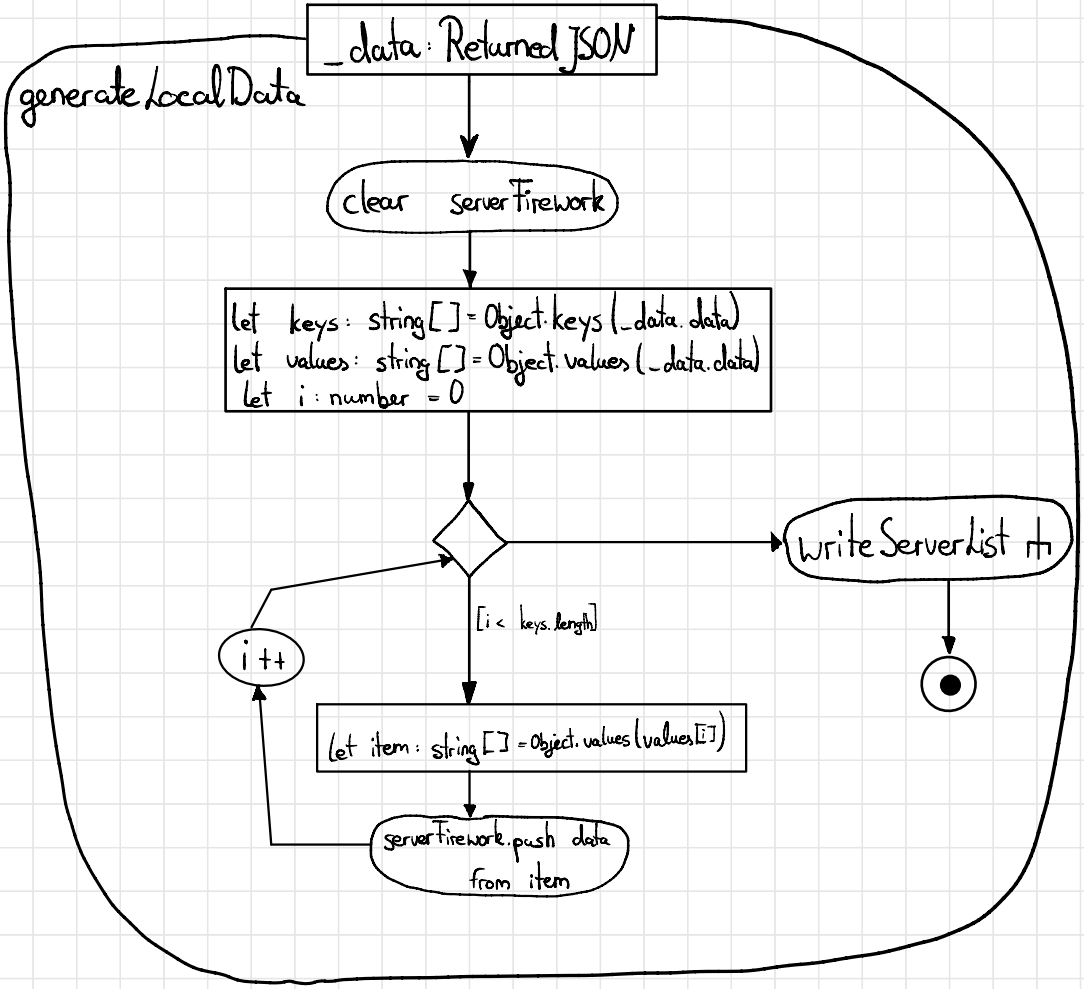
Activity Diagram: Firework

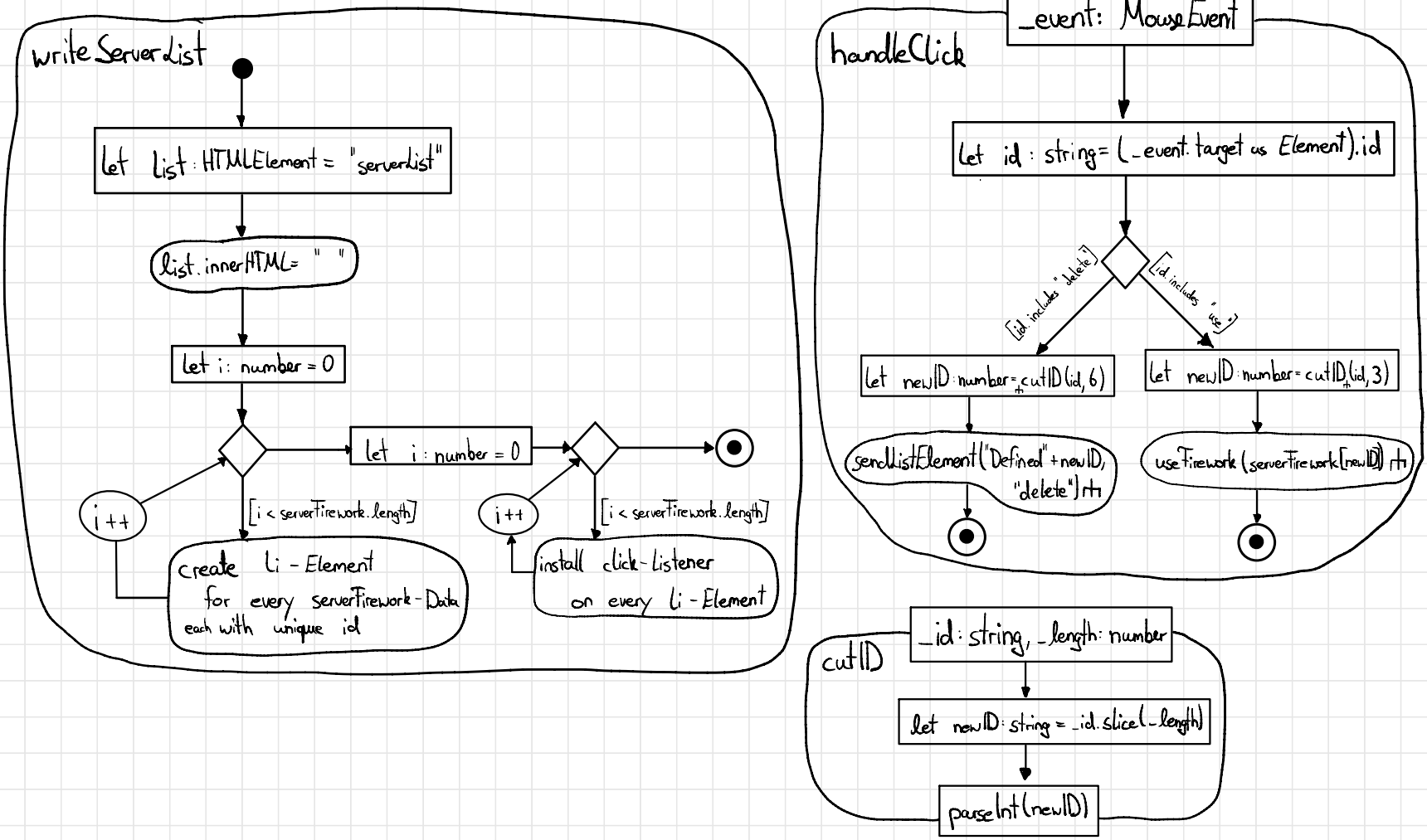


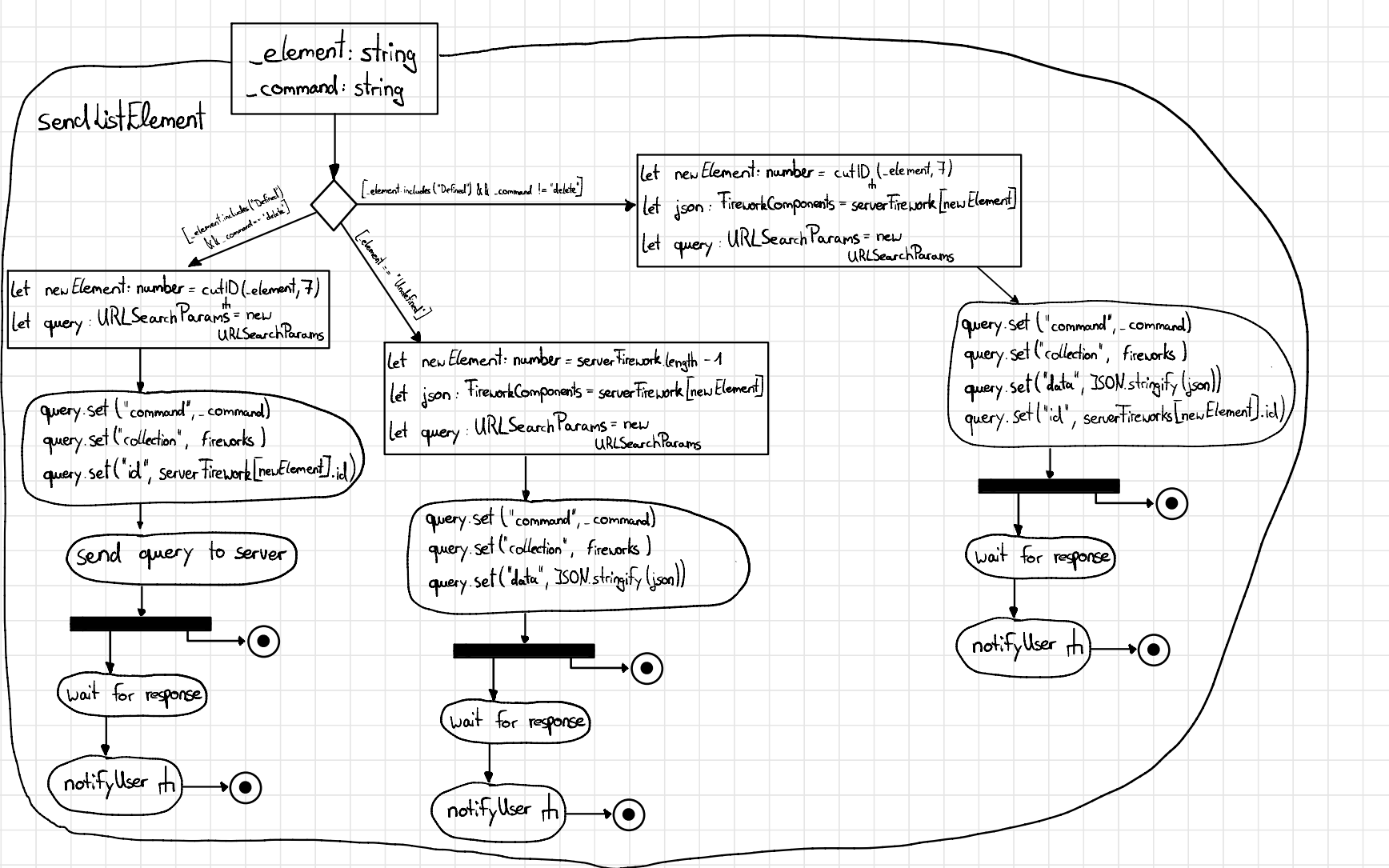
Interface: FireworkComponents

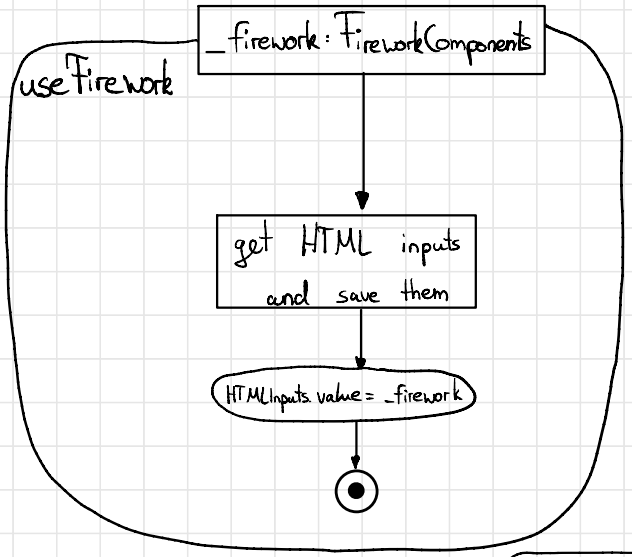
```
name: string;
color: string;
pattern: string;
size: number;
lifespan: number;
id: string;
serverSafed: boolean;
```

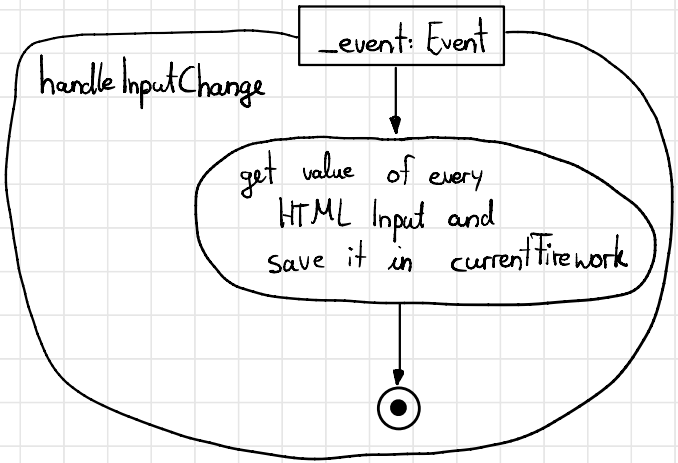

handleLoad
install change listener on div with id = "create"
install click-listener on every button + <canvas>
cc2 = get Canvas Rendering Context
drawBackground
background = get ImageData
requestList
drawBackground
cc2.beginPath()
cc2.fillRect() with black
cc2.closePath()
requestList
request Data-List
wait for response
response
extract list from response
wait for list
list
data = parsed list (JSON)
generateLocalData(data)

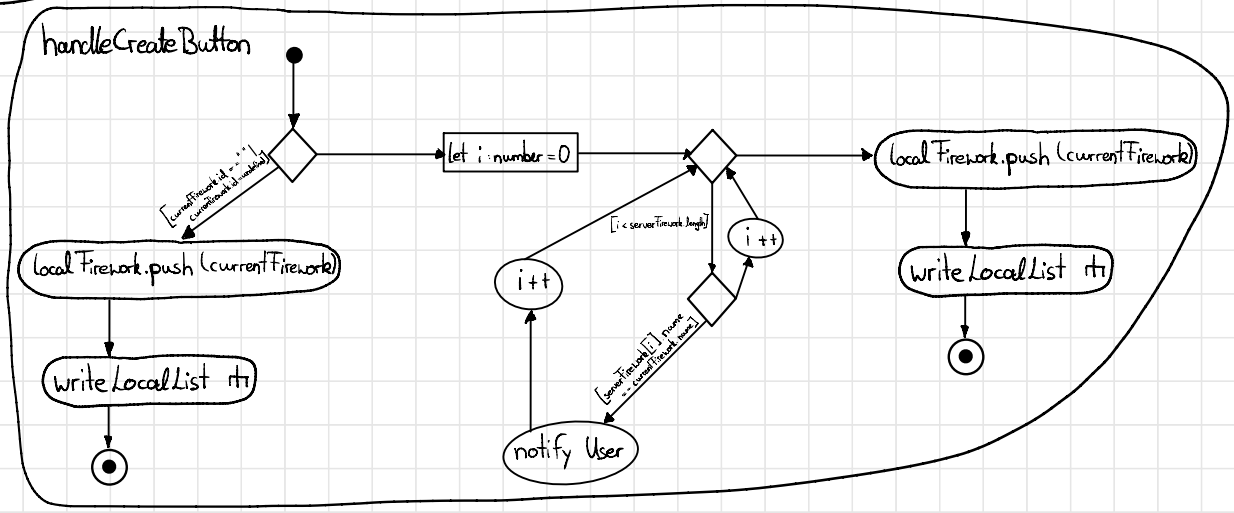

generateLocalData
_data : ReturnedJSON
clear serverFirework
let keys : string[] = Object.keys(_data.data)
let values : string[] = Object.values(_data.data)
let i : number = 0
writeServerlist
[i < keys.length]
i ++
let item : string[] = Object.values(values[i])
serverFirework.push data from item

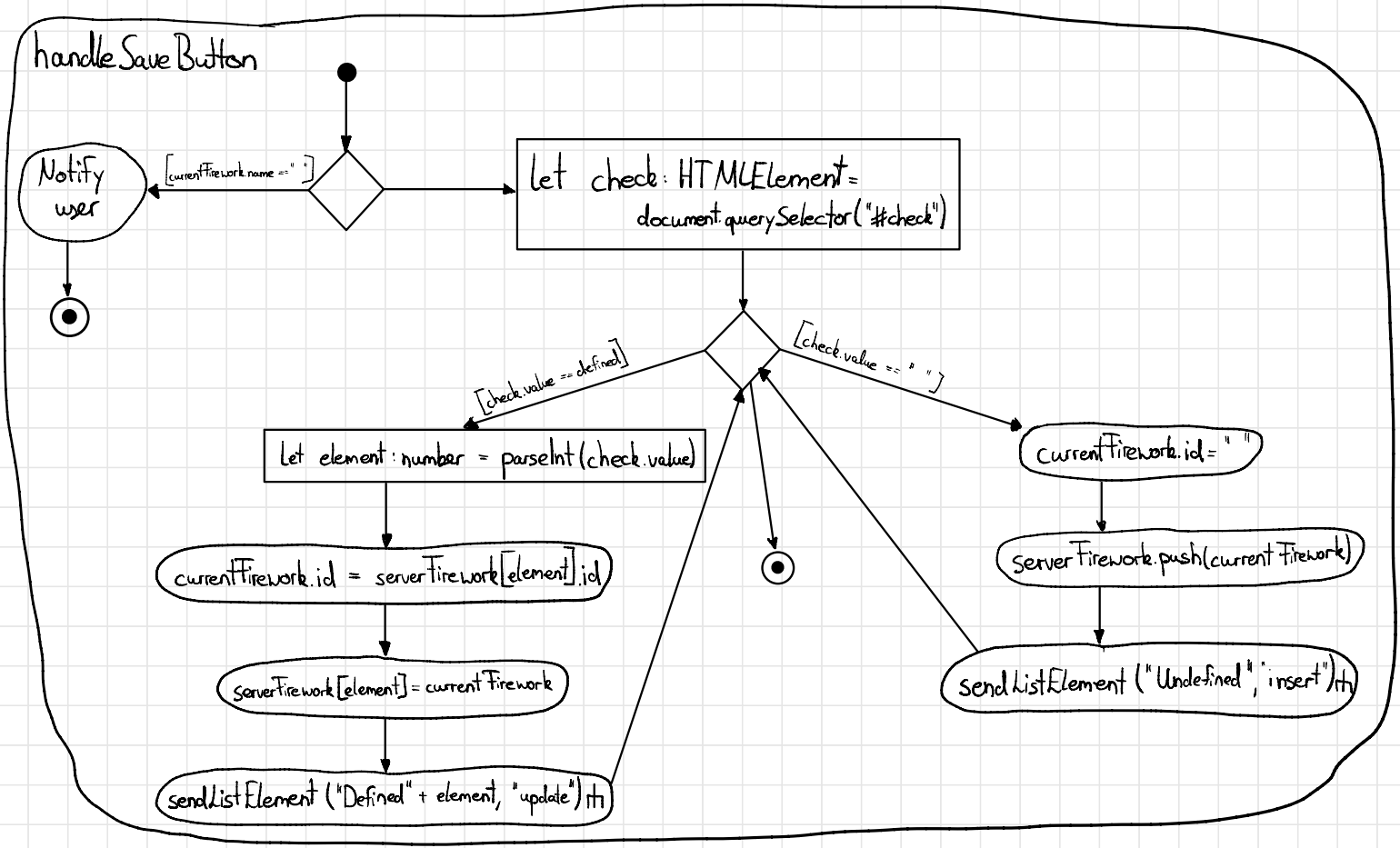

writeServerList
let list : HTMLElement = "serverlist"
list.innerHTML = ""
let i : number = 0
i < serverFirework.length
create li - Element for every serverFirework - Data each with unique id
i++
let i : number = 0
i < serverFirework.length
install click - Listener on every li - Element
i++
handleClick
_event: MouseEvent
let id : string = (_event.target as Element).id
[id.includes "delete"]
[id.includes "use"]
let newID : number = cutID(id, 6)
let newID : number = cutID(id, 3)
sendListElement("Defined" + newID, "delete")
useFirework(serverFirework[newID])
cutID
_id : string, _length : number
let newID : string = _id.slice(-length)
parseInt(newID)


_element: string
_command: string
SendListElement
[_element.includes("Defined") && _command != "delete"]
[_element.includes("Defined") && _command == "delete"]
[_element == "Undefined"]
let newElement: number = cutID(_element, 7)
let json : FireworkComponents = serverFirework[newElement]
let query : URLSearchParams = new URLSearchParams
let newElement: number = cutID(_element, 7)
let query : URLSearchParams = new URLSearchParams
let newElement: number = serverFirework.length - 1
let json : FireworkComponents = serverFirework[newElement]
let query : URLSearchParams = new URLSearchParams
query.set("command", _command)
query.set("collection", fireworks)
query.set("id", serverFirework[newElement].id)
query.set("command", _command)
query.set("collection", fireworks)
query.set("data", JSON.stringify(json))
query.set("command", _command)
query.set("collection", fireworks)
query.set("data", JSON.stringify(json))
query.set("id", serverFireworks[newElement].id)
send query to server
wait for response
notifyUser
wait for response
notifyUser
wait for response
notifyUser


useFirework
_firework: FireworkComponents
get HTML inputs and save them
HTMLInputs.value = _firework


handleInputChange
_event: Event
get value of every HTML Input and save it in currentFirework


handleCreateButton
[currentFirework.id = = = ""]
[currentFirework.id.isNotEmpty]
let i: number = 0
localFirework.push (currentFirework)
writeLocalList
[i < serverFirework.length]
i++
i++
[serverFirework[i].fname = = = currentFirework.fname]
notify User
localFirework.push (currentFirework)
writeLocalList


handleSaveButton
Notify user
[currentFirework.name == ""]
let check: HTMLElement = document.querySelector("#check")
[check.value == defined]
[check.value == " "]
let element: number = parseInt(check.value)
currentFirework.id = serverFirework[element].id
serverFirework[element] = currentFirework
sendListElement("Defined" + element, "update")
currentFirework.id = ""
serverFirework.push(currentFirework)
sendListElement("Undefined", "insert")

writeLocalList