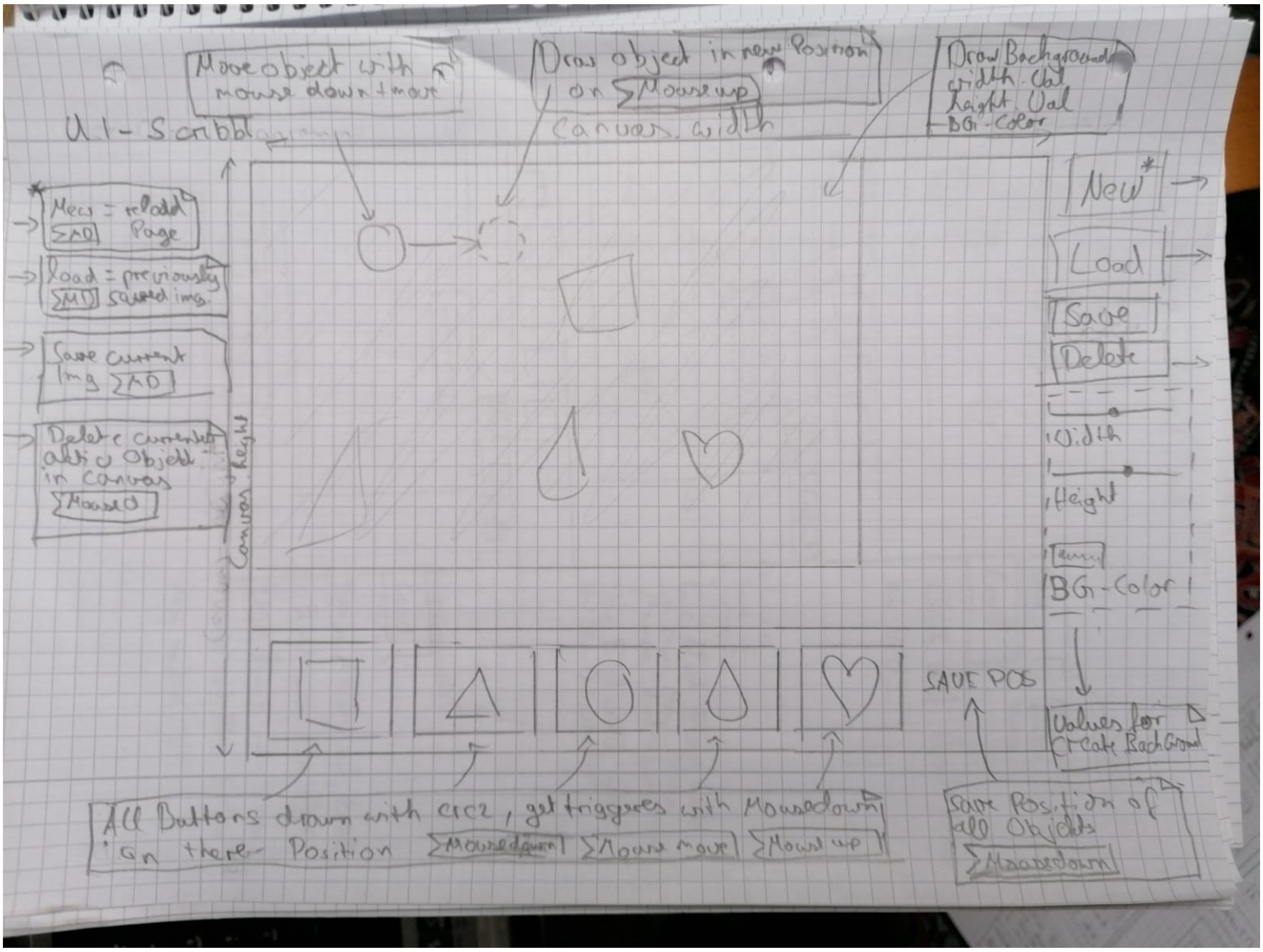
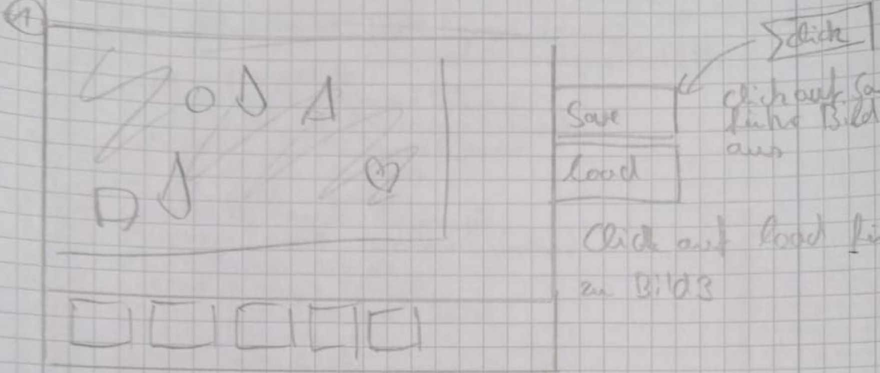


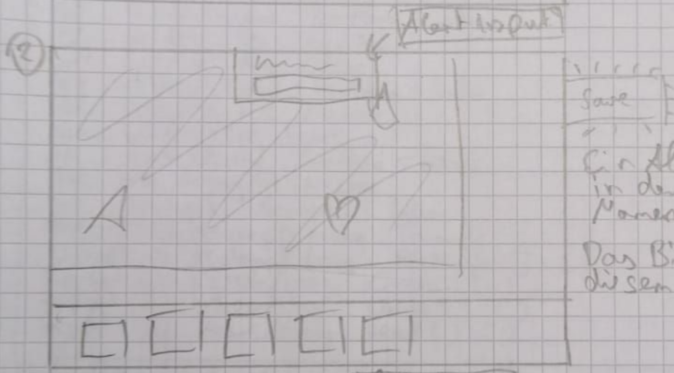
U I - Scribble
Move object with mouse down + move
Draw object in new Position on ∑Mouseup
Draw Background width Val height Val BG-Color
Canvas width
Canvas height
New = reload Page ∑MD
Load = previously saved img. ∑MD
Save current Img ∑MD
Delete current active Object in Canvas ∑MouseD
New
Load
Save
Delete
Width
Height
BG-Color
Values for Create BackGround
SAVE POS
All Buttons drawn with crc2, get triggered with Mousedown on there Position ∑Mousedown ∑Mouse move ∑Mouse up
Save Position of all Objects ∑Mousedown

User Interface Save/load

①


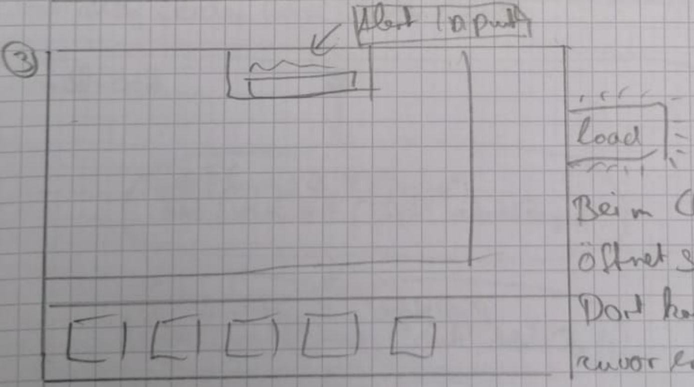
Click auf Save führt Bild ② aus

Click auf load führt zu Bild 3

②


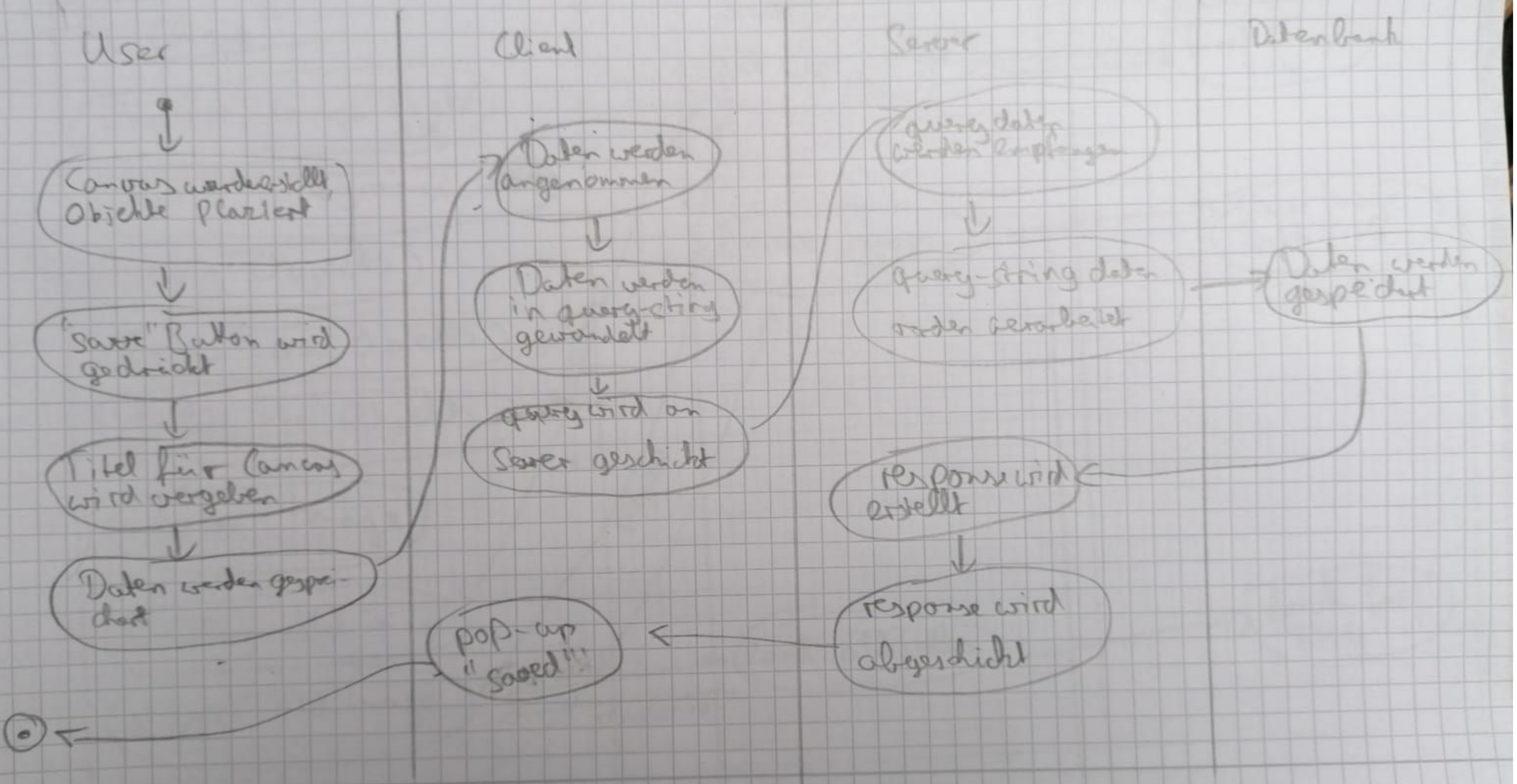
Ein Alert Ploppt auf in dem Man einen Namen eintragen muss.

Das Bild wird unter diesem Gespeichert

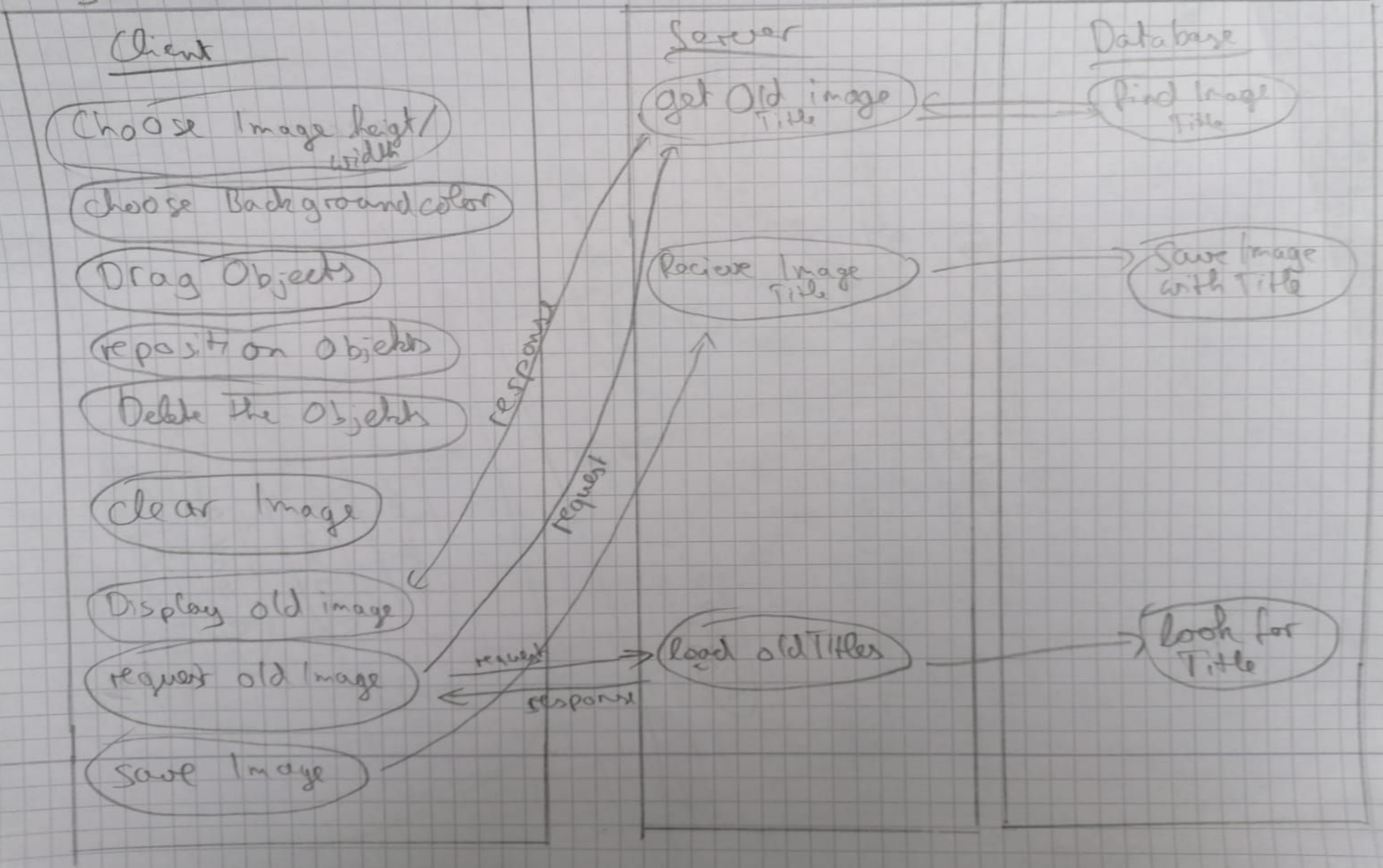
③


Beim Click auf load öffnet sich ein Alert. Dort kann man seine zuvor erstellten Bilder neu laden.

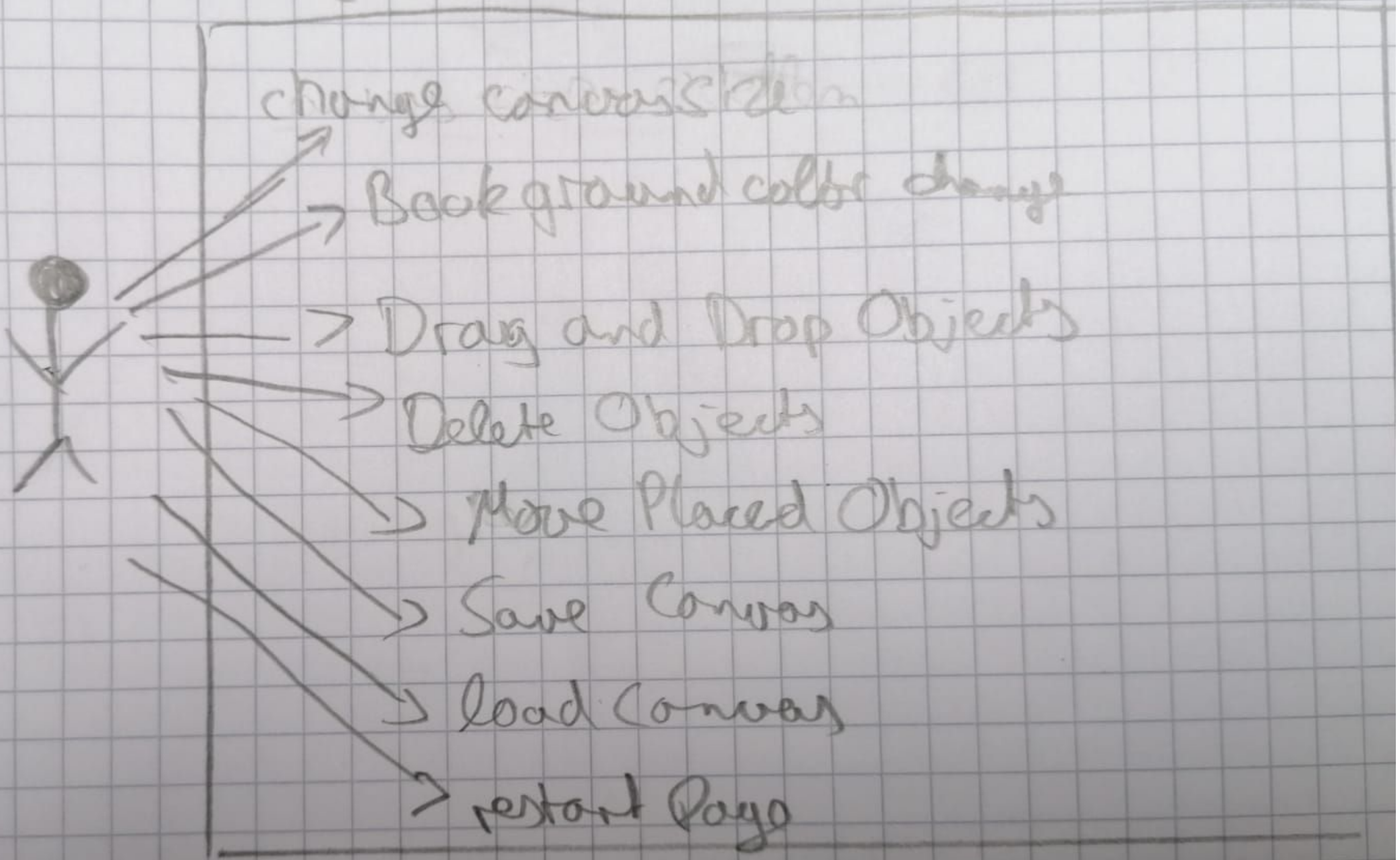
Dom übergreifendes Aktivitätsdiagramm


User
Client
Server
Datenbank
Canvas wurde erstellt, Objekte plaziert
"save" Button wird gedrückt
Titel für Canvas wird vergeben
Daten werden gespeichert
Daten werden angenommen
Daten werden in query-string gewandelt
query wird an Server geschickt
query daten wurden empfangen
query-string daten werden verarbeitet
Daten werden gespeichert
response wird erstellt
response wird abgeschickt
pop-up "saved"!

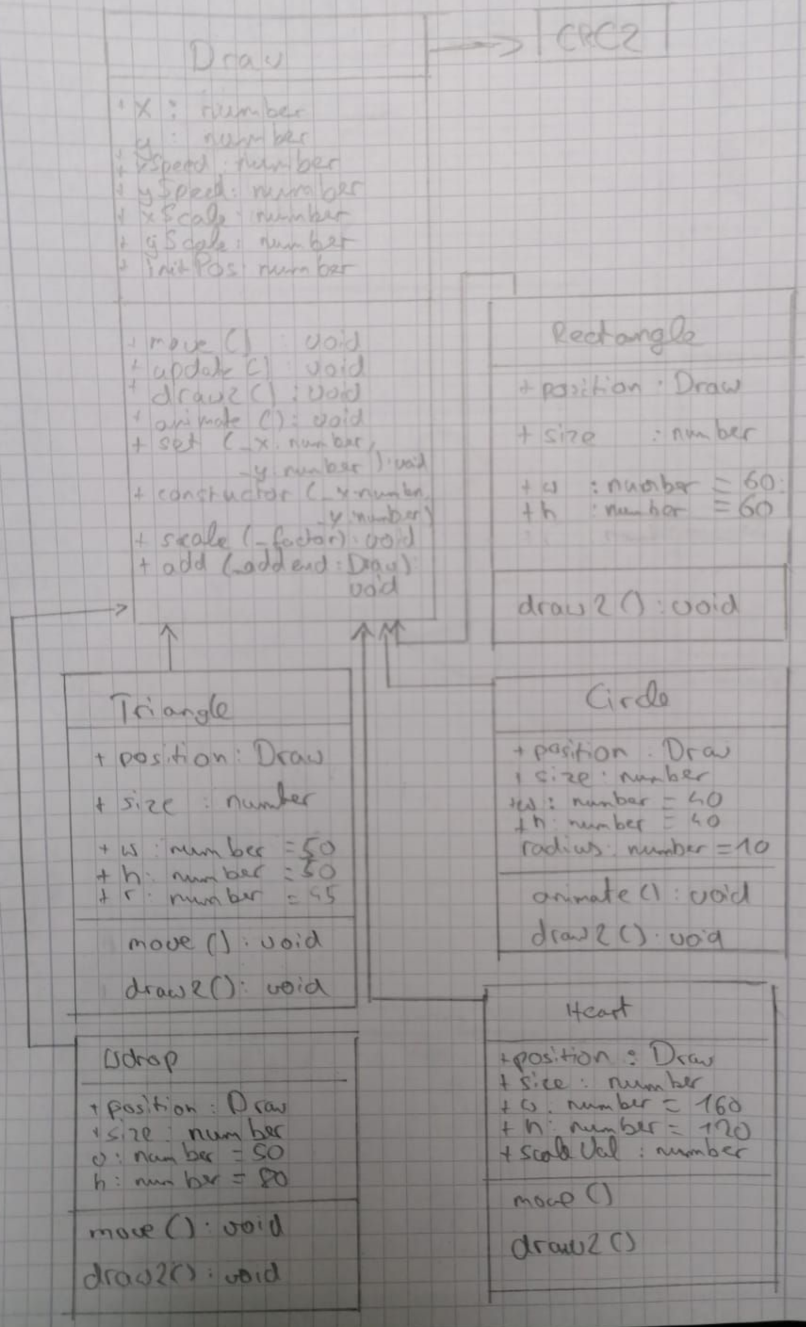
Server - Client - Comunication



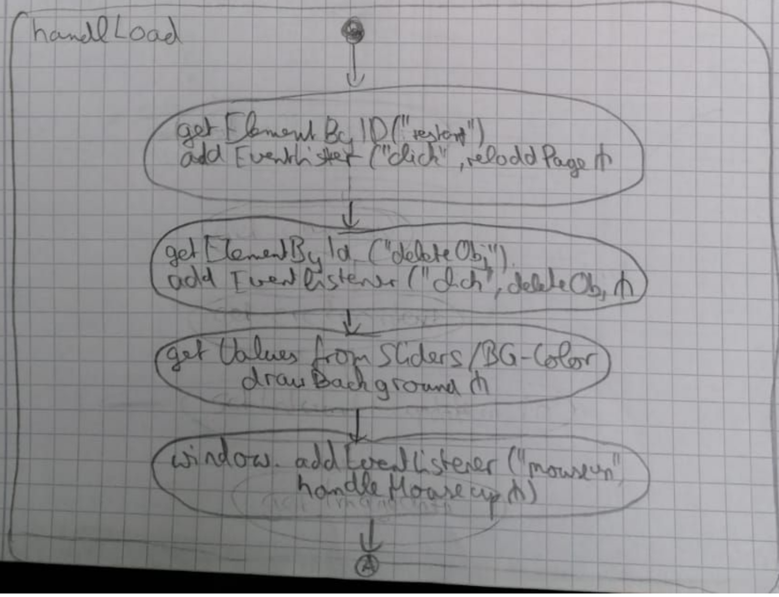
Use case Diagram



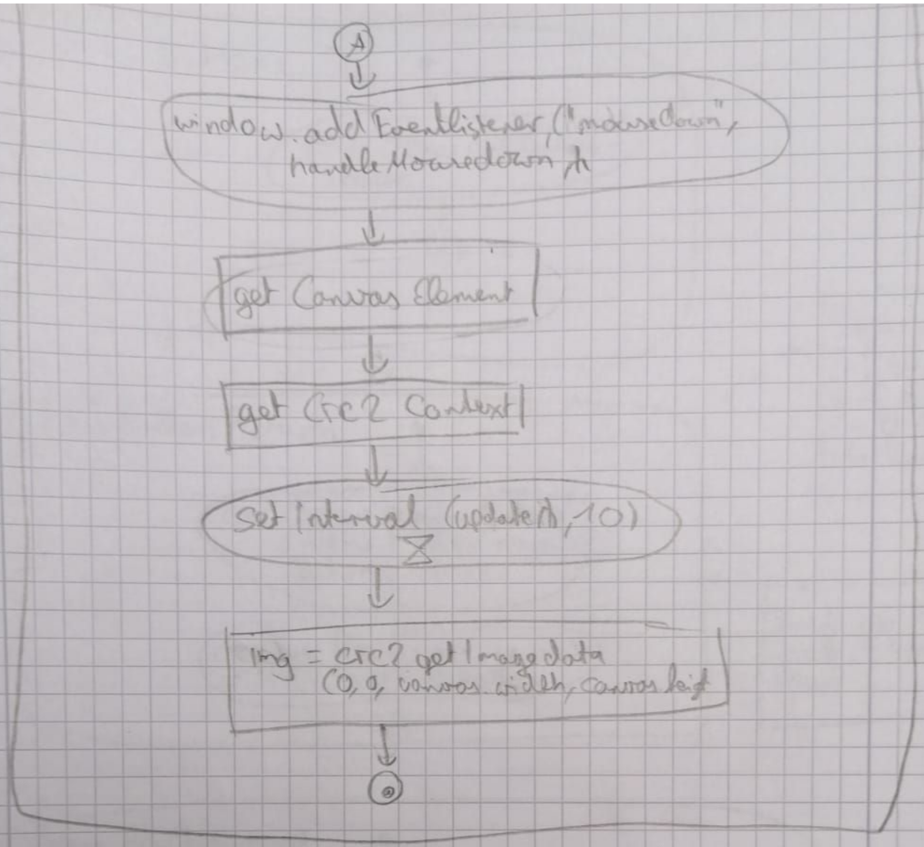
Class Diagram

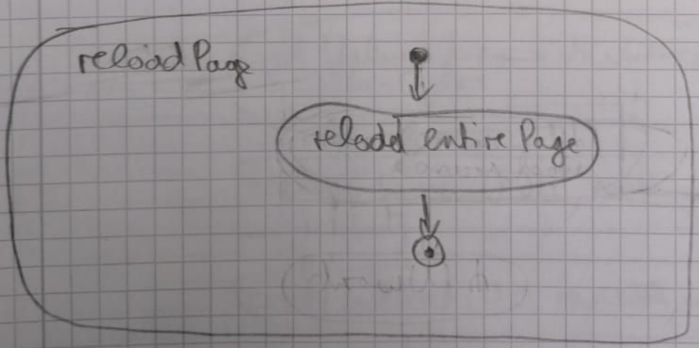


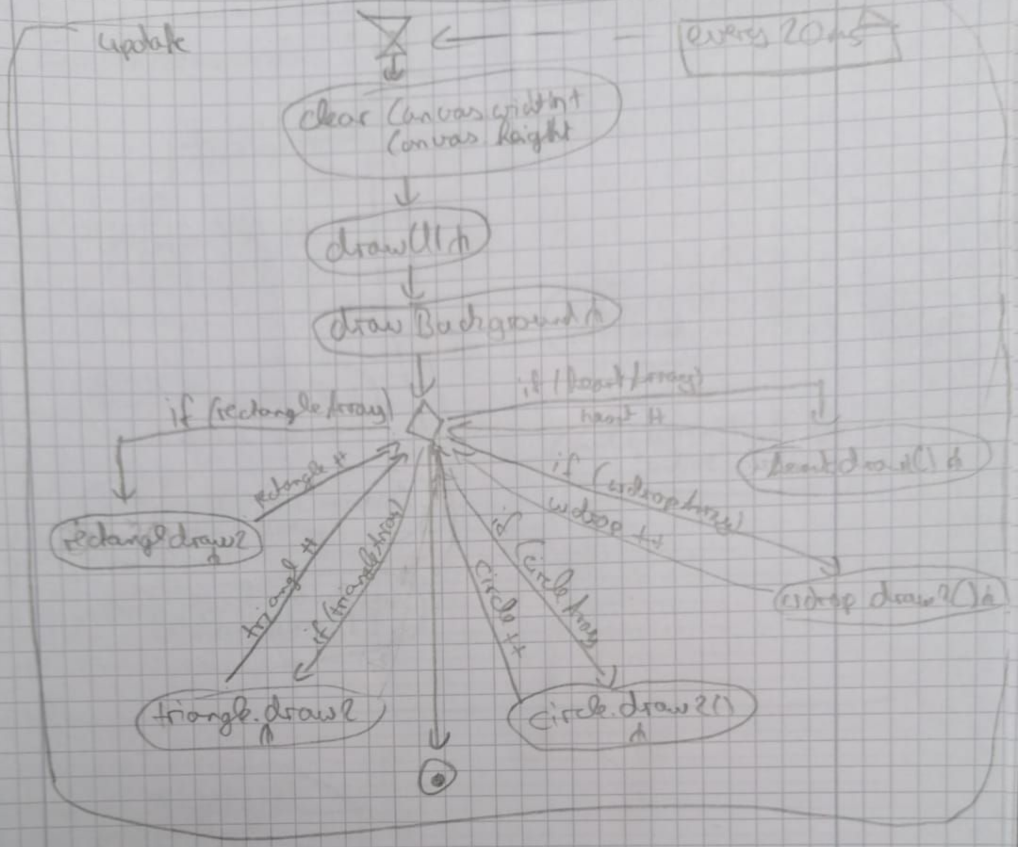
```
window addeventListener ("load", handlLoad ());

export let canvas: HTMLCanvasElement;
export let crc2 : CanvasRenderingContext2D;
let img : Imagedata;
let widthSlider: HTMLInputElement;
let heightSlider: HTMLInputElement;
let yourColor: HTMLInputElement
export let width: number;
export let heigt : number;
let currentRectangle: Rectangle;
let rectanglPresent: boolean = false;
let recArrayPos : number;
let currentTriangle: Triangle;
let trianglPresent: boolean = false;
let triArrayPos : number;
let currentCircle: Circle;
let circlePresent : boolean = false;
let cirArrayPos : number;
let currentWdrop: Wdrop;
let wdropPresent : boolean = false;
let wdrArrayPos : number
let currentHeart: Heart;
let heartPressent: boolean = false;
let heaArrayPos: number
let dragged : boolean = false;
export ~~let~~ triangleArray: Triangle [] = [];
export let rectangleArray: Rectangle [] = [];
export let circleArray: Circle [] = [];
export let wdropArray : Wdrop [] = [];
export let heartArray: Heart [] = [];
```
Variaben werden beim Starten erstellt

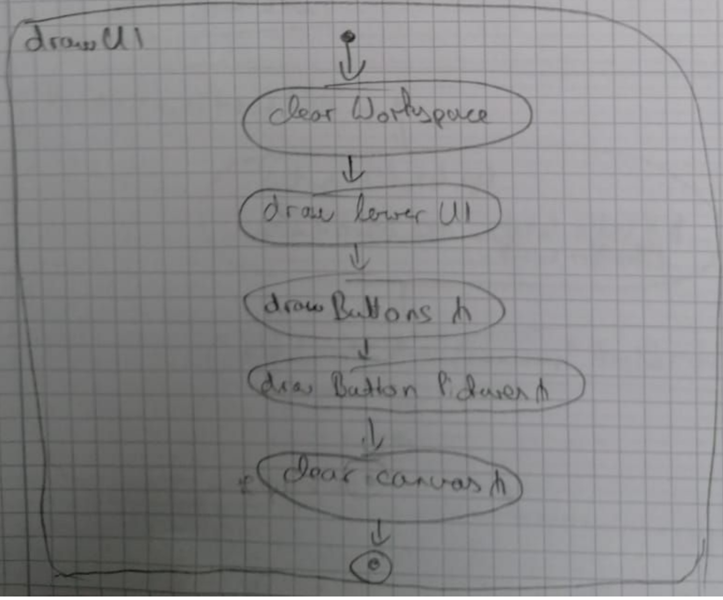

handlLoad

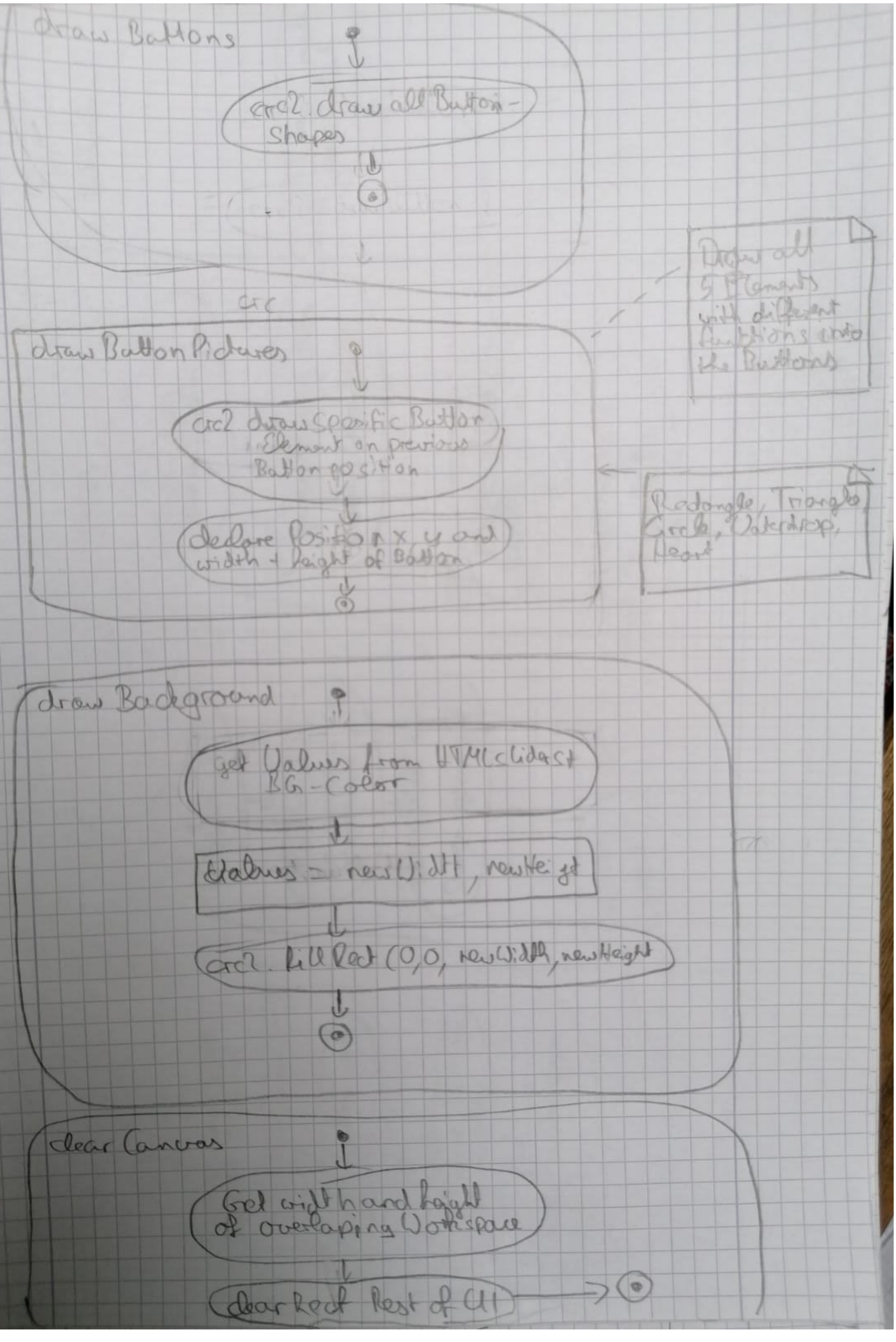
```
getElementById ("restart")
add EventListener ("click", reloadPage ())

getElementById ("deleteObj")
add EventListener ("click", deleteObj ())

get Values from Sliders / BG-Color
drawBackground ()

window. addEventListener ("mouseup",
handleMouseup ())
```

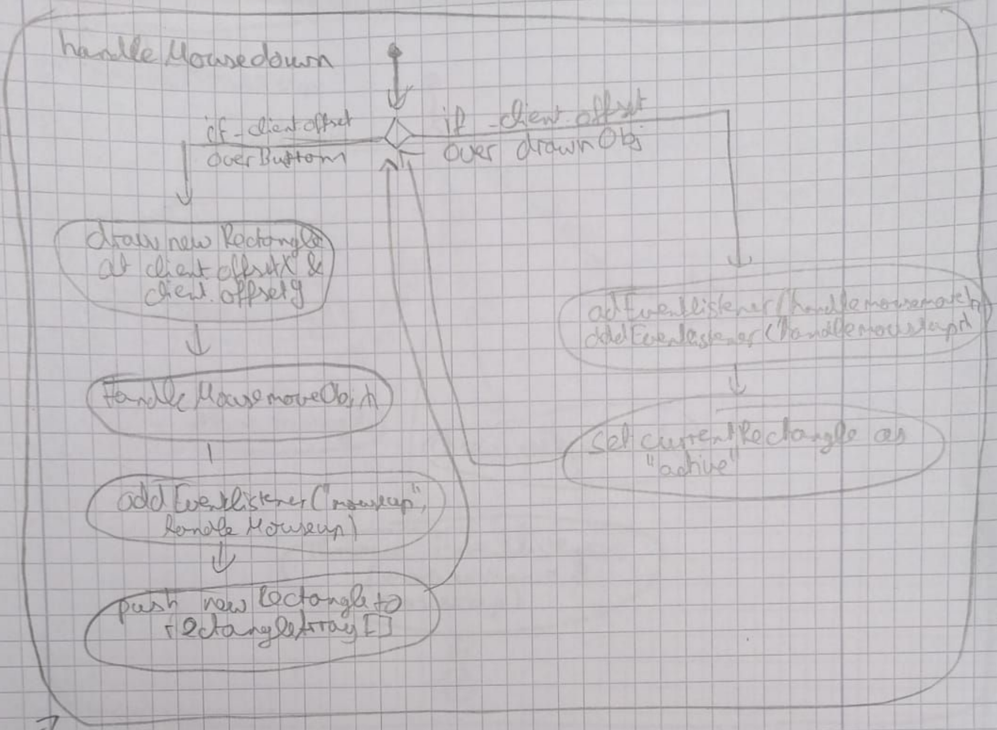

A
window.addEventListener("mousedown", handleMousedown);
get Canvas Element
get Crc2 Context
setInterval (update(), 10)
img = crc2.getImagedata (0, 0, canvas.width, canvas.heigt


reloadPage
reload entire Page

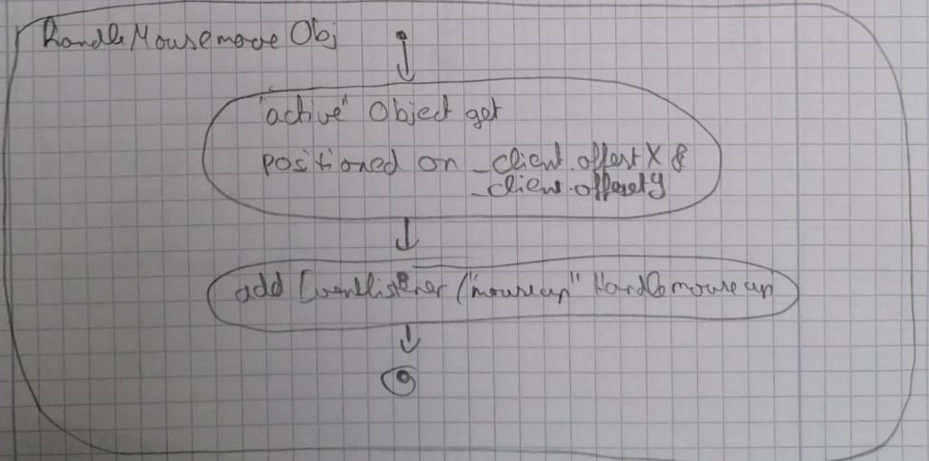

update
every 20ms
clear Canvas width+
Canvas height
drawUI
draw Background
if (rectangle Array)
rectangle ++
rectangle draw2
if (heart Array)
heart ++
heart draw2()
if (wdrop Array)
wdrop ++
wdrop draw2()
if (triangle Array)
triangle ++
triangle.draw2
if (circle Array)
circle ++
circle.draw2()

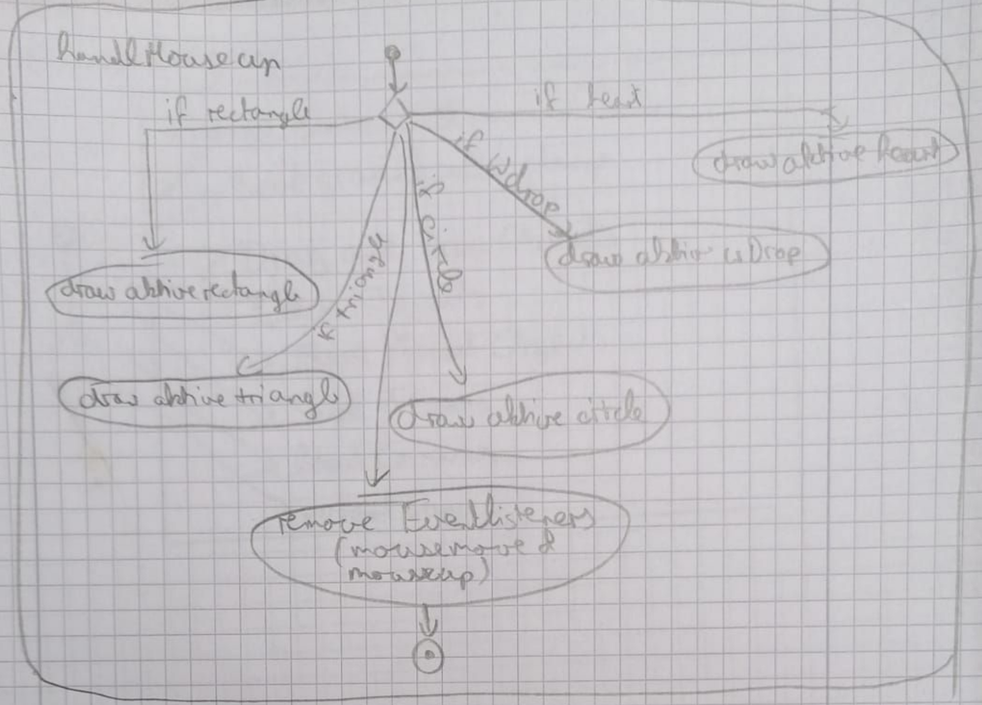

drawUI
clear Workspace
draw lower UI
draw Buttons
draw Button Pictures
clear canvas

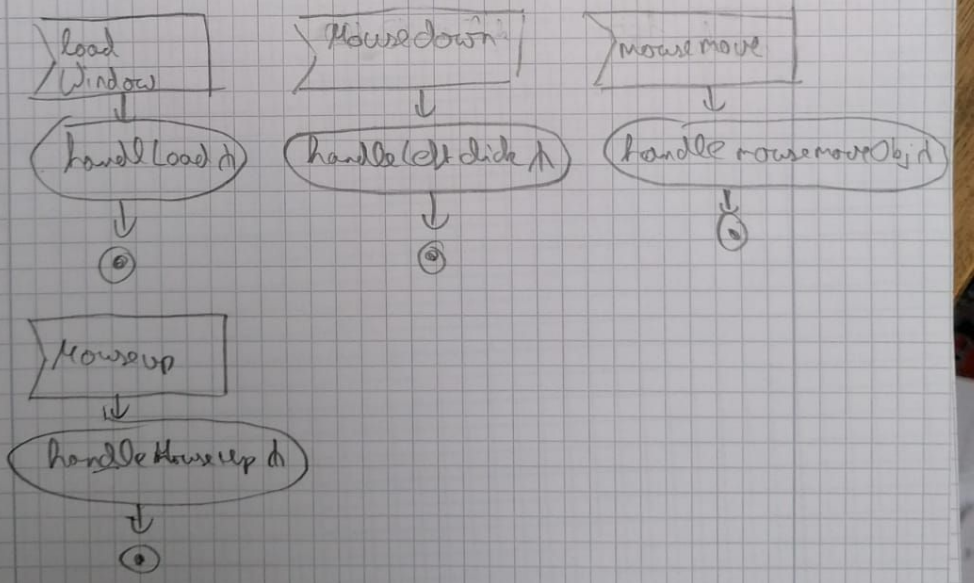

draw Buttons
crc2.draw all Button-Shapes
crc
draw Button Pictures
crc2 draw specific Button Element on previous Button position
declare Position x y and width + height of Button
Draw all 5 Elements with different functions onto the Buttons
Rectangle, Triangle, Circle, Waterdrop, Heart
draw Background
get Values from HTMLclidast BG-Color
Values = newWidth, newHeight
crc2. fillRect (0,0, newWidth, newHeight
clear Canvas
Get width and height of overlaping Workspace
clearRect Rest of CH

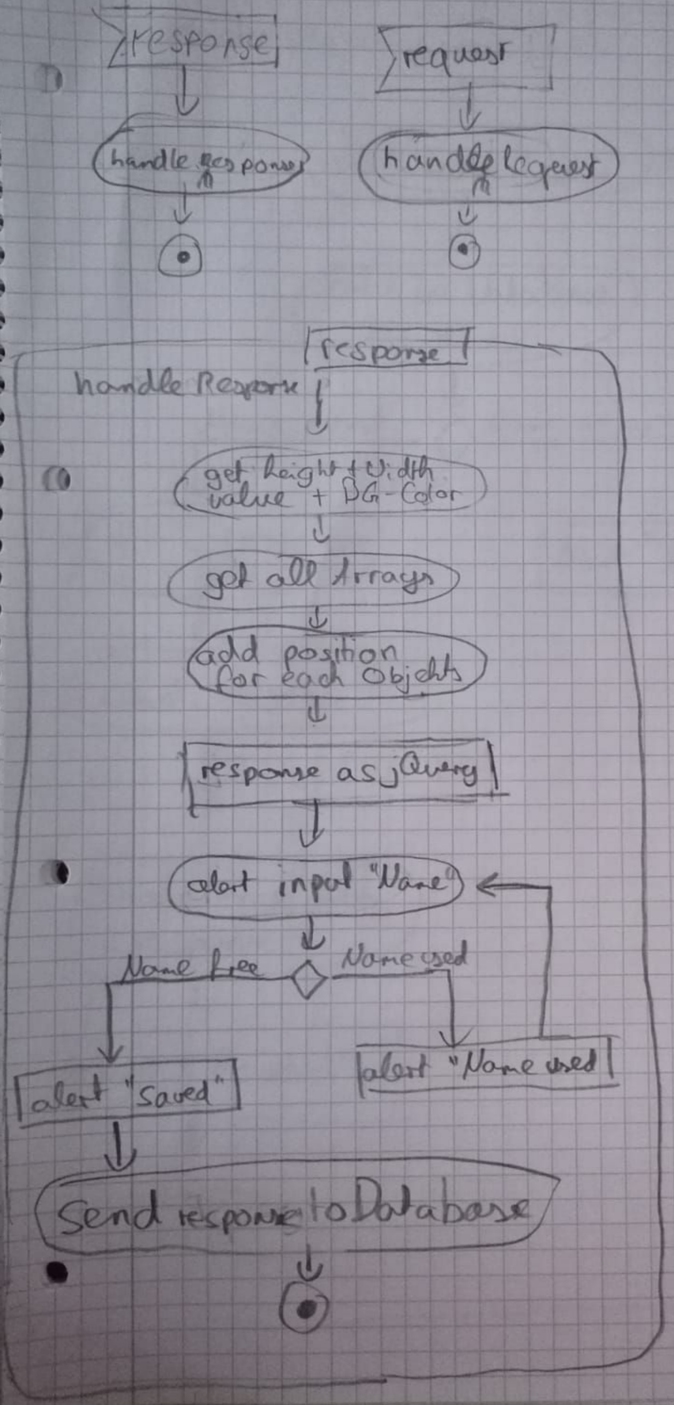

handle Mousedown
if _client.offset over Button
if _client.offset over drawnObj
draw new Rectangle at client.offsetX & client.offsetY
handle MousemoveObj()
addEventlistener("mouseup", handle Mouseup)
push new Rectangle to rectangleArray[]
adEventlistener("handlemousemove")
addEventlistener("handlemouseup")
Set currentRectangle as "active"

if Bedingung bezieht sich nicht nur auf Rectange sondern auf alle 3 Elemente sie steuern die selben Funktionen an

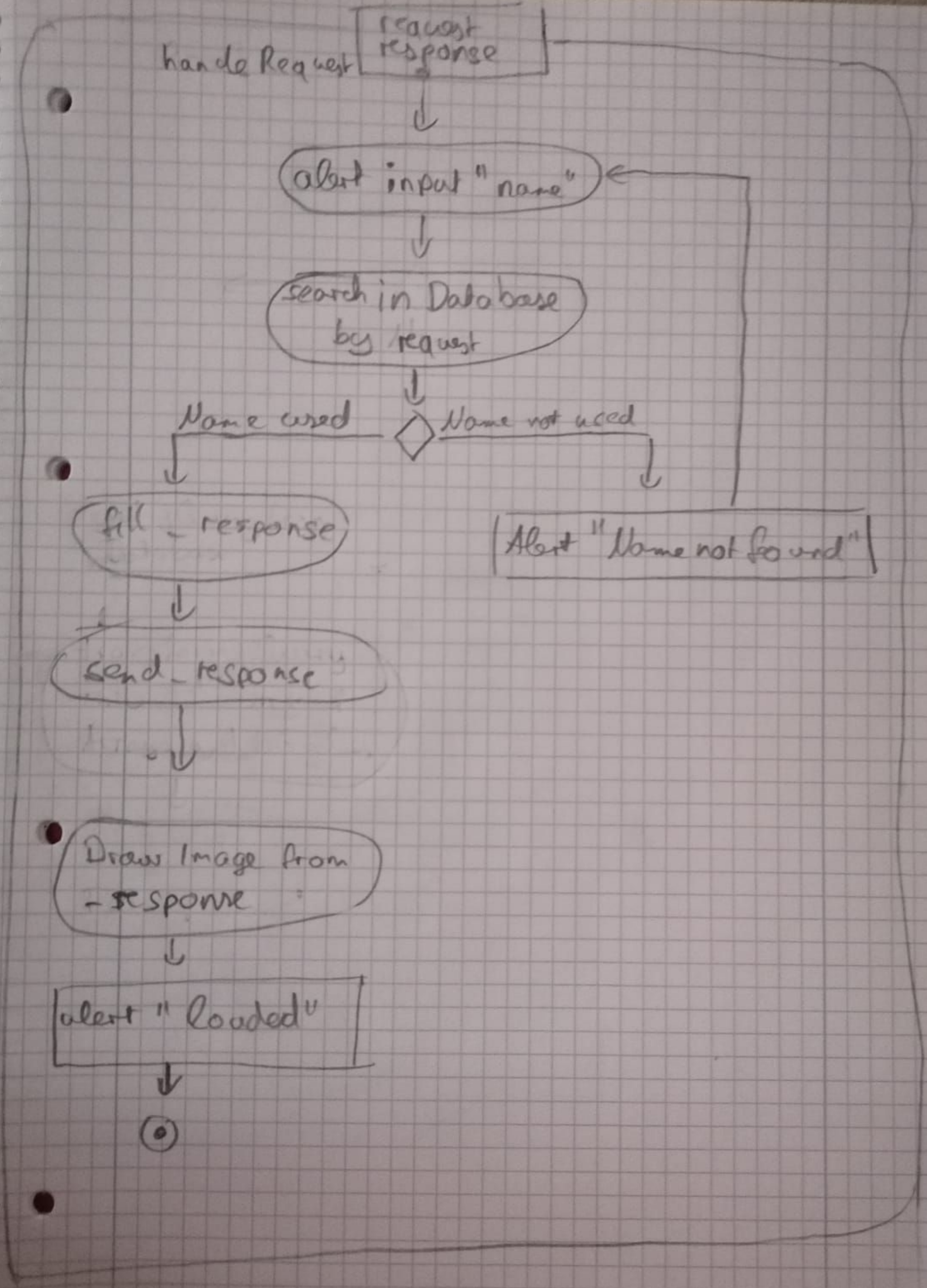

handleMousemove Obj
"active" Object get positioned on _client.offset X & _client.offsetY
add Eventlistener ("mouseup" Handlemouseup


handleMouseup
if rectangle
if heart
if WDrop
if circle
if triangle
draw aktive heart
draw aktive WDrop
draw aktive rectangle
draw aktive triangle
draw aktive circle
remove EventListeners (mousemove & mouseup)


load Window
handleLoad()
Mousedown
handleLeftClick()
mousemove
handle mousemoveObj()
Mouseup
handleMouseup()



response
request
handle response
handle request
response
handle Response
get height + width value + BG-Color
get all Arrays
add position for each Objekts
response as jQuery
alert input "Name"
Name free
Name used
alert "saved"
alert "Name used"
Send response to Database


request
response
handle Request
alert input "name"
search in Database
by request
Name used
Name not used
fill_response
Alert "Name not found"
send_response
Draw Image from
_response
alert "loaded"