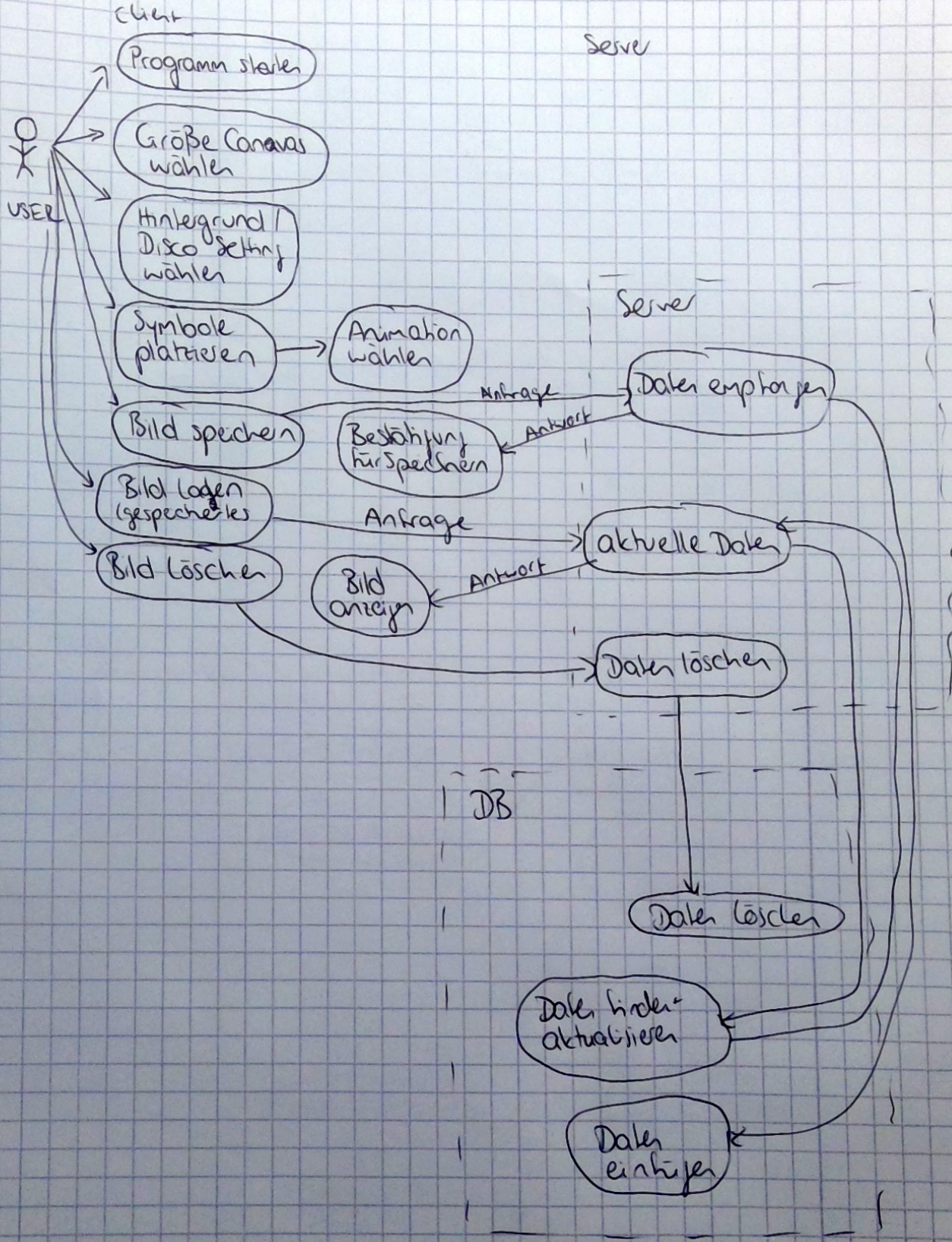
Anwendungsfalldiagramm Abschlussarbeit SOZO

Nathalie Schneider

TITEL: Lockdown Disco 2020



Schwimmbahnen - Abschlussaufgabe 2020

Bild speichern:

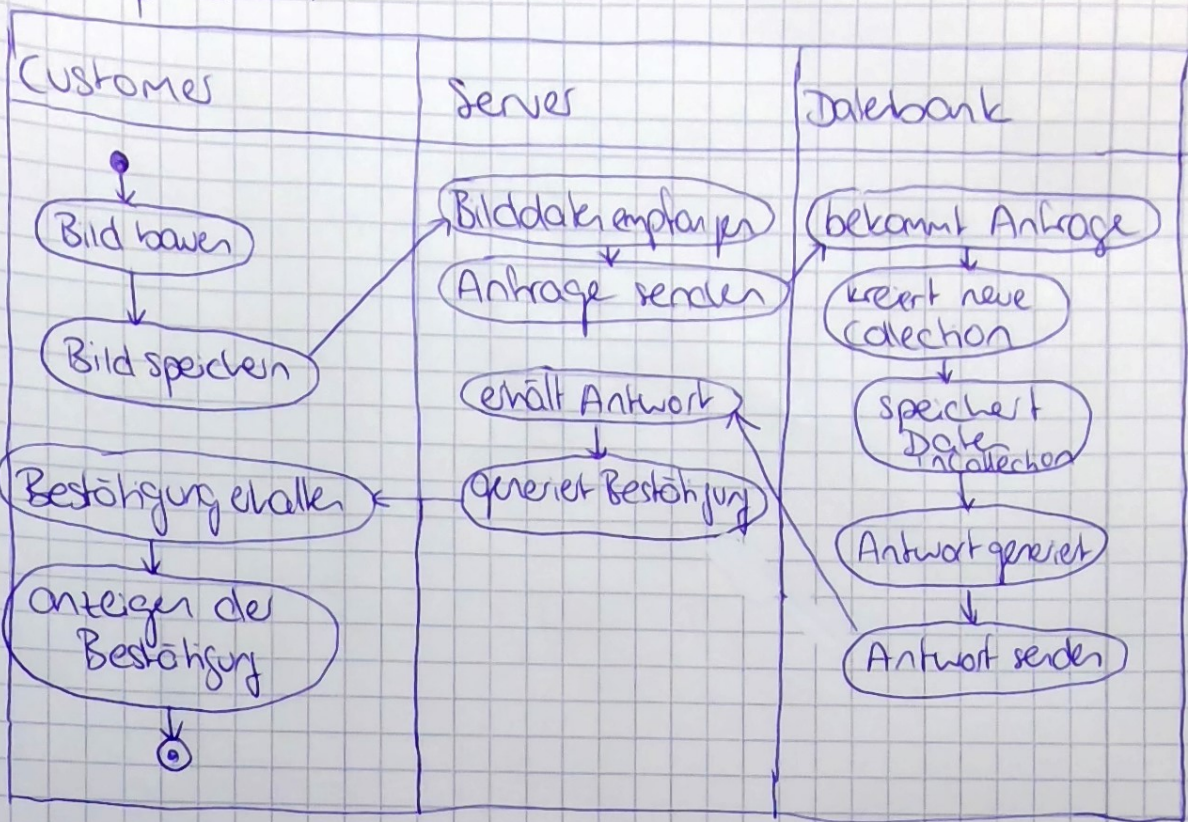
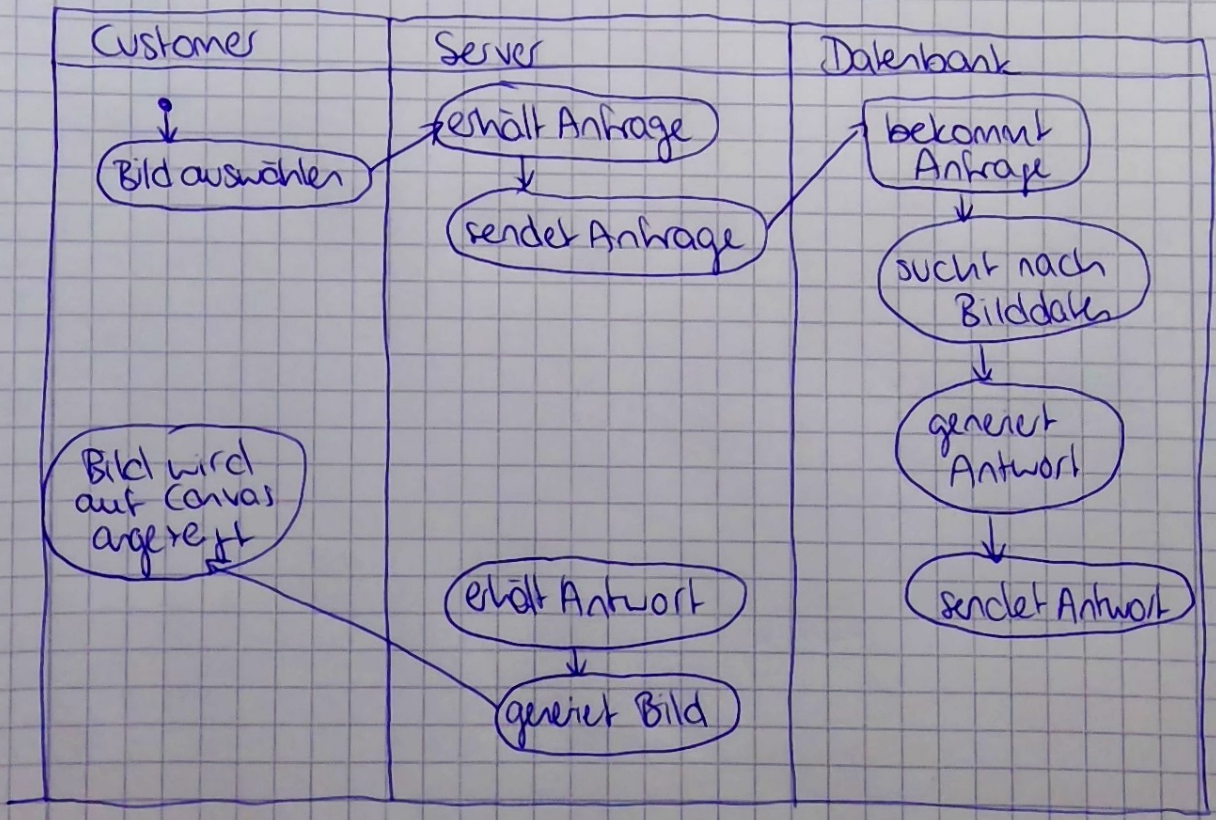
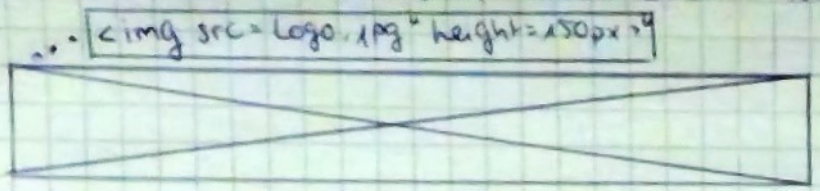
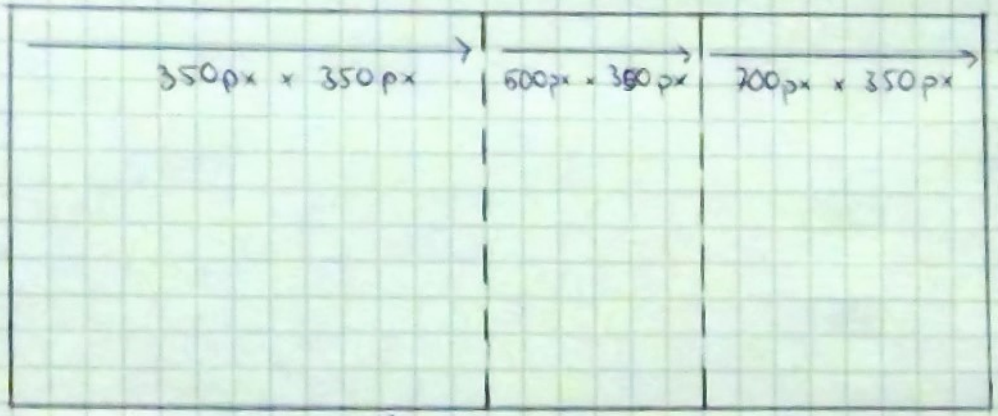


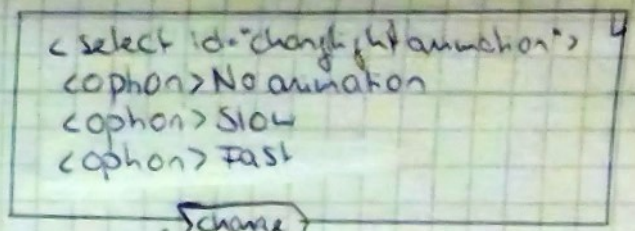
Bild laden:

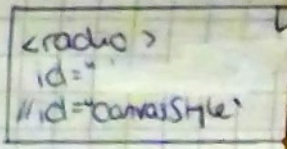


SCRIBBLE - ABSCHLUSSARBEIT SOSE 2020 - Nathalie Schneider



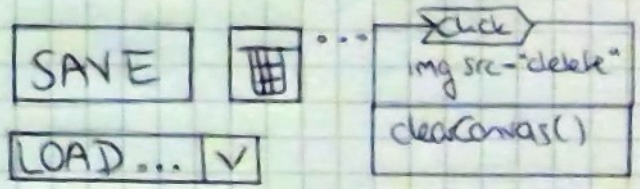


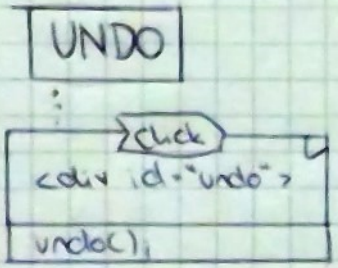


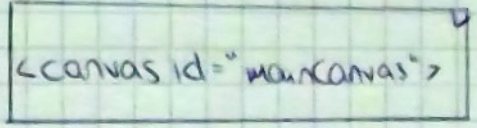


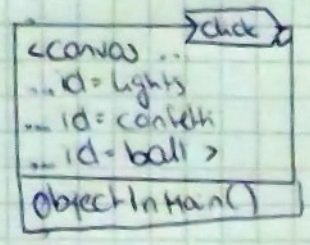
CHOOSE YOUR SIZE!
- ULTRAPANORAMA
- MEGASITE
- INFLUENCER STYLE

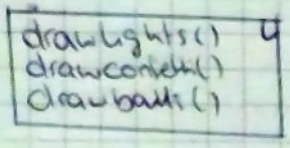
CHOOSE YOUR LOCATION!
- CLASSY
- FUNKY
- DARK

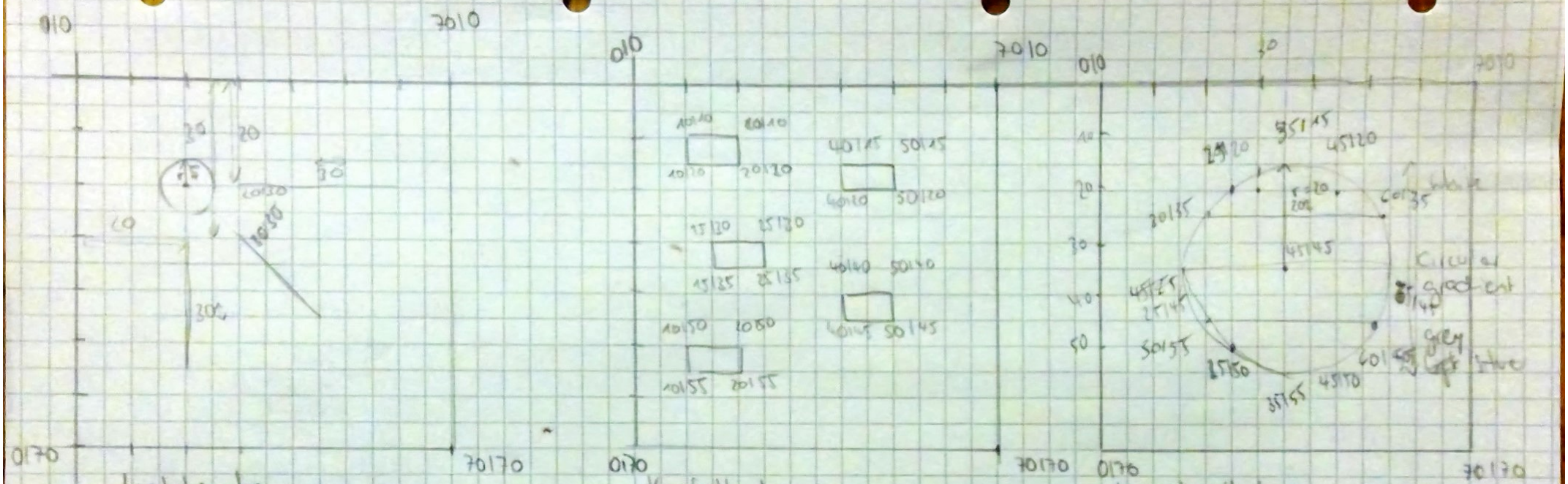


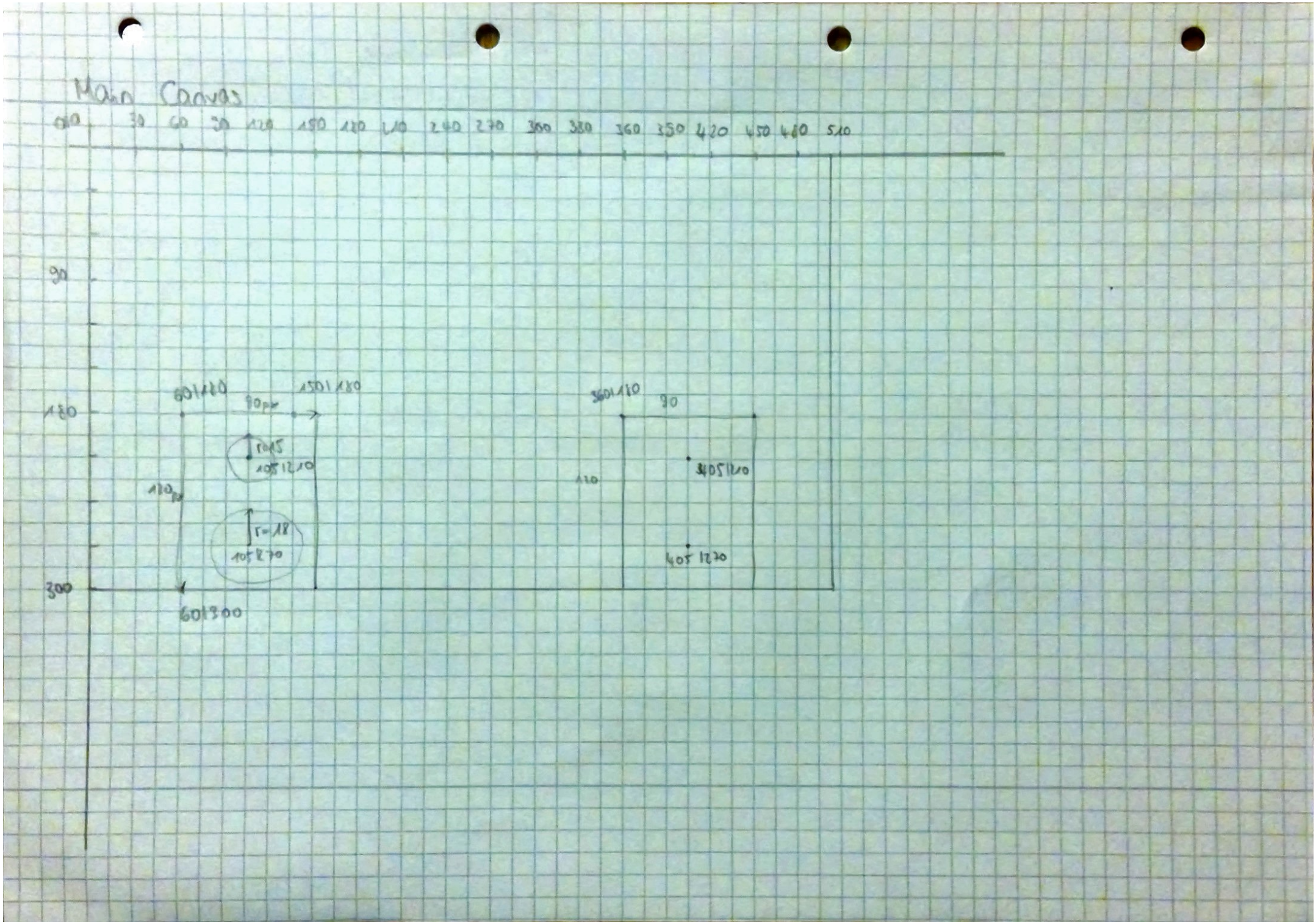




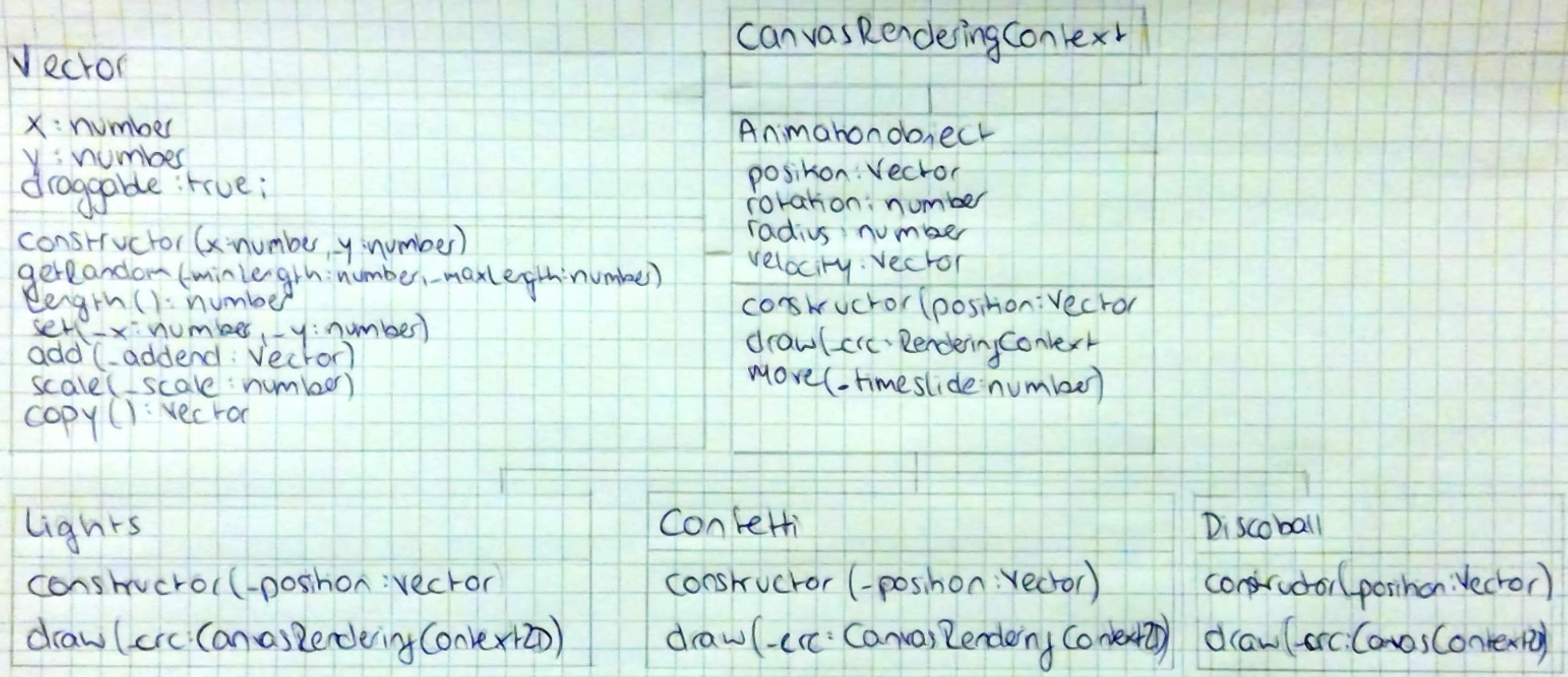




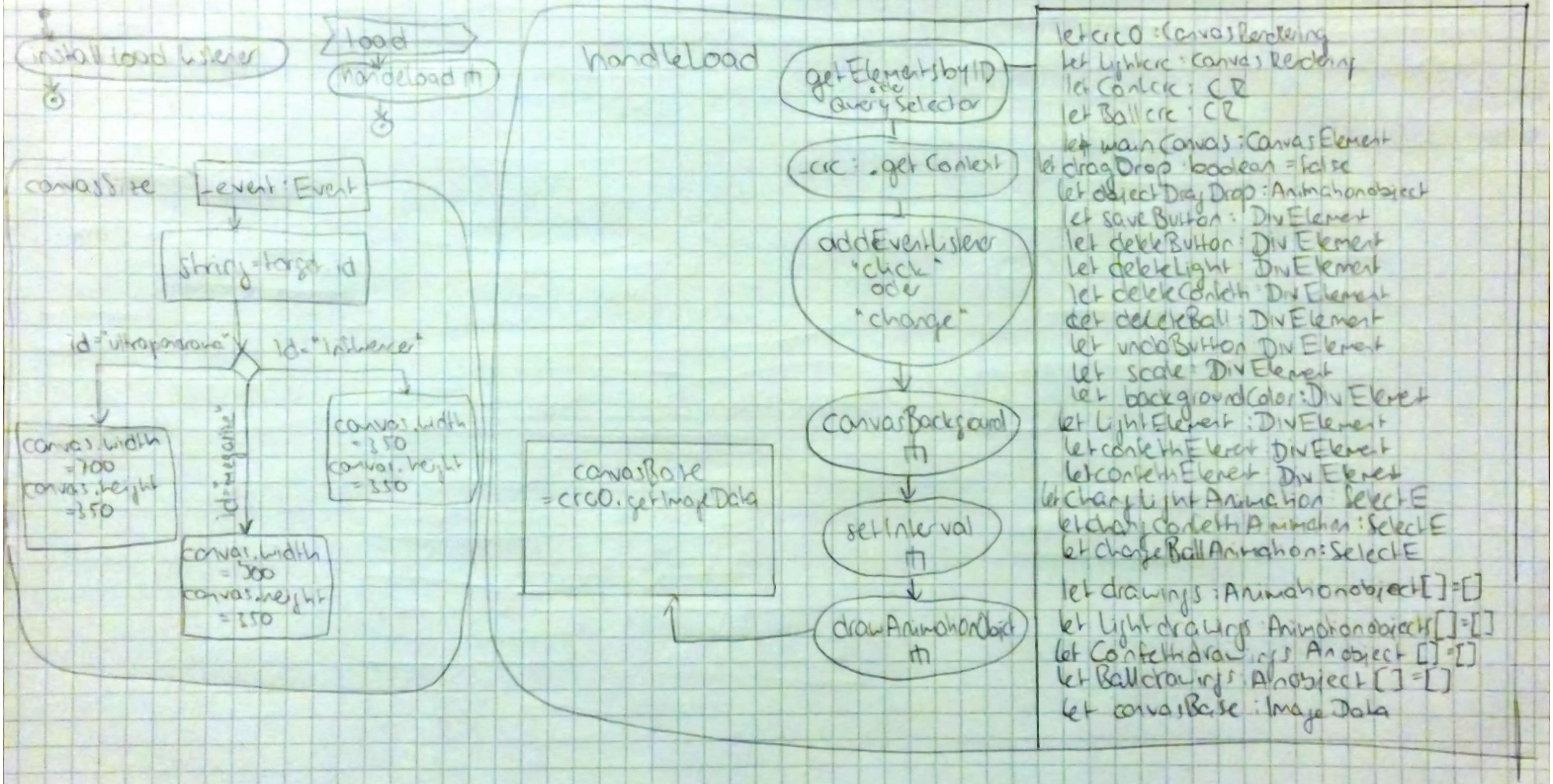

010
7010
0170
70170
Lights.ts
Animation: immer ein Strich ausgeblendet, abwechselnd
Variation Geschwindigkeit
Konfetti.ts
Animation: fällt nach unten
Variation: nach unten, nach oben
discoball.ts
Animation Drehen
Variation: links rum, rechts rum
Circular gradient
grey

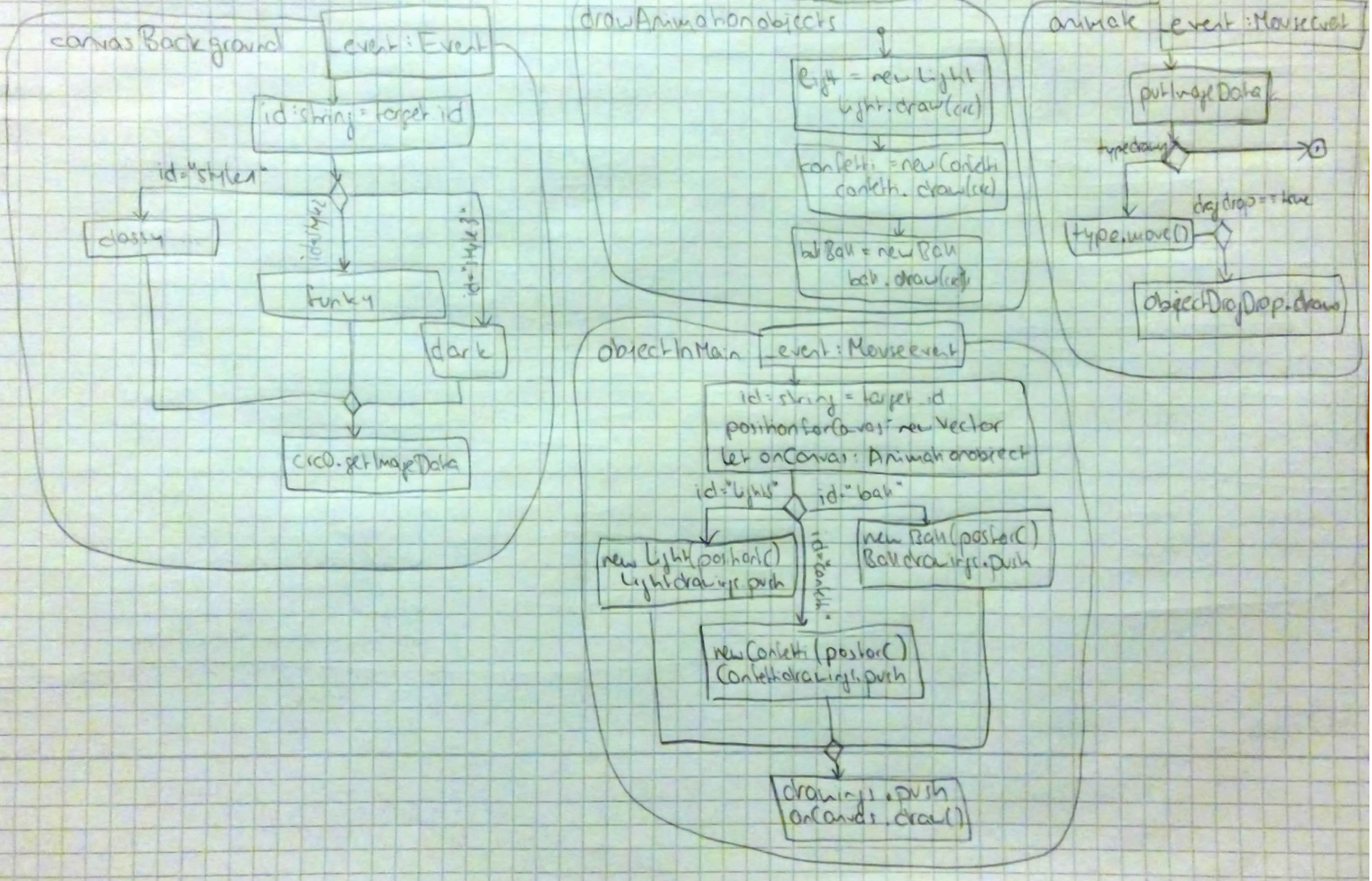

Main Canvas
0 30 60 90 120 150 180 210 240 270 300 330 360 390 420 450 480 510
90
180
300
60 180
150 180
90px
r=45
105 120
120px
r=18
105 270
60 300
360 180
90
405 120
120
405 270

KLASSENDIAGRAMM - Abschlussarbeit SoSe 20 - Nathalie Schneider



AKTIVITÄTSDIAGRAMM - Abschlussarbeit SOSE 20 - Nathalie Schneider




canvasBackground
event : Event
id : string = target.id
id="stylen"
classy
id="styles"
funky
id="styles3"
dark
circO.getImageData
drawAnimationObjects
light = new Light
light.draw(circ)
confetti = new Confetti
confetti.draw(circ)
ballBall = new Ball
ball.draw(circ)
objectInMain
event : Mouseevent
id : string = target.id
positionforCanvas: new Vector
let onCanvas: Animationobject
id="lights"
id="ball"
new Light(posforC)
Lightdrawings.push
id="confetti"
new Ball(posforC)
Balldrawings.push
new Confetti(posforC)
Confettidrawings.push
drawings.push
onCanvas.draw()
animate
event : Mouseevent
putImageData
typedrag
type.move()
dragdrop = true
objectDragDrop.draw

XY entweder lights, confetti oder balls

changeAnimationXY | _event: Event

value: string = target.value

case "noanimation"

type XYdrawings

type.velocity = new Vector(0,0)
type.draw(crc0)

"schell"

type.velocity = new Vector()
type.draw

"bargraph"

typeXYdrawings

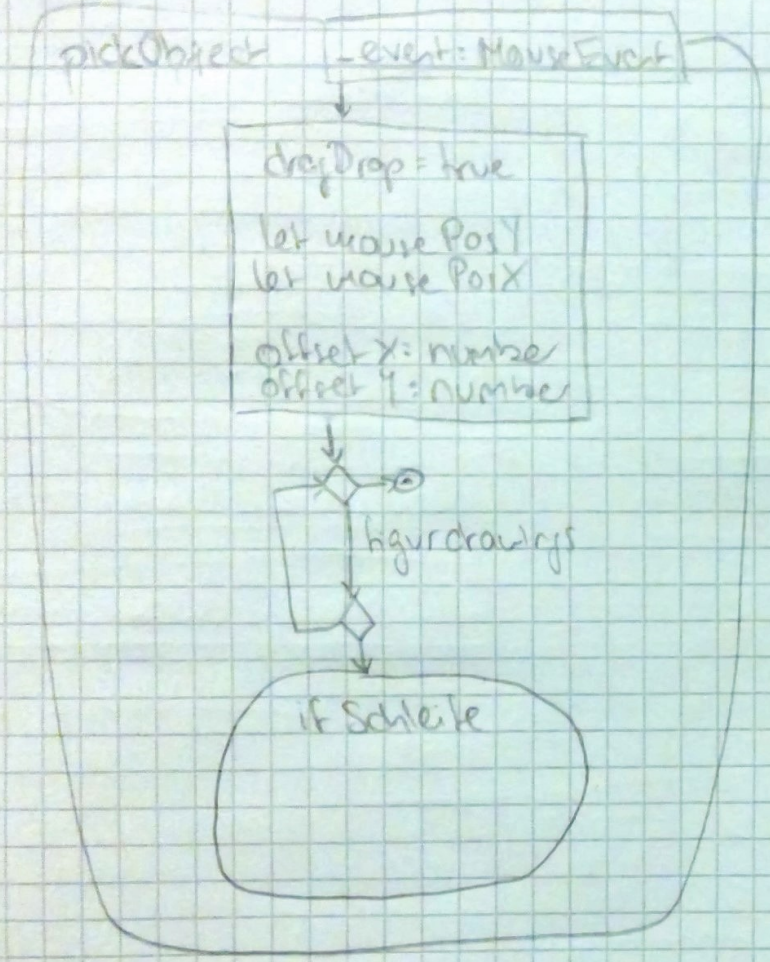
type.velocity = new Vector()
type.draw

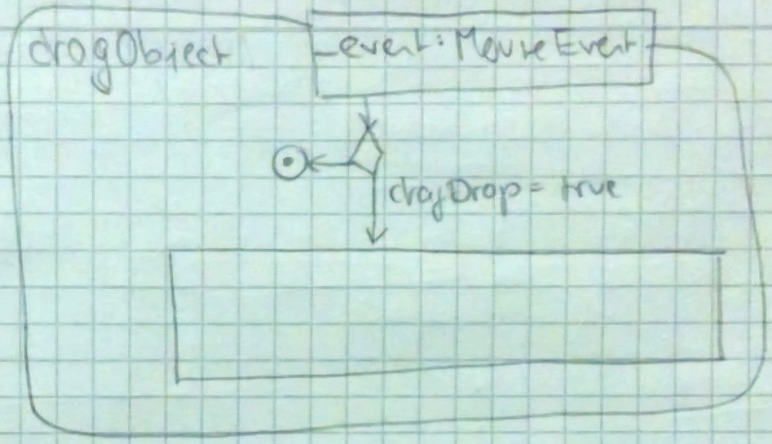
clearCanvas | _event: MouseEvent

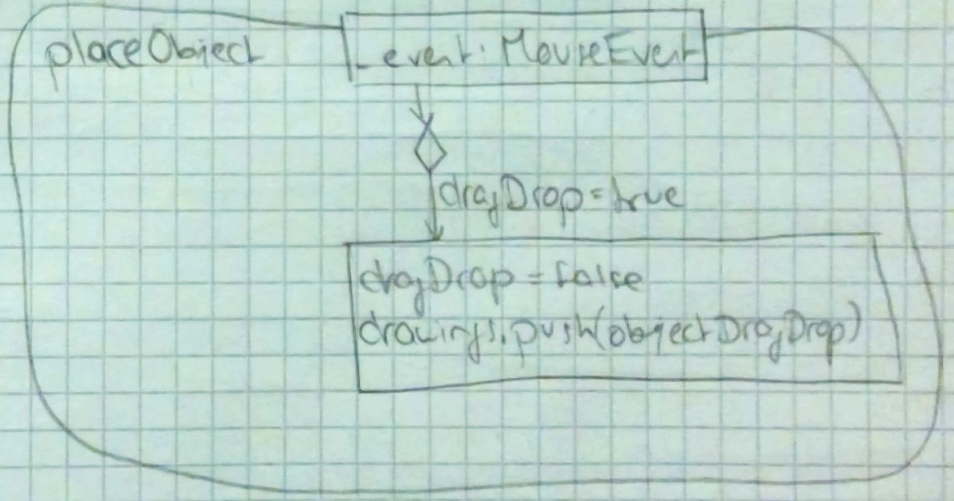
crc0.clearRect
drawings = []

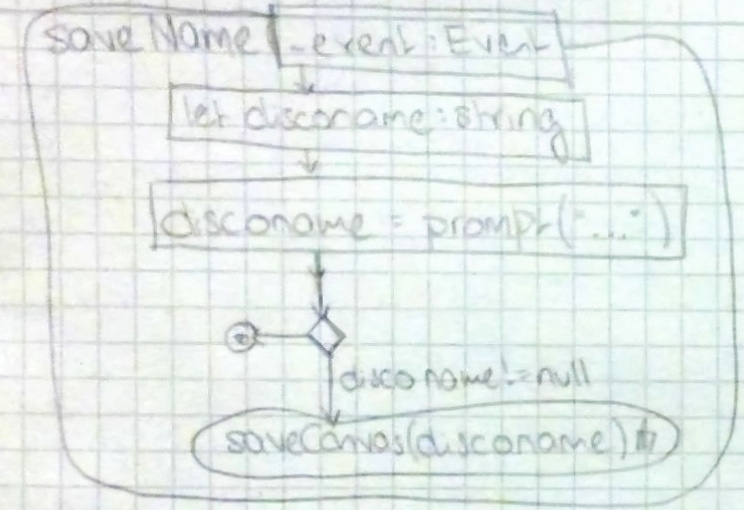
undo | _event: Event

drawings.splice(drawings.length-1)

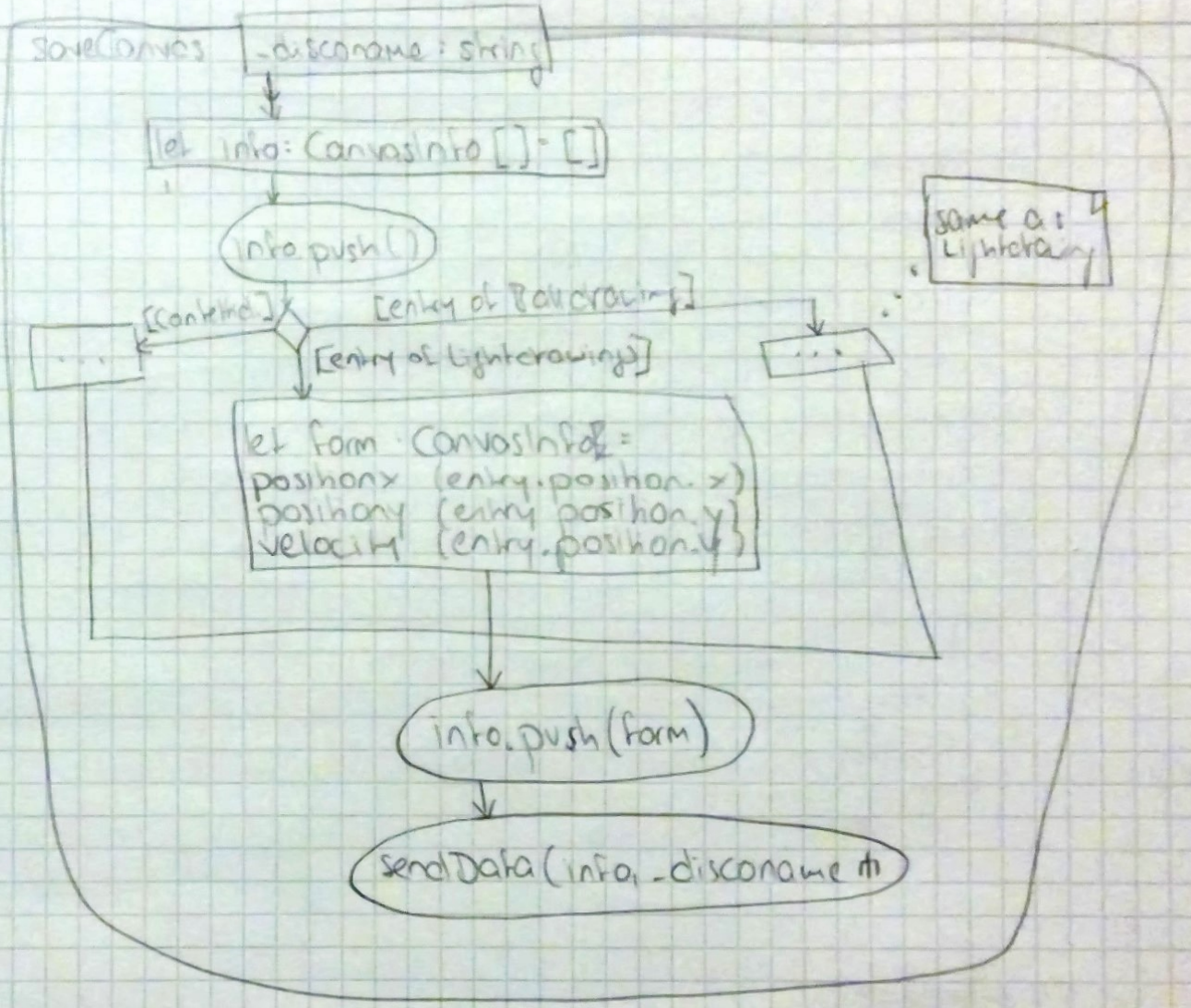

pickObject
_event: MouseEvent
dragDrop = true
let mousePosY
let mousePosX
offset X: number
offset Y: number
figurdrawings
if Schleife

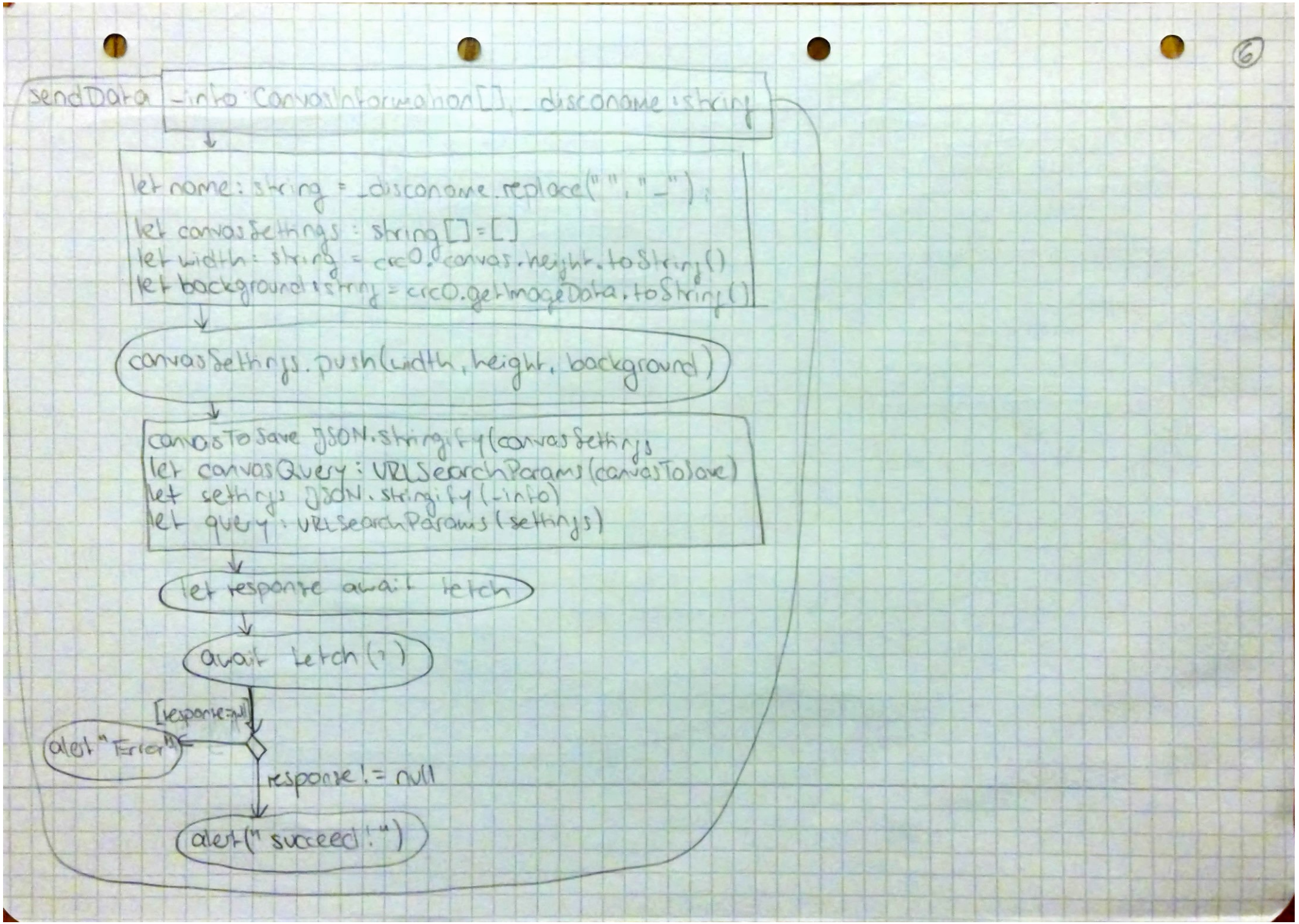

dragObject
_event: MouseEvent
dragDrop = true


placeObject
_event: MouseEvent
dragDrop = true
dragDrop = false
drawings.push(objectDragDrop)


saveName
_event : Event
let disconame : String
disconame = prompt("...")
disconame != null
saveCanvas(disconame)

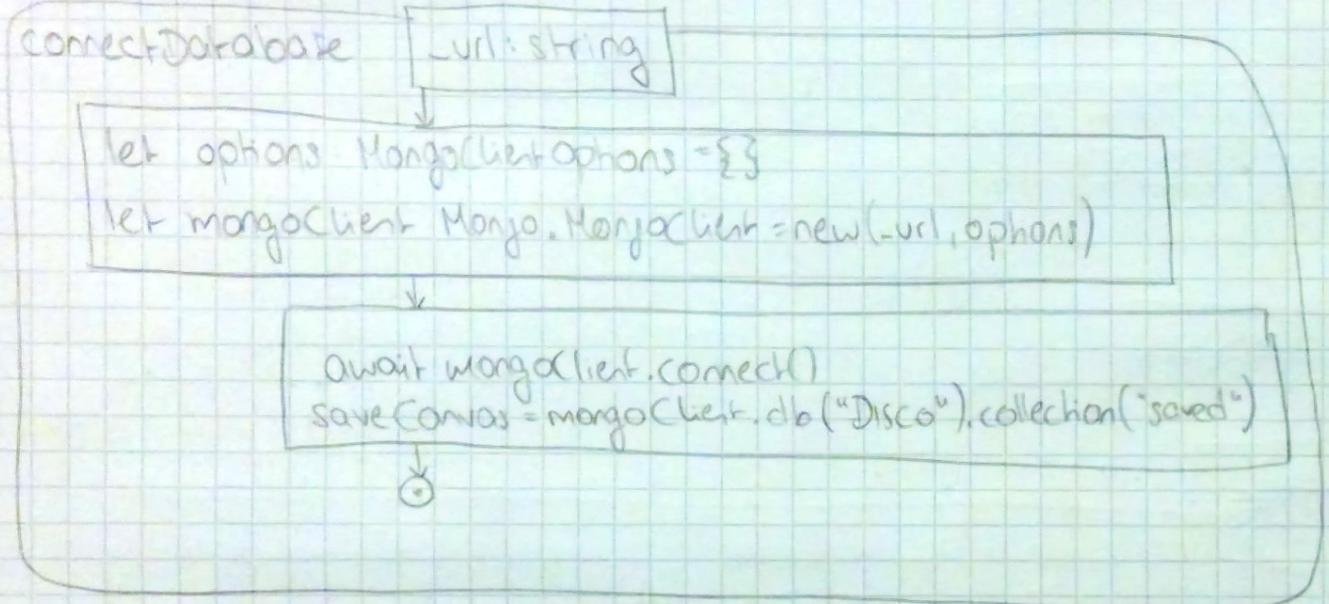
~~send~~

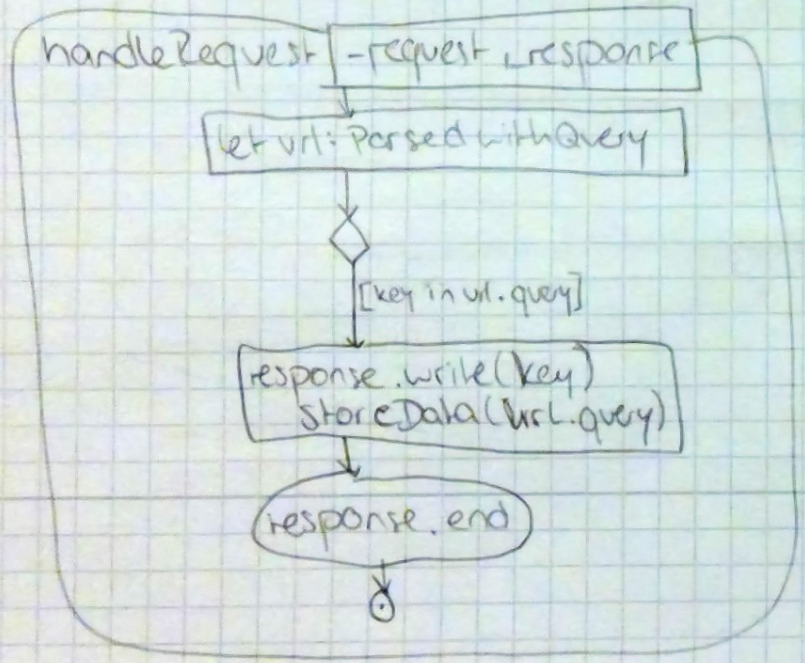

saveCanvas
_disconame : String
let info: CanvasInfo [] = []
info.push()
[contains]
[entry of Raincrawlers]
[entry of Lightcrawlers]
...
...
let form: CanvasInfo{} =
positionX (entry.position.x)
positionY (entry.position.y)
velocity (entry.position.y)
info.push(form)
sendData(info, _disconame)
some a : Lightcrawler

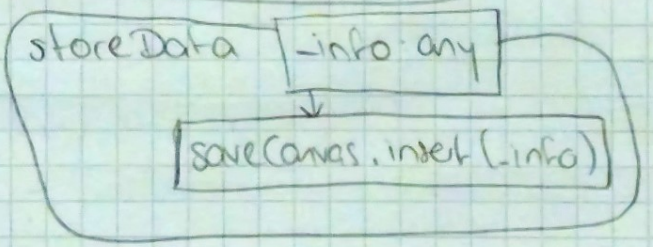

sendData(_info: CanvasInformation[], _discoName: string)
let name: string = _discoName.replace(" ", "_");
let canvasSettings: string[] = []
let width: string = crc0.canvas.height.toString()
let background: string = crc0.getImageData.toString()
canvasSettings.push(width, height, background)
canvasToSave JSON.stringify(canvasSettings
let canvasQuery: URLSearchParams(canvasToSave)
let settings JSON.stringify(_info)
let query: URLSearchParams(settings)
let response await fetch
await fetch(?)
[response=null]
alert "Error"
response != null
alert("succeed!")

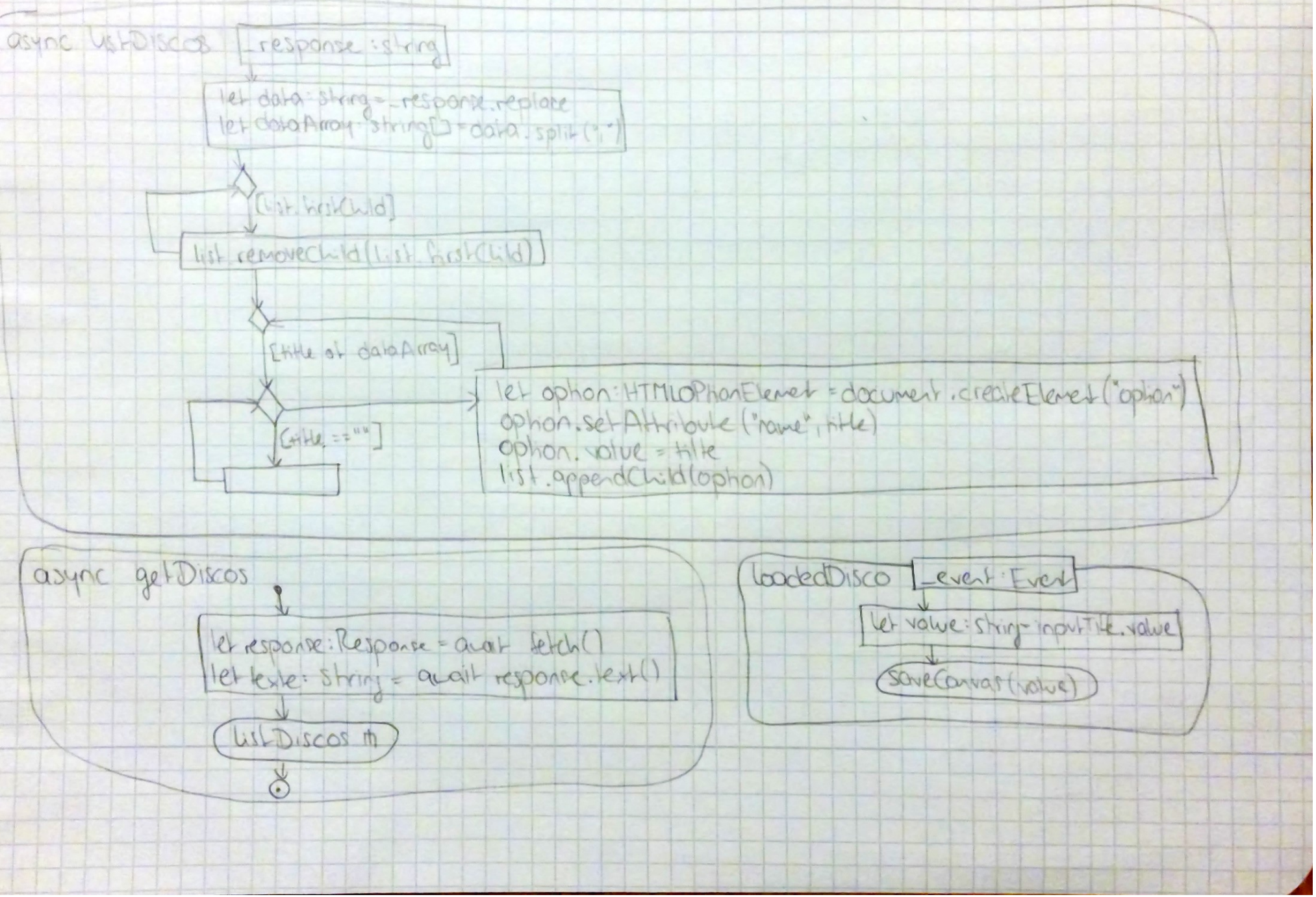
Server.ts

```
let serve: Http.Serve = Http.createServer()
let saveCanvas: Mongo.Collection
let port: number | string | undefined
  = process.env.PORT
let databaseURL: string = ""
serve.listen
serve.addEventListener(request,
handleRequest())
connectDataBase(databaseURL)
```








async listDiscos
_response : string
let data : string = _response.replace
let dataArray : string[] = data.split(",")
[list.hasChild]
list.removeChild(list.firstChild)
[title of dataArray]
[title == ""]
let option : HTMLOptionElement = document.createElement("option")
option.setAttribute("name", title)
option.value = title
list.appendChild(option)
async getDiscos
let response : Response = await fetch()
let texte : string = await response.text()
listDiscos()
loadedDisco
_event : Event
let value : string = inputTitle.value
saveCanvas(value)