

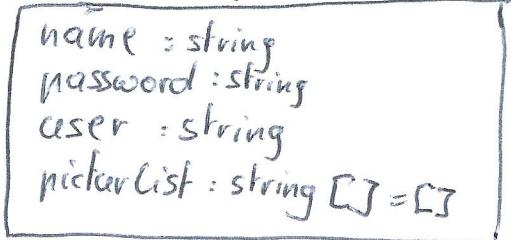
Datenbank:

→ Datenmodelle

⇒ asso. Arrays ~~siehe infotyps.ts~~

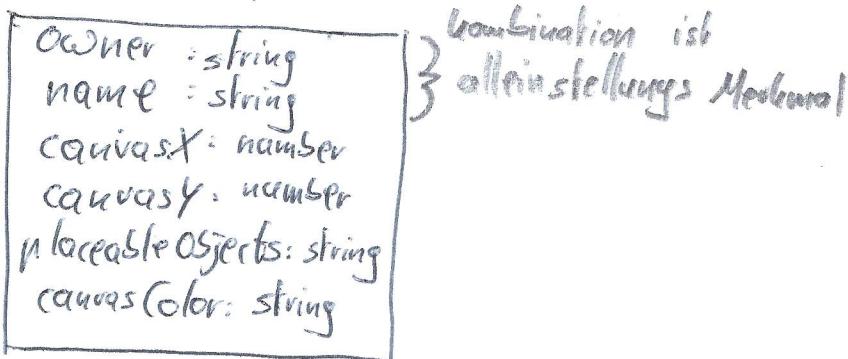
collection Userdatabase

⇒ UserData



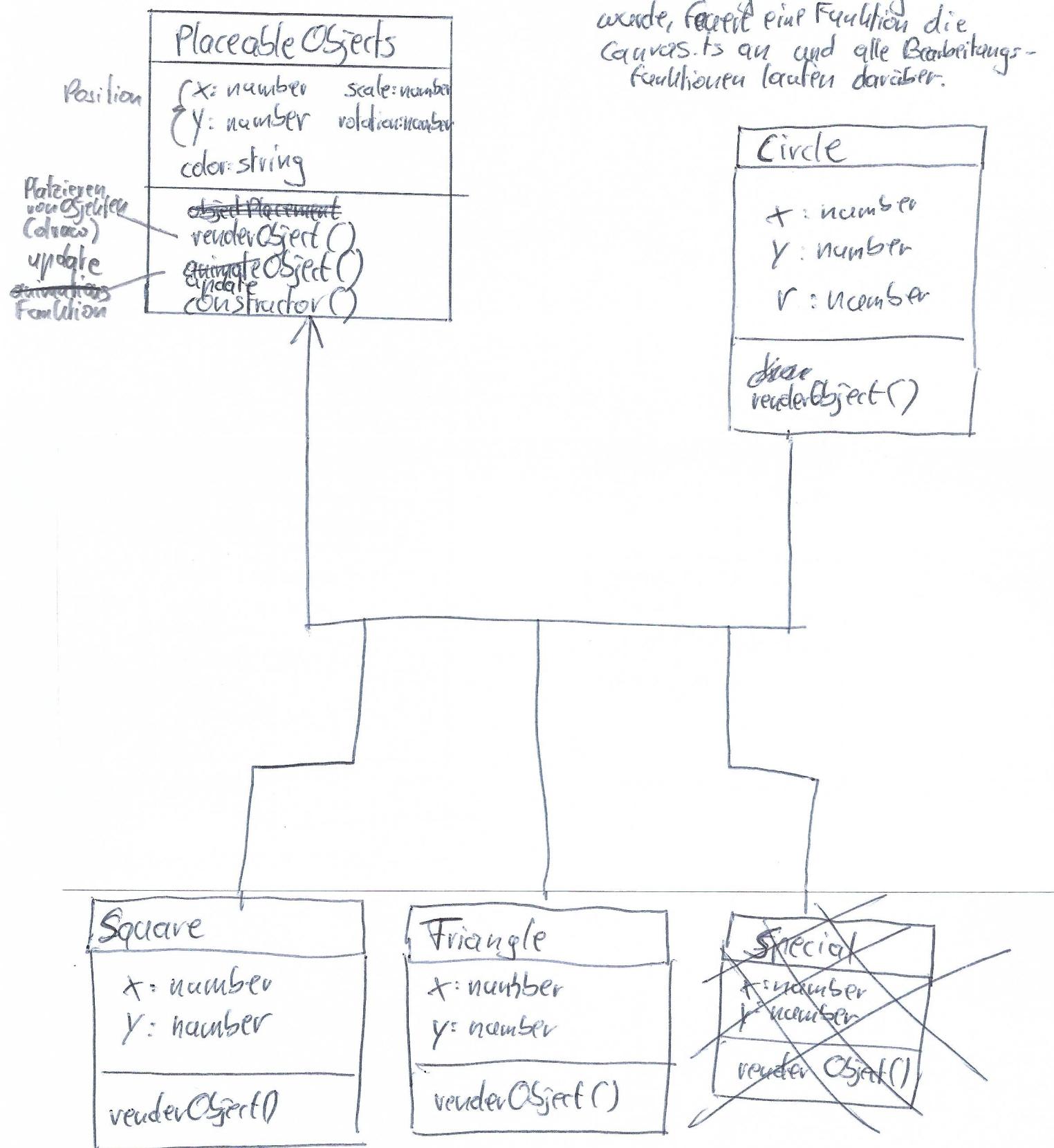
collection canvasDatabase

⇒ Canvas Data



Die Classes von Radle MS Paint

→ canvas.ts dabei auch in zusätzliches File
 ↳ nach dem die Grandcanvas geladen
 wurde, fügt es Funktion die
 canvas.ts an und alle Bearbeitungs-
 funktionen laufen darüber.

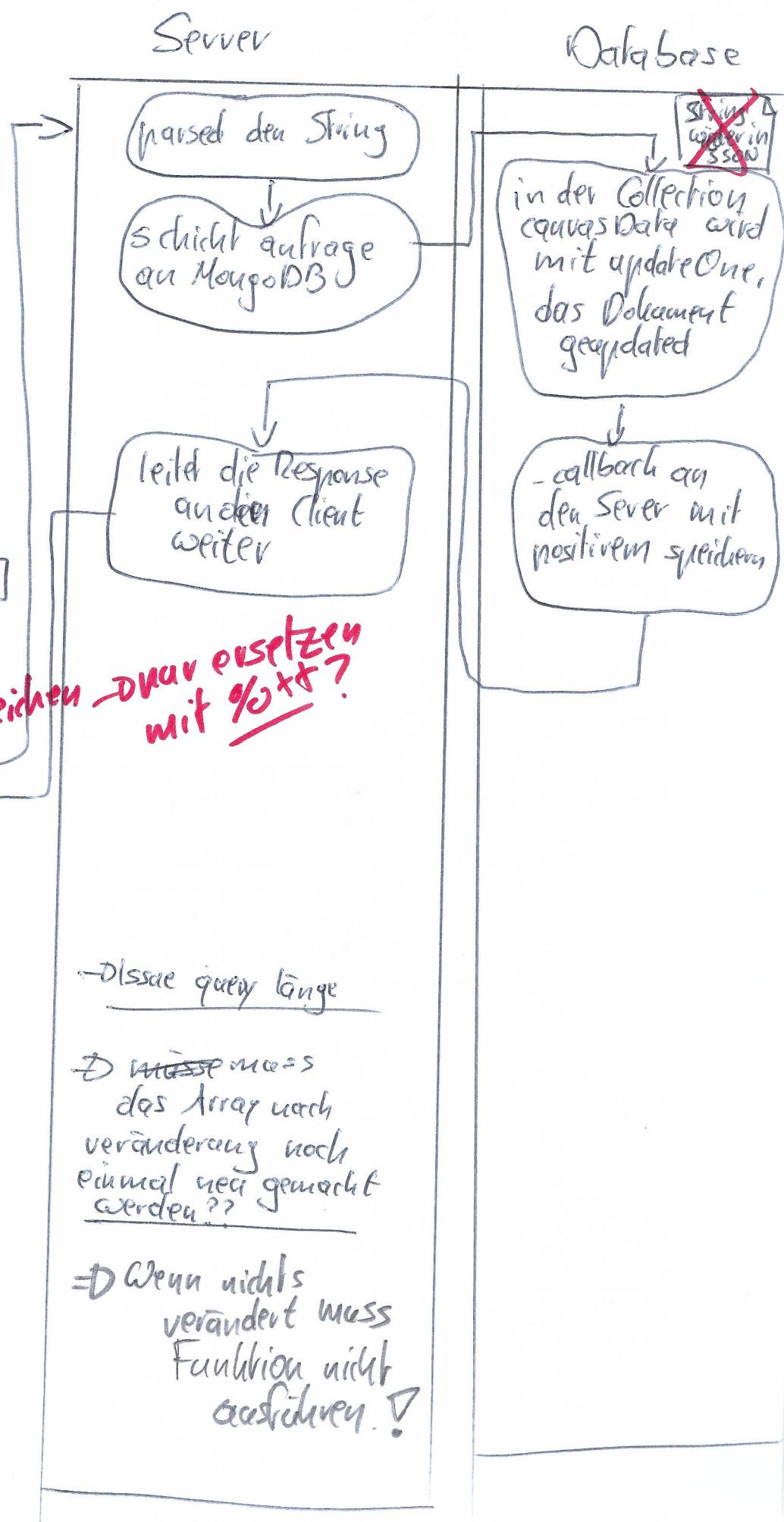
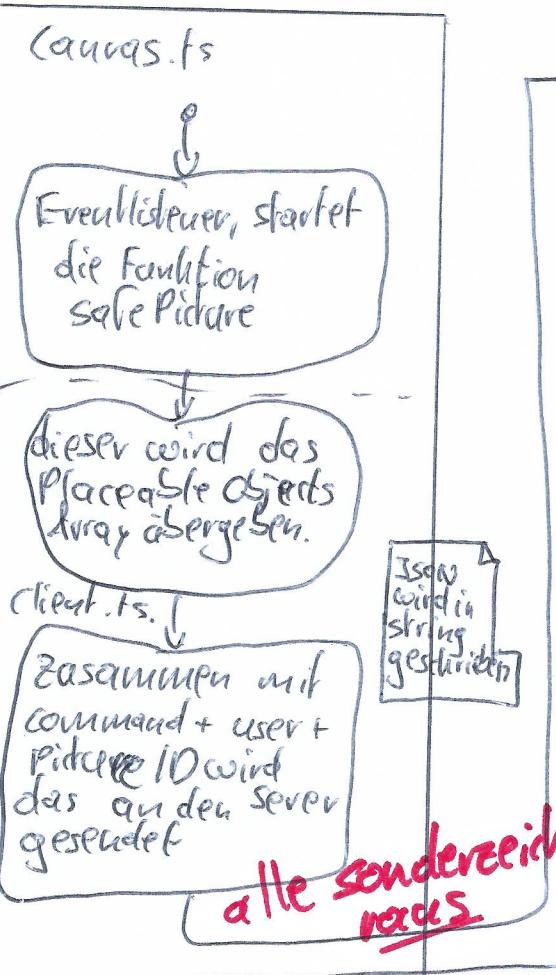


Working Title
 → alle Sachen darstellen

lassen
 → dann Clasesstruktur anwenden

Bild speichern

Client



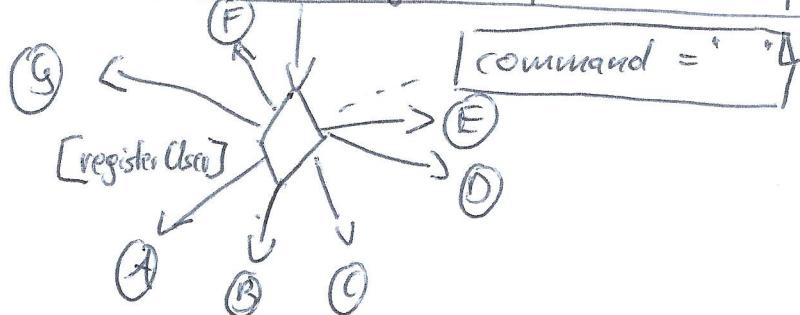
Server seitige Verarbeitung

handlerequest

> Listener request

query Ctrl wird in die einzelnen Schnipsel gepräst

let command = query["command"]



- user = query["username"]
- pic = query["pictureName"]

[registerUser]

(A)

let NewUser: UserData

queries an die einzelnen Array stellen schreiben

DB.registerUserName("username", findCallback, NewUser)

(G) [loadSelectedPicture]

DB.loadPictureFromDBC
findCallback, query["username"], query["pictureName"]

[loginUser]

(B)

password: string

DB.loginUser("username", password, findCallback)

[Load PictureList]

(C)

DB.loadListFromDB("username", findCallback)

CanvasData: CanvasData

queries in das Array schreiben

DB.pushPictureToDB(CanvasData, findCallback)

[InitiatePicture] [deletePicture]

(D)

queryString eulgepar uehmen

CanvasData: CanvasData

(E)

DB.deletePictureFromDB("username", picName, findCallback)

(F)

user: string
- pic: string

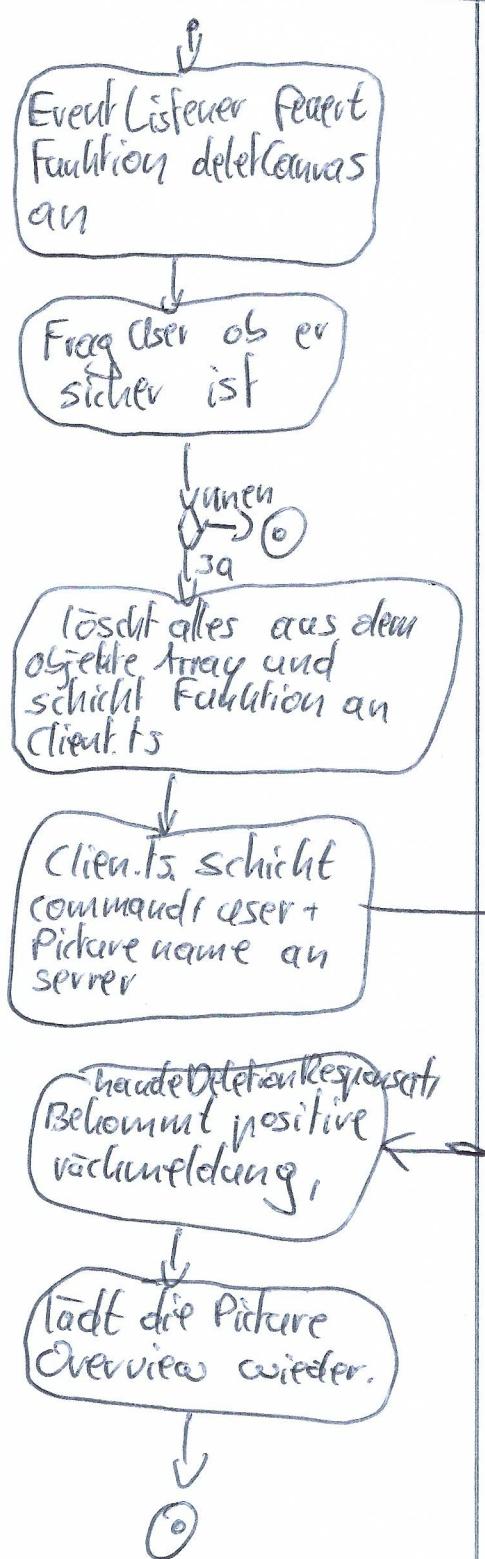
DB.safePictureToDBC(picName, findCallback)

- user, - pic,
query["placeableObject"]

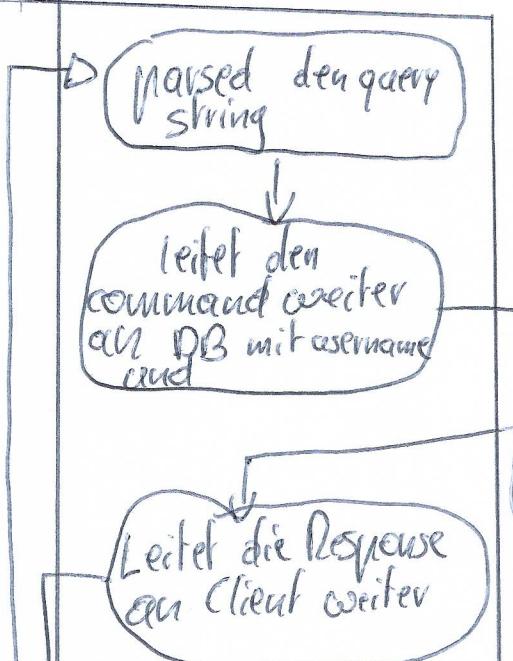
(O)

Bild löschen

Client



Server



Database

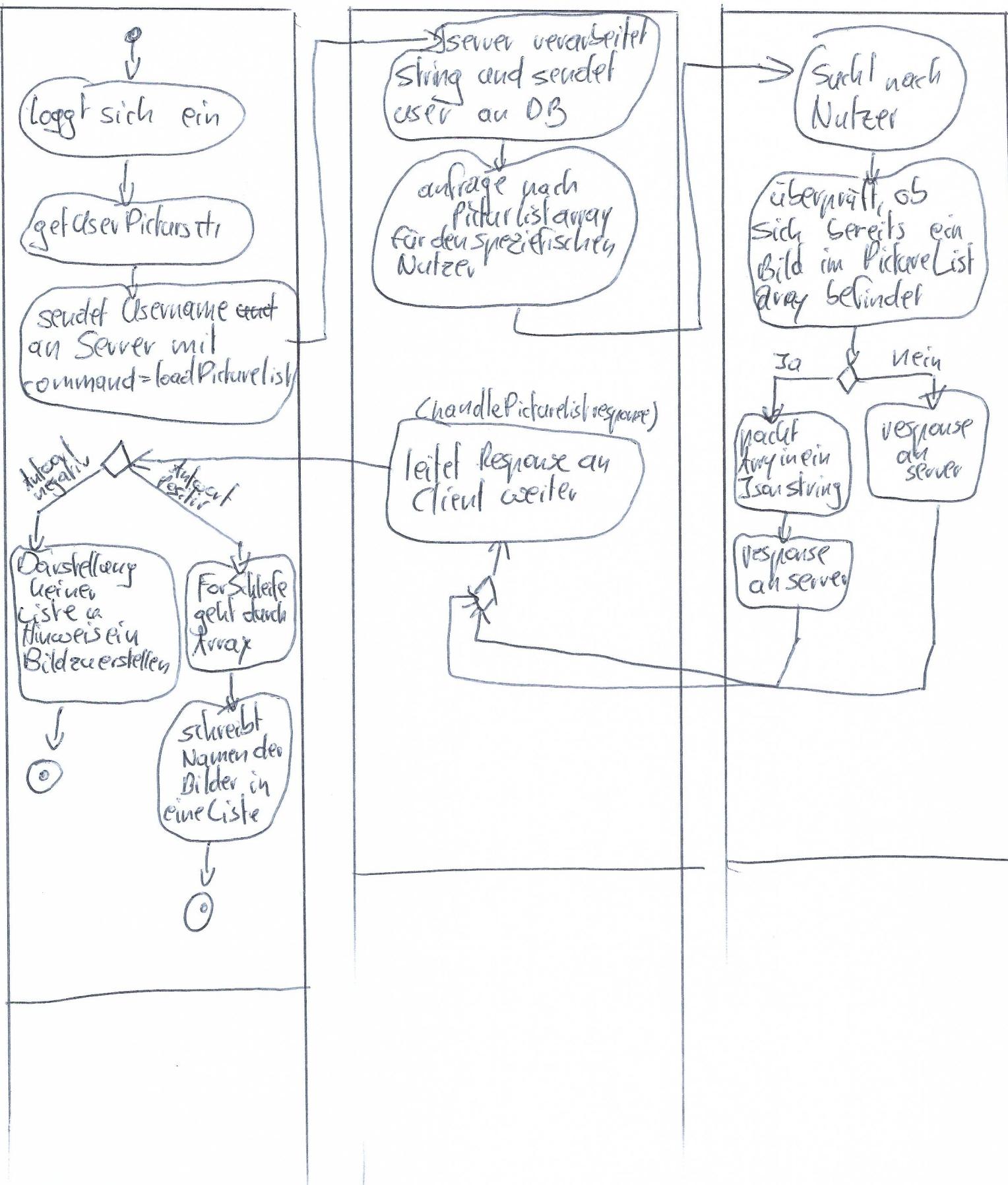


Bild übersichts Liste laden

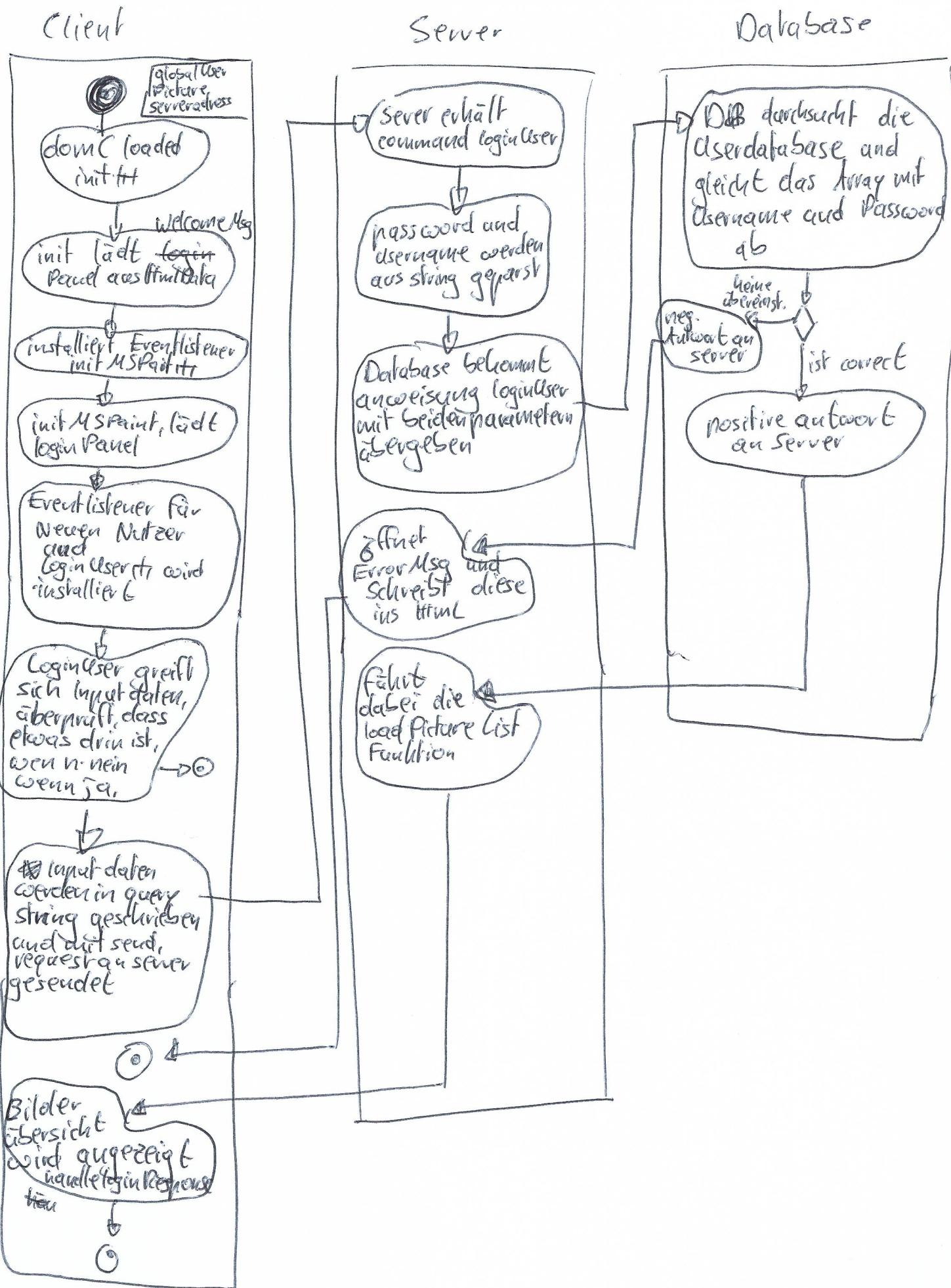
Client

Server

Database

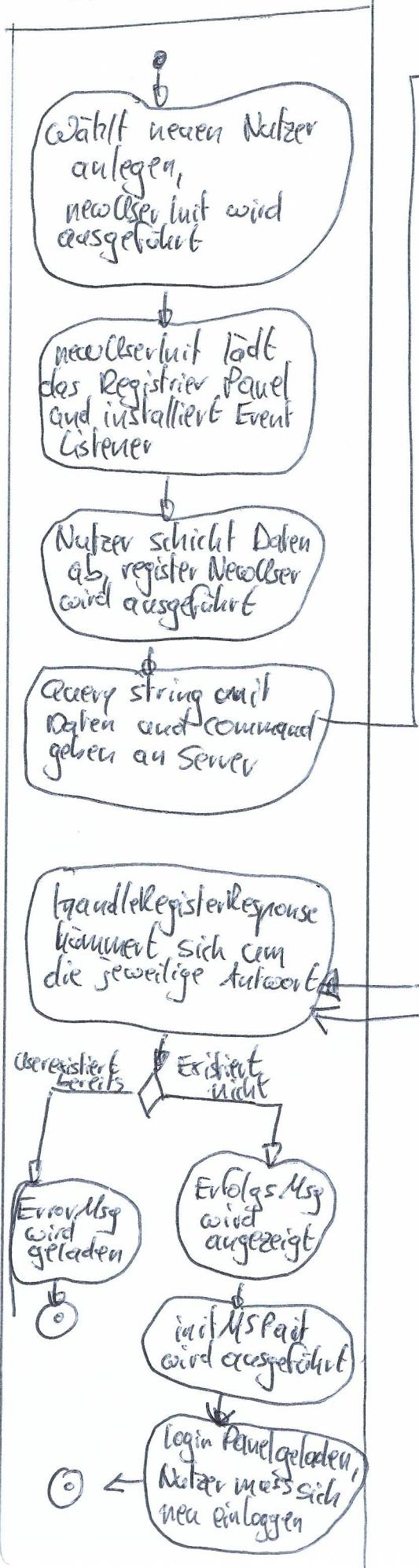


Login Funktion 1 Start

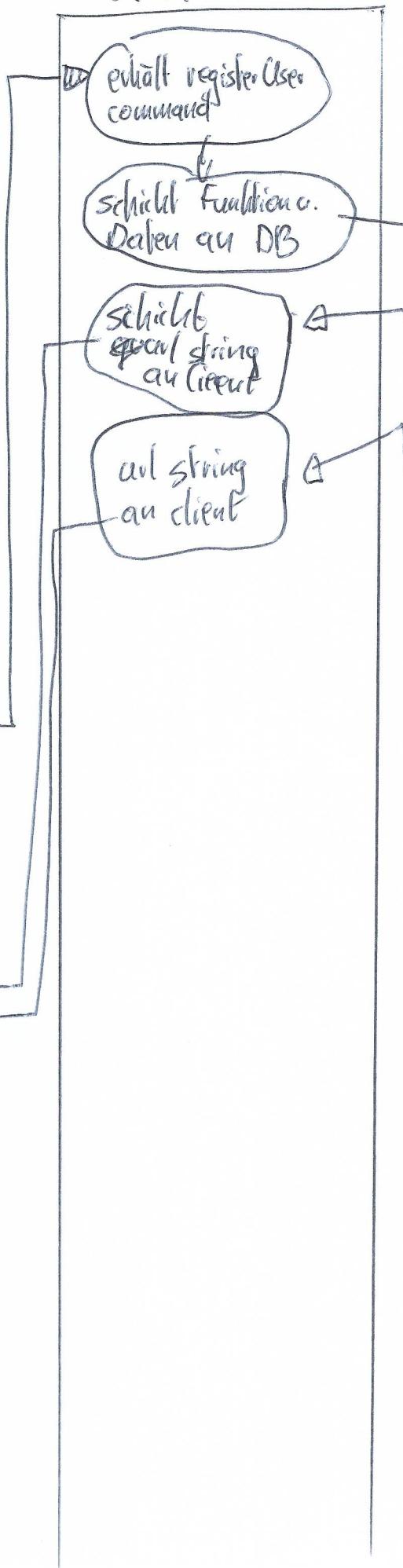


Neuen Nutzer registrieren

Client



Server



DB

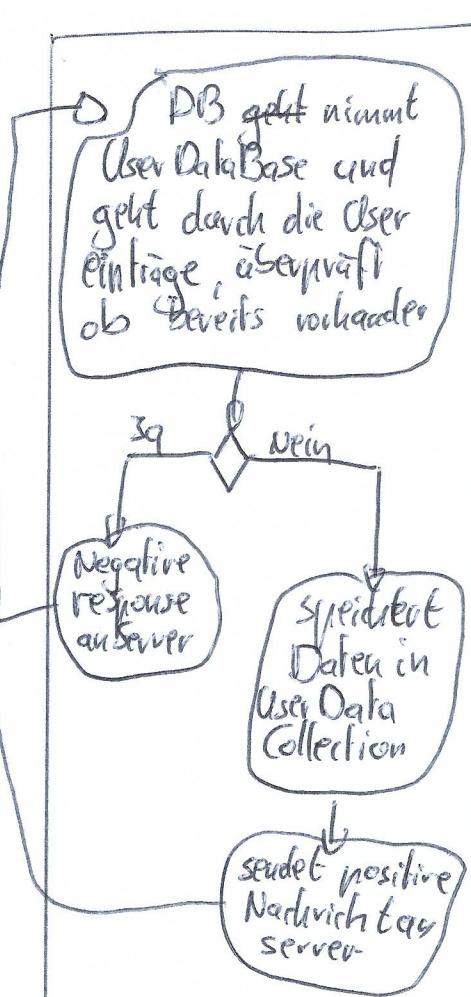


Bild anlegen / Canvas erstellen

Client

Server

Datenbank

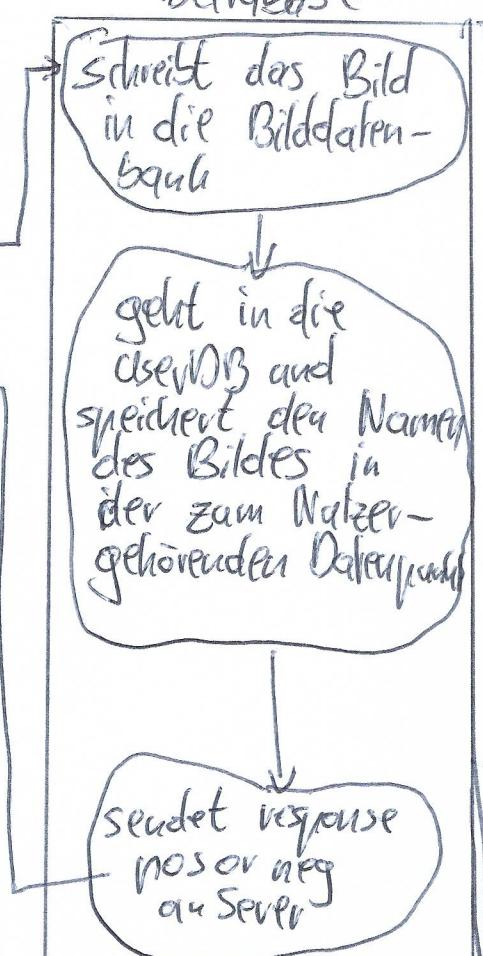
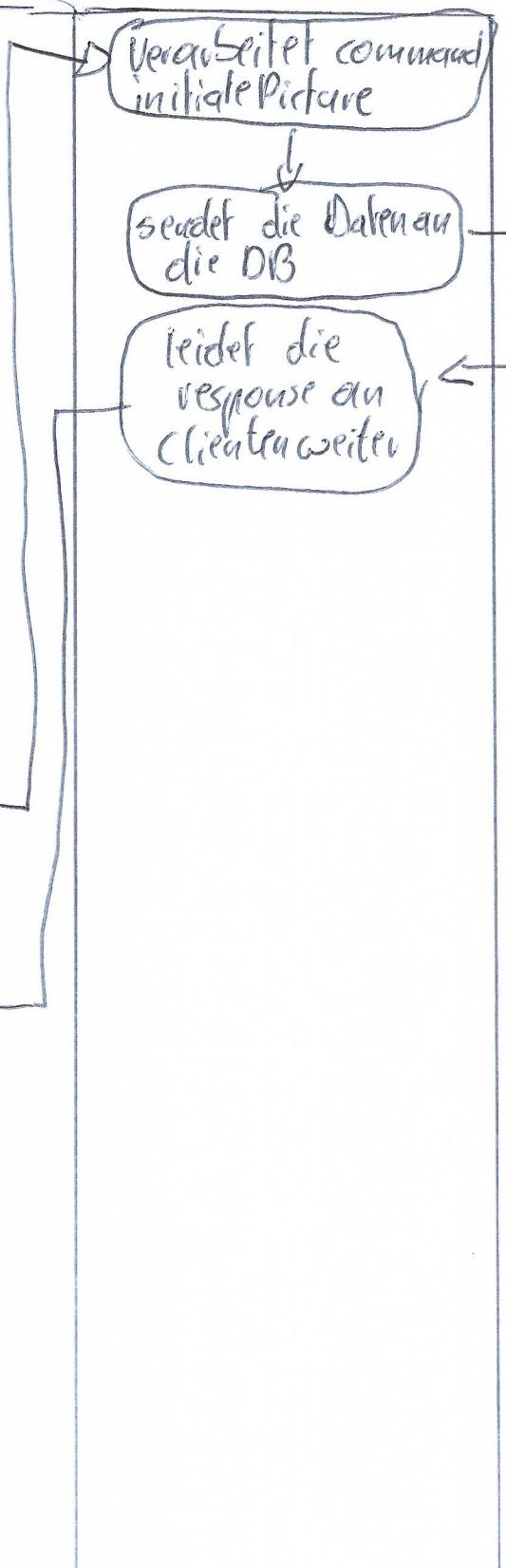
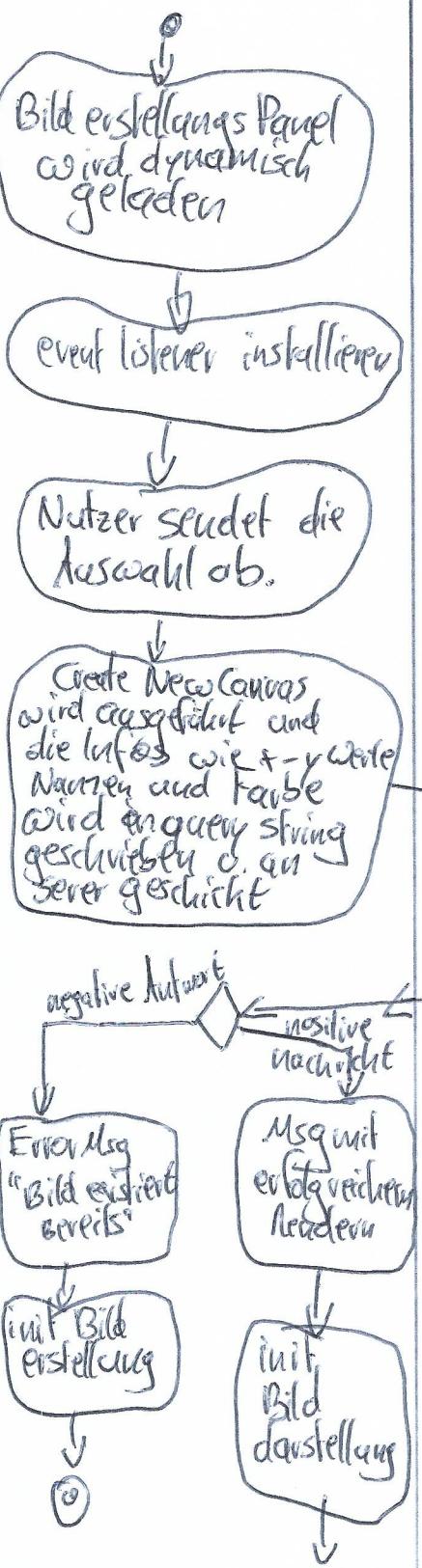


Bild & das erstmal/ erstellen und laden:

global Variables

> click: createNewCanvas

```
let globalUser: string  
let globalPictureName: string  
let canvasSizeX: number  
let canvasSizeY: number  
let canvasColor: string
```

```
let re: string = "#"  
let query: string = "command=initiatePicture"  
let inputs: HTMLCollection
```

```
let queryColor: string = inputs[3].value
```

```
queryColor = queryColor.replace(re, "%23")
```

restliche inputs mit den gewöhnlichen
query keys in der query schreiben
und die globalen Variablen

```
query += `&${"canvasColor"}=${queryColor}
```

Send Request Query, handle NewCanvas Response

New
handle Canvas Response

Wenn der Server "Save Positive" weitergeleitet
hat, wird initRenderCanvas geführt

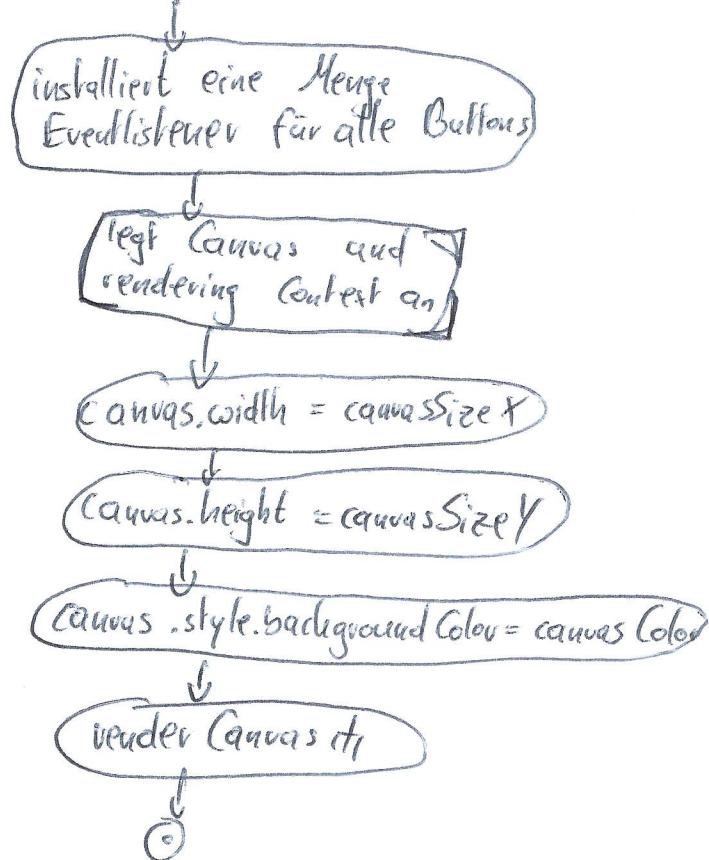
initRenderNewCanvas

innerHTML wird ersetzt durch htmlData["mainCanvasPanel"]

initCanvas()

in der
canvas.js

init Canvas



delete Canvas

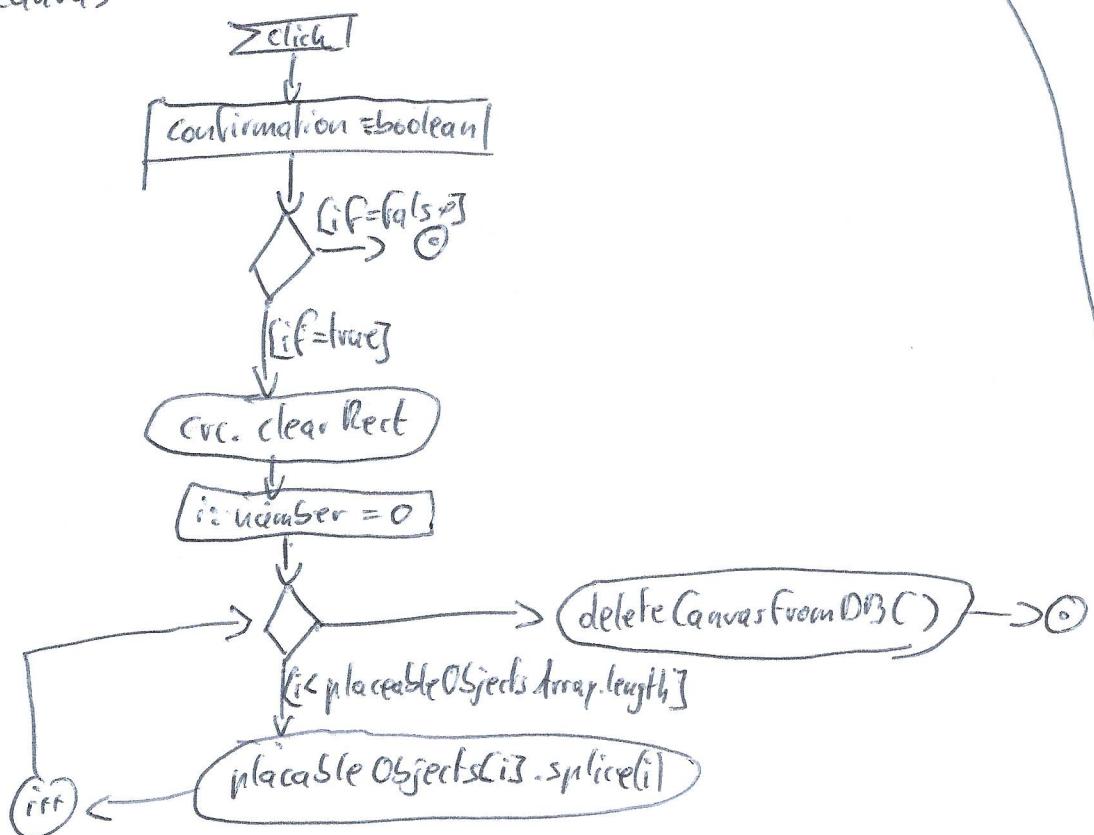
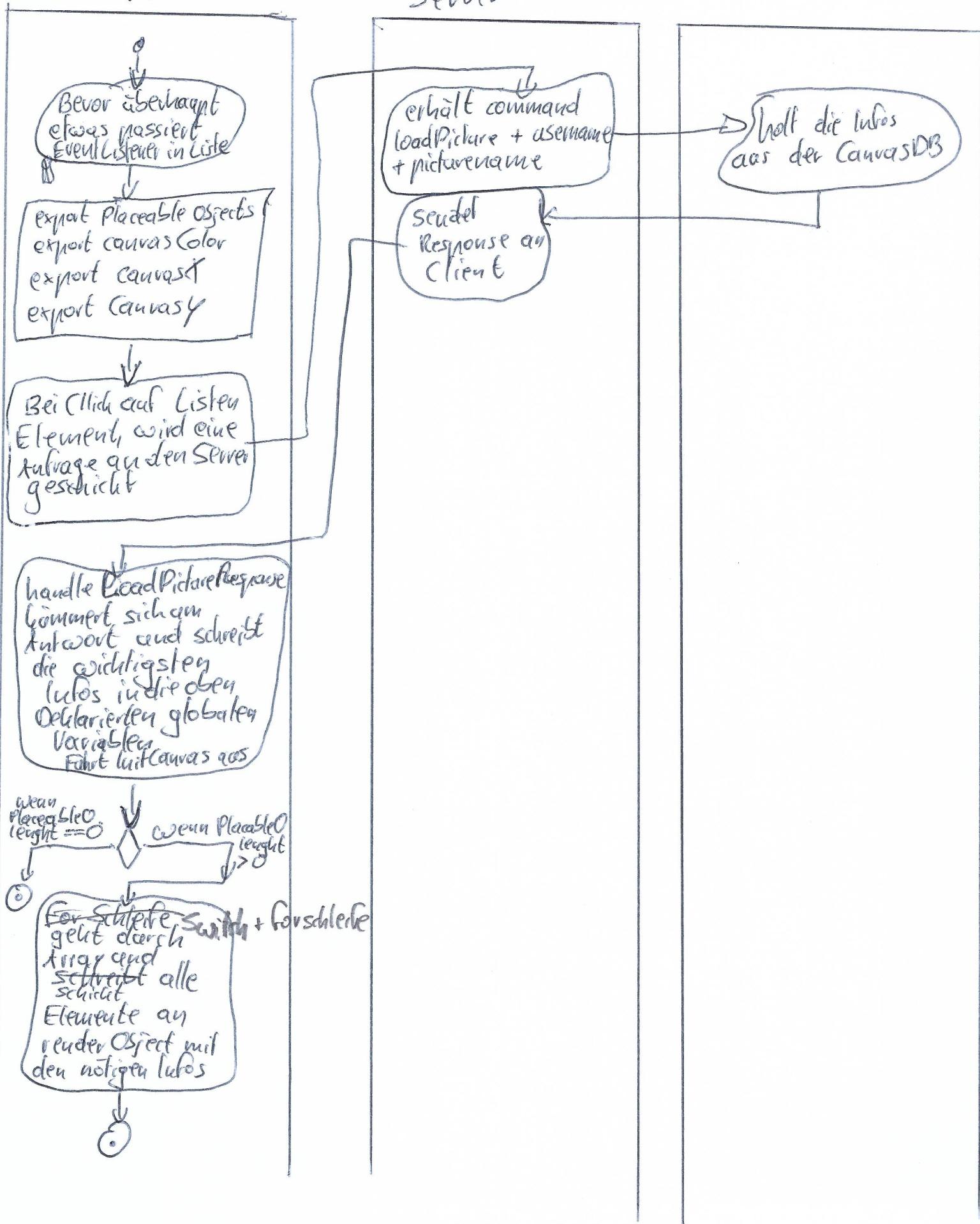


Bild erneut bearbeiten:
⇒ woher weiß der Button, welche ID gedrückt wurde?

Client

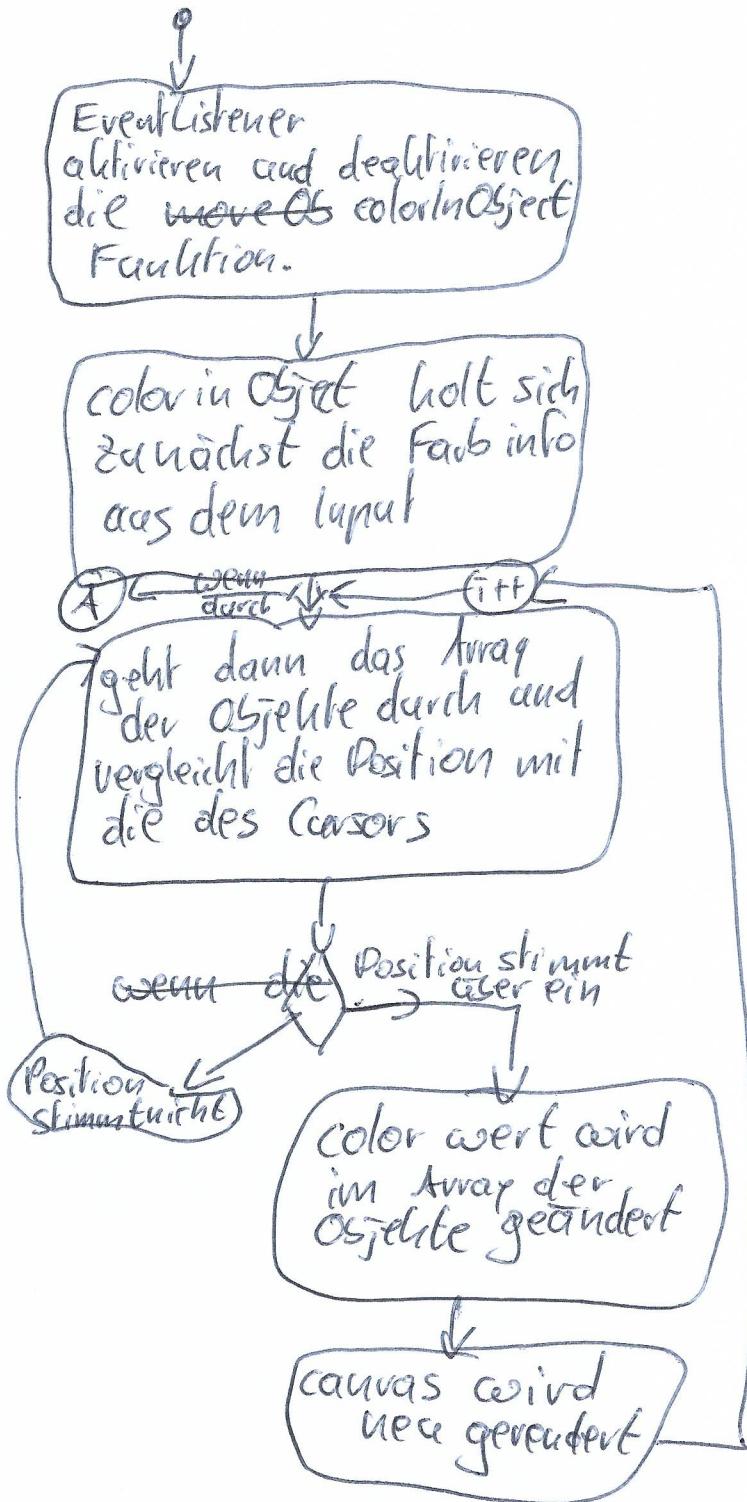
Server

Database



Objekte Färben:

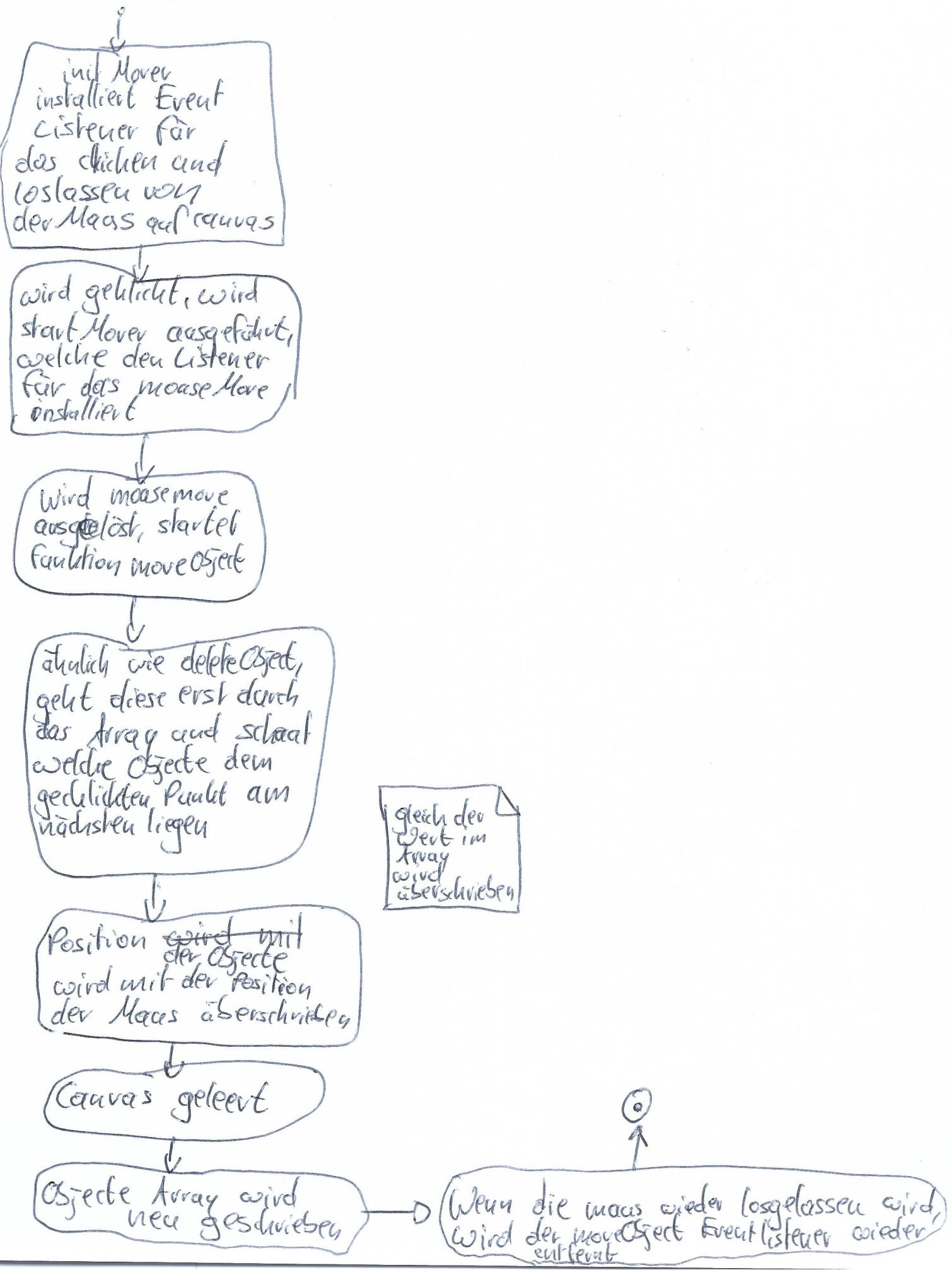
⇒ im großen und ganzen die Bewegen Funktion



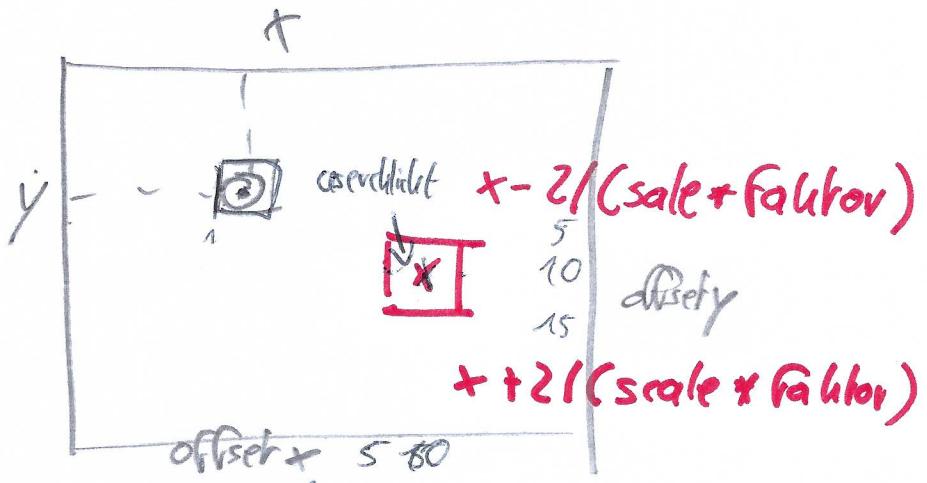
Änderung des Array
⇒ selber Anatz,
nur wird die
type Variable im
Objekte Array um
geschrieben.

Bewegen von Objekten

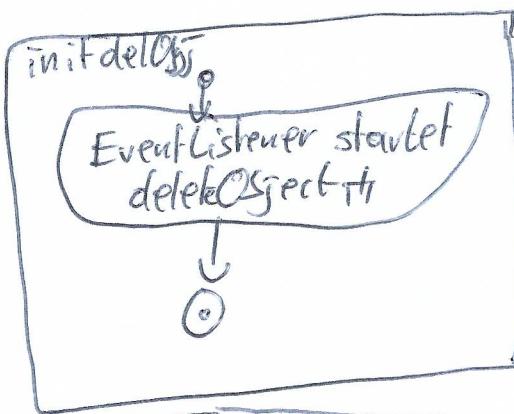
→ Grund daran kommt vom Löschen



Objekt von Canvas löschen.

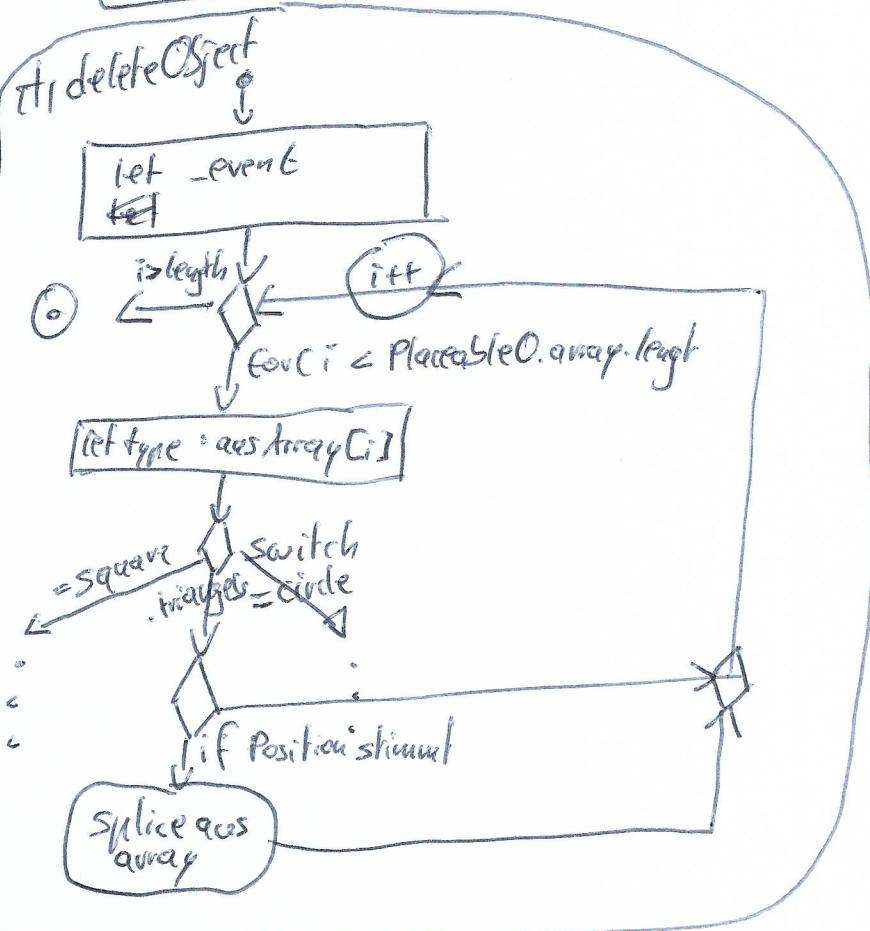


Für x : $\frac{\text{array}.x - \text{scale} * \text{faktor}}{\text{array}.x + \text{scale} * \text{faktor}} \geq \text{offset}$ \wedge
 $\text{array}.x + \text{scale} * \text{faktor} \leq \text{offset}$
⋮



→ in if clauses berechnet es komischerweise falsch
⇒ Lösung zu nächst & in Variablen berechnen

iFsizeX
iFsizeYm
iFsizeYf
iFsizeYm



z.B.

X 54 688

assert = 55

+ 62

X 53

Bild animieren lassen

Bild Animation pausieren

canvas.ts

canvas.ts

Event-Listener für Motor Animer
Button

Funktion updateObject()
Fügt feuer ~~an~~ e in einer
Forschleife die einzelnen
Objekt updatet für
jedes Objekt im
Object Array

abfrage wo sich der
Animation Counter befindet

ist 1
Stop Animation

ist 2
Start Animation

geht mit
window.setTimeout(handler, time, true/false)

→ das Bild muss
nach Bewegung
neu gerendert
werden
dafür muss
das Objekt Array
aktualisiert werden.

First draft