# Praktikum XML-Technologie: Final Project

Team WebSocket (ws):

Achraf Aroua

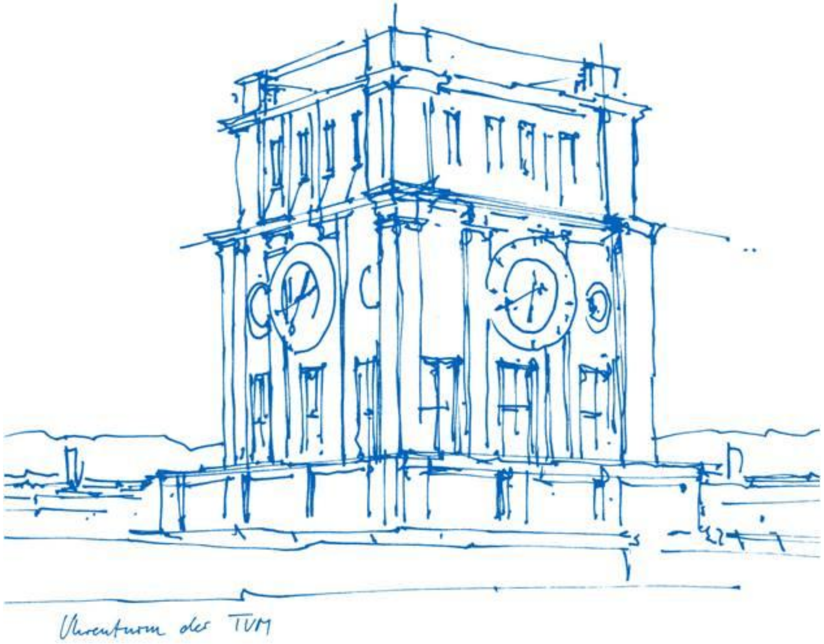
Ali Rabeh

Mariia Borysova

Yousri Cherif

Technical University of Munich

Munich, 28.04.2020



# Team WebSocket (ws)

Achraf Aroua
Ali Rabeh
Mariia Borysova
Yousri Cherif

# Table of Contents

- Introduction
- Architectural solutions
- Design
- Functionality
- Demo

# Introduction

## Requirements

- Lounge
- Multi-client solution (WebSocket)
- XML technologies only
- MVC architectural style

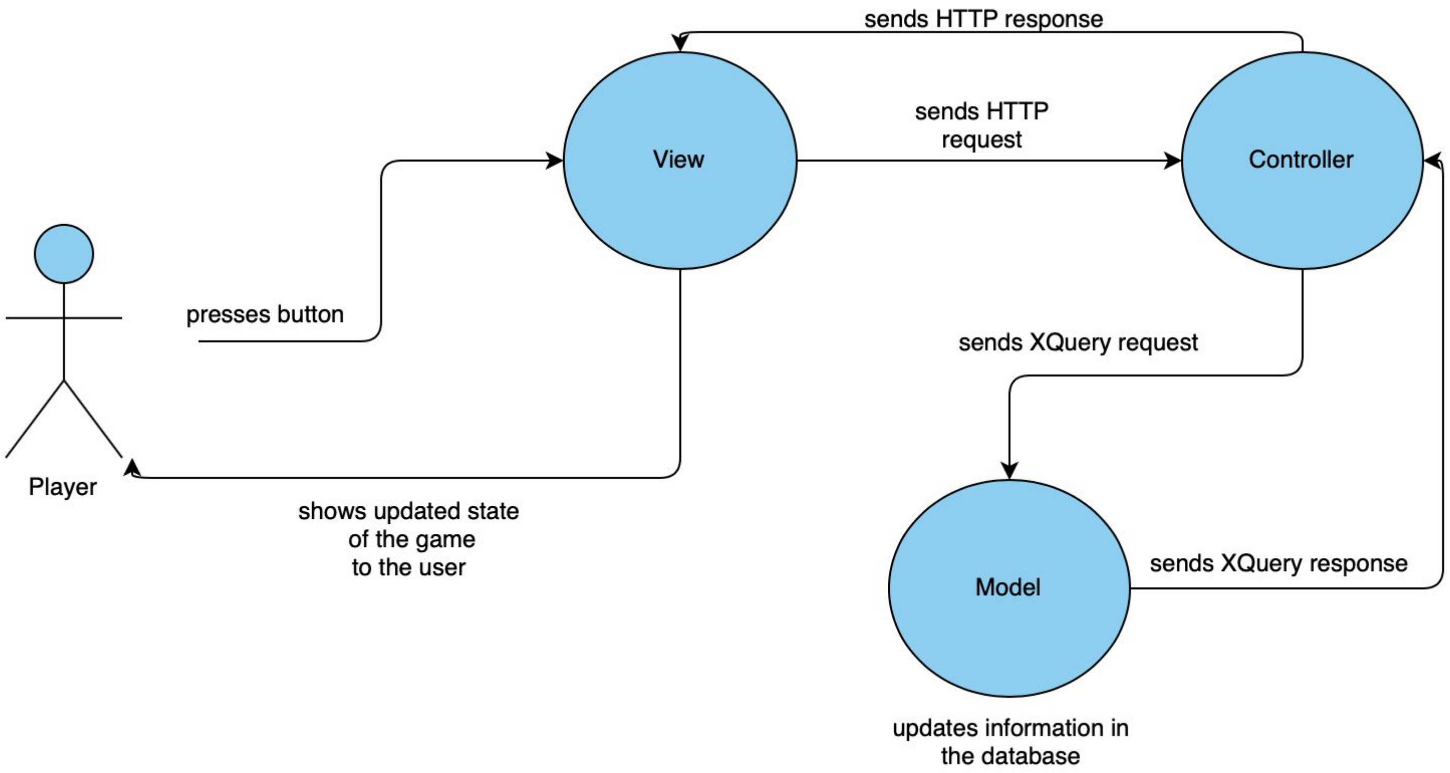
# Introduction

## BlackJack Rules

- Beat the dealer
- Any sum of 21 is blackjack
- Hit, Stand, Double, Surrender
- No Draws
- Player’s blackjack always wins 3:2

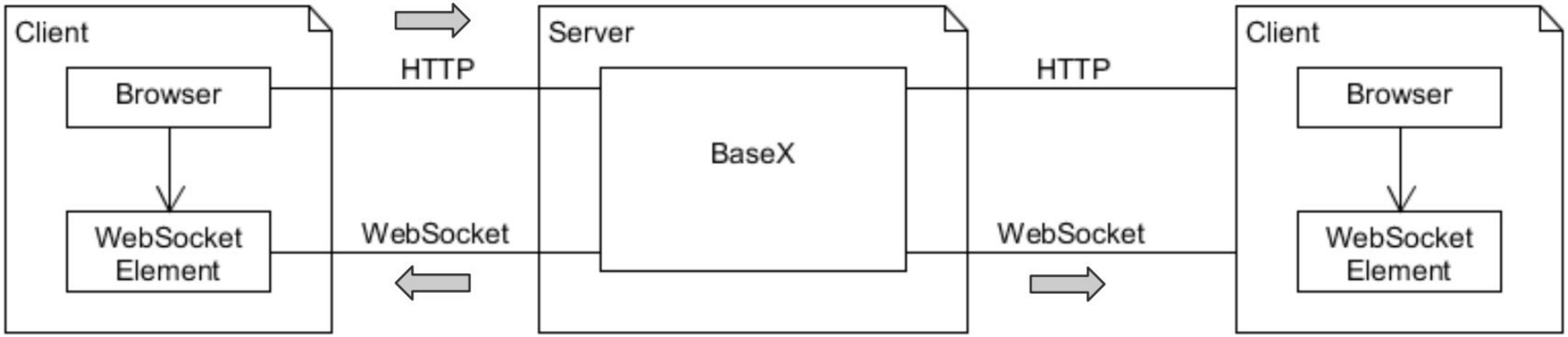
# Architectural solutions

## Model-View-Controller



# Architectural solutions

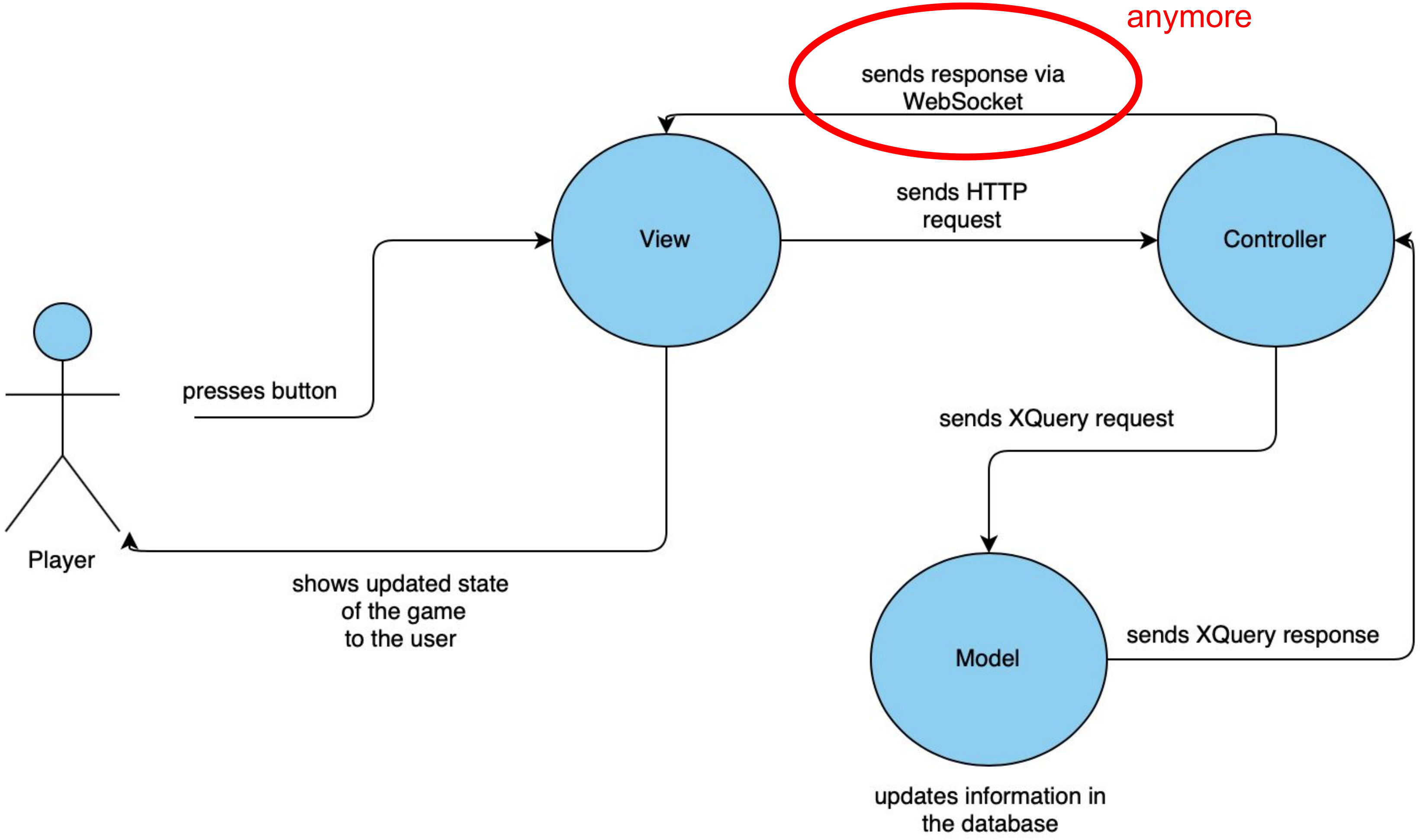
## WebSocket



*Source: “Multi-client Webanwendungen mit XML Technologien” by Philipp Ulrich*

# Architectural solutions

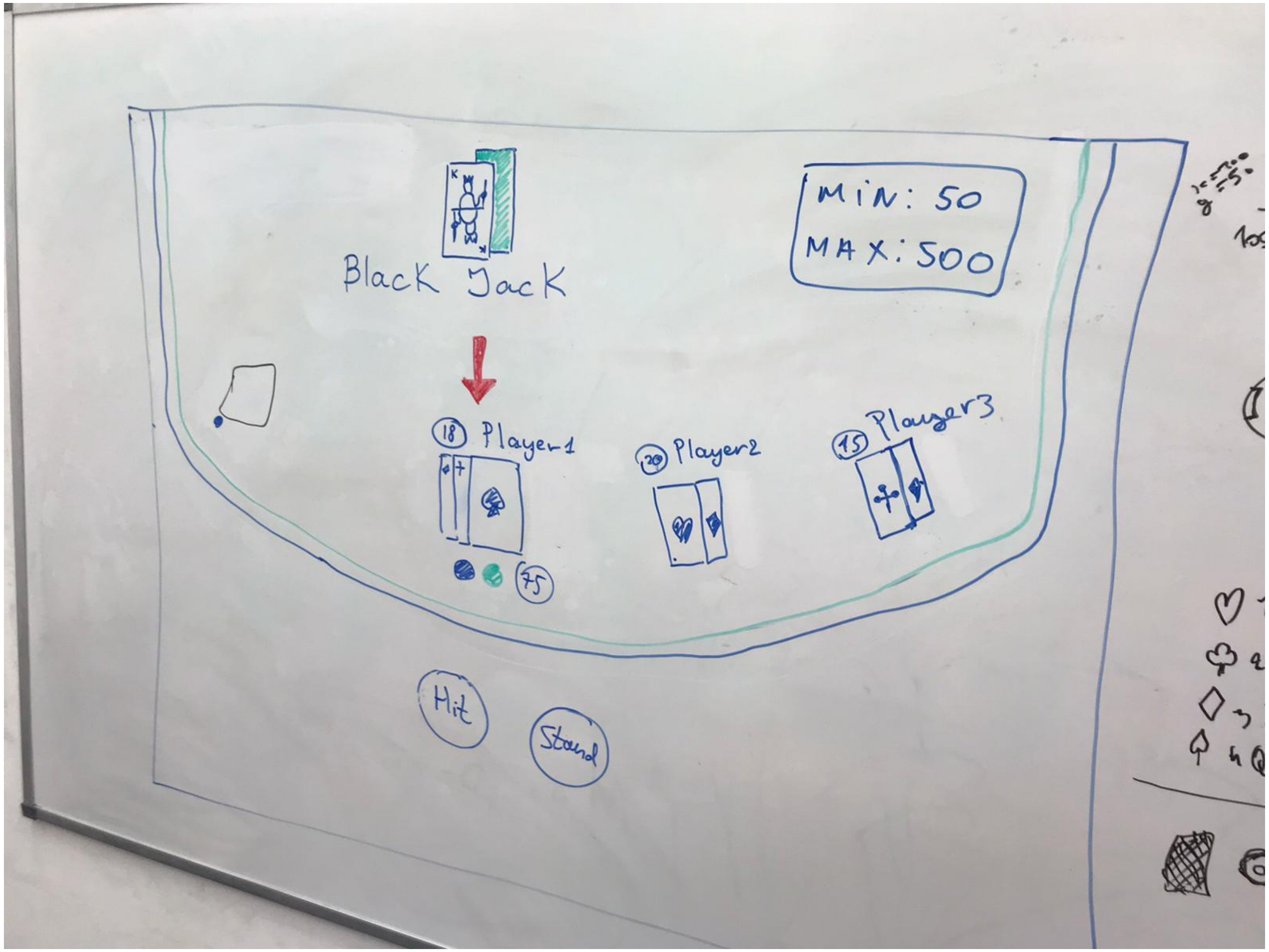
## WebSocket


sends response via WebSocket
View
sends HTTP request
Controller
presses button
sends XQuery request
Player
shows updated state of the game to the user
Model
sends XQuery response
updates information in the database

not an HTTP response anymore

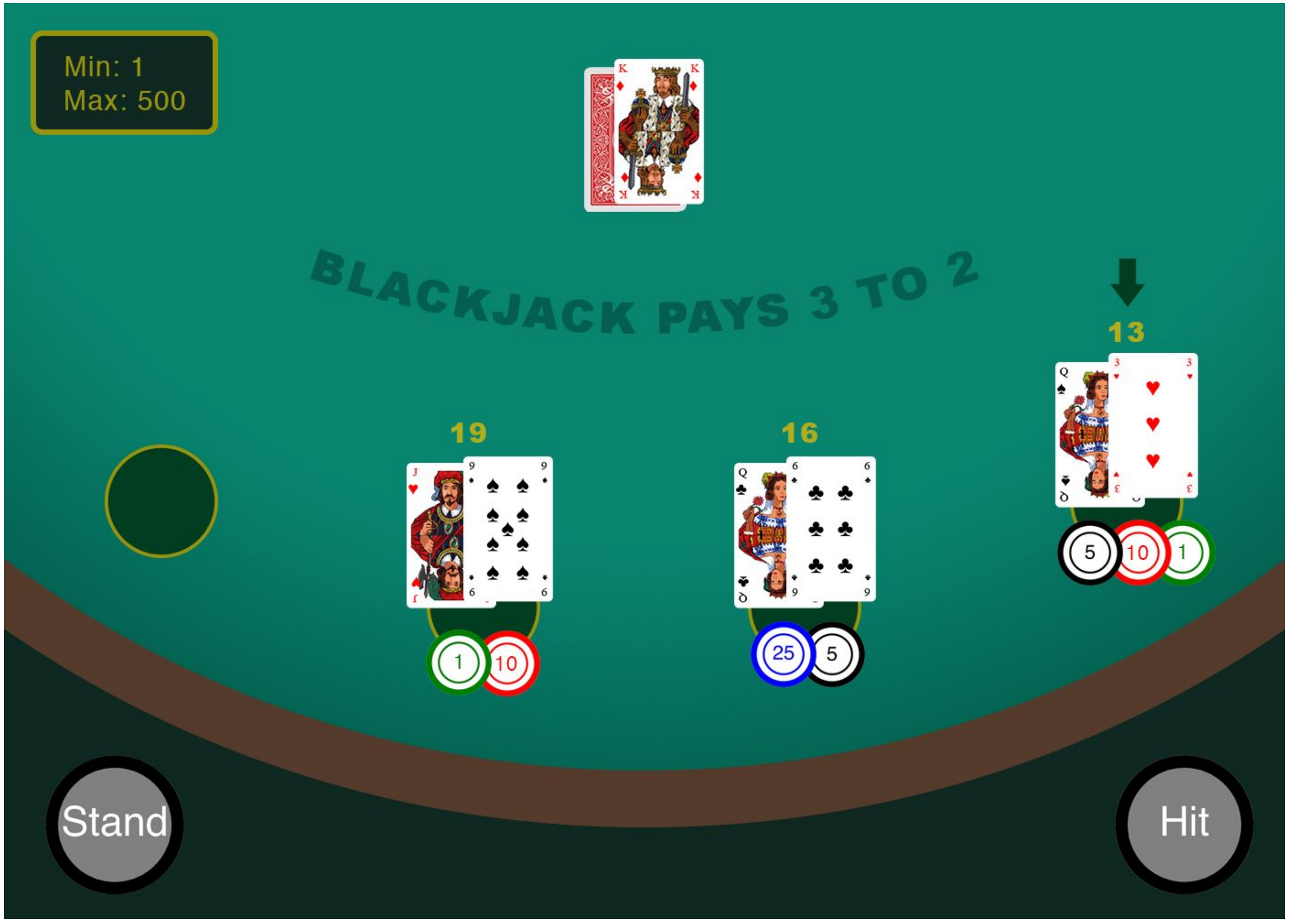
# Design

## Brainstorming



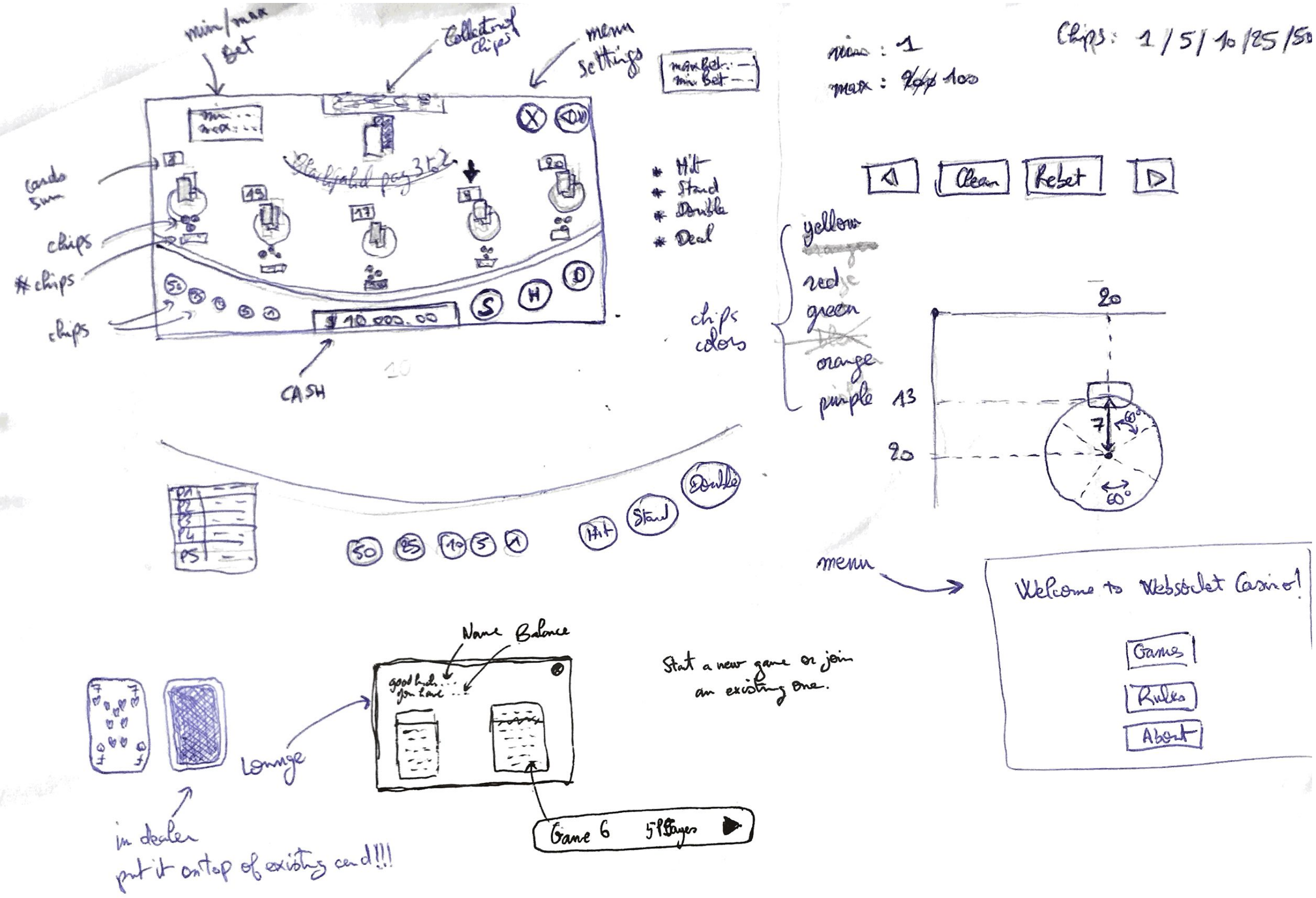
# Design

## Prototyping (Photoshop)



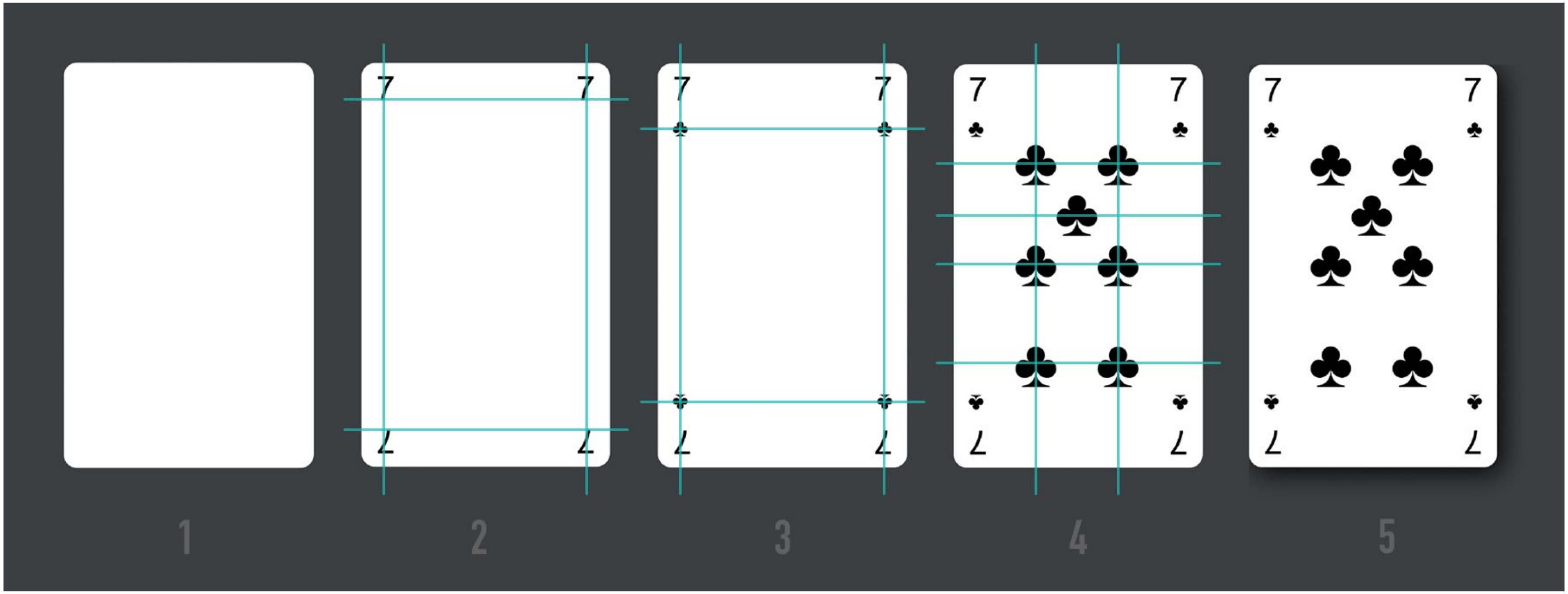
# Design

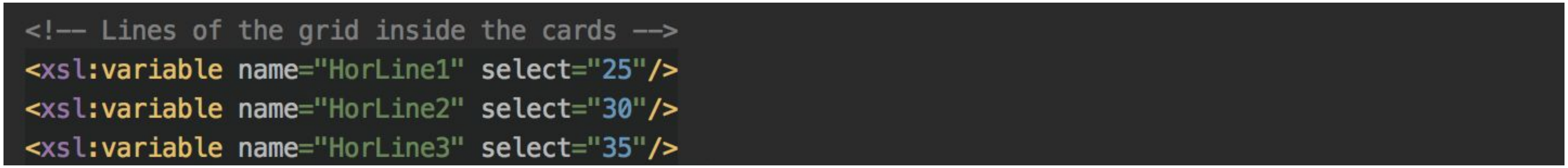
## Prototyping (hand sketches)



# Design

## Cards

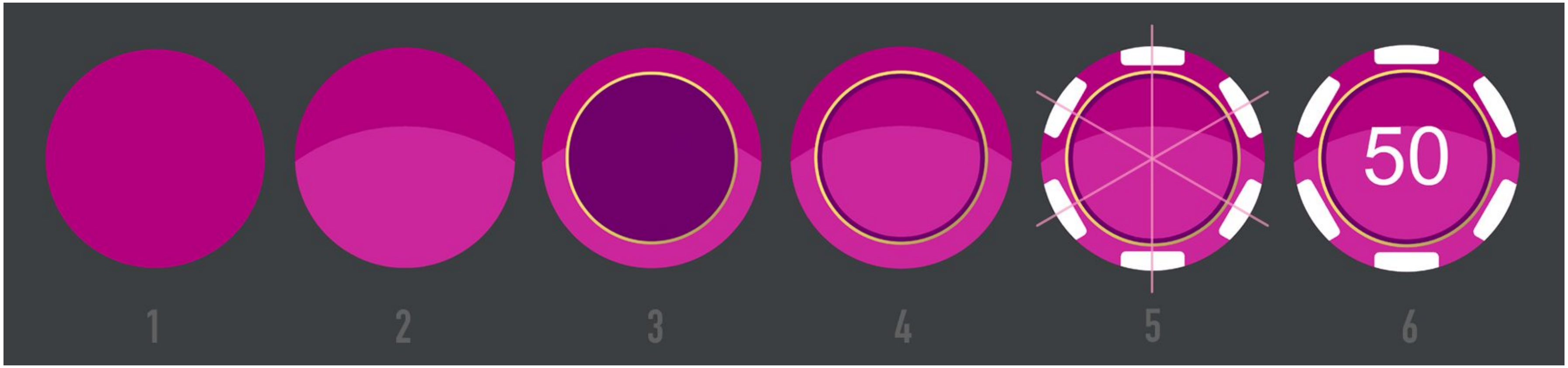


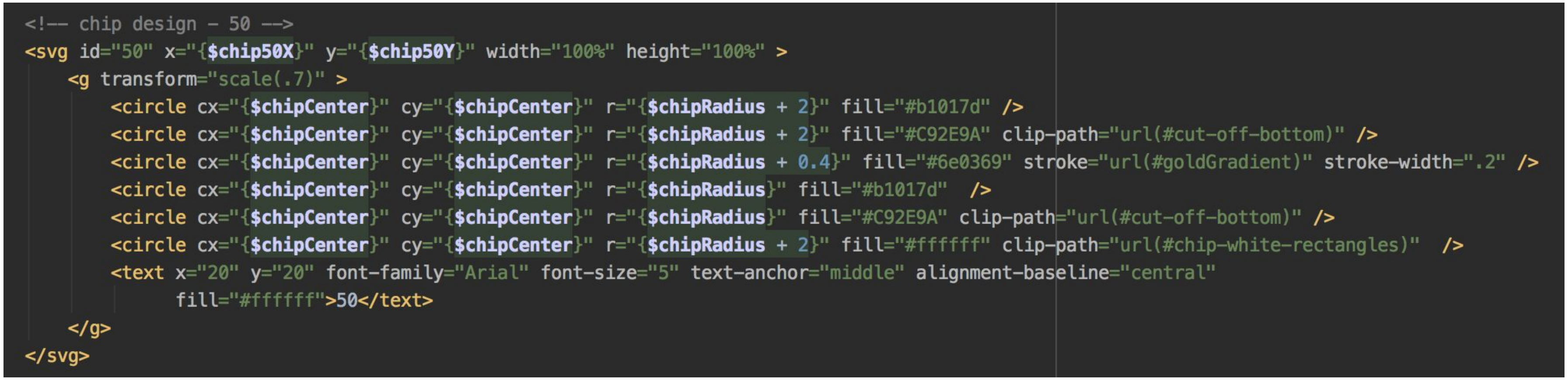


```
<!-- Lines of the grid inside the cards -->
<xsl:variable name="HorLine1" select="25"/>
<xsl:variable name="HorLine2" select="30"/>
<xsl:variable name="HorLine3" select="35"/>
```

# Design

## Chips

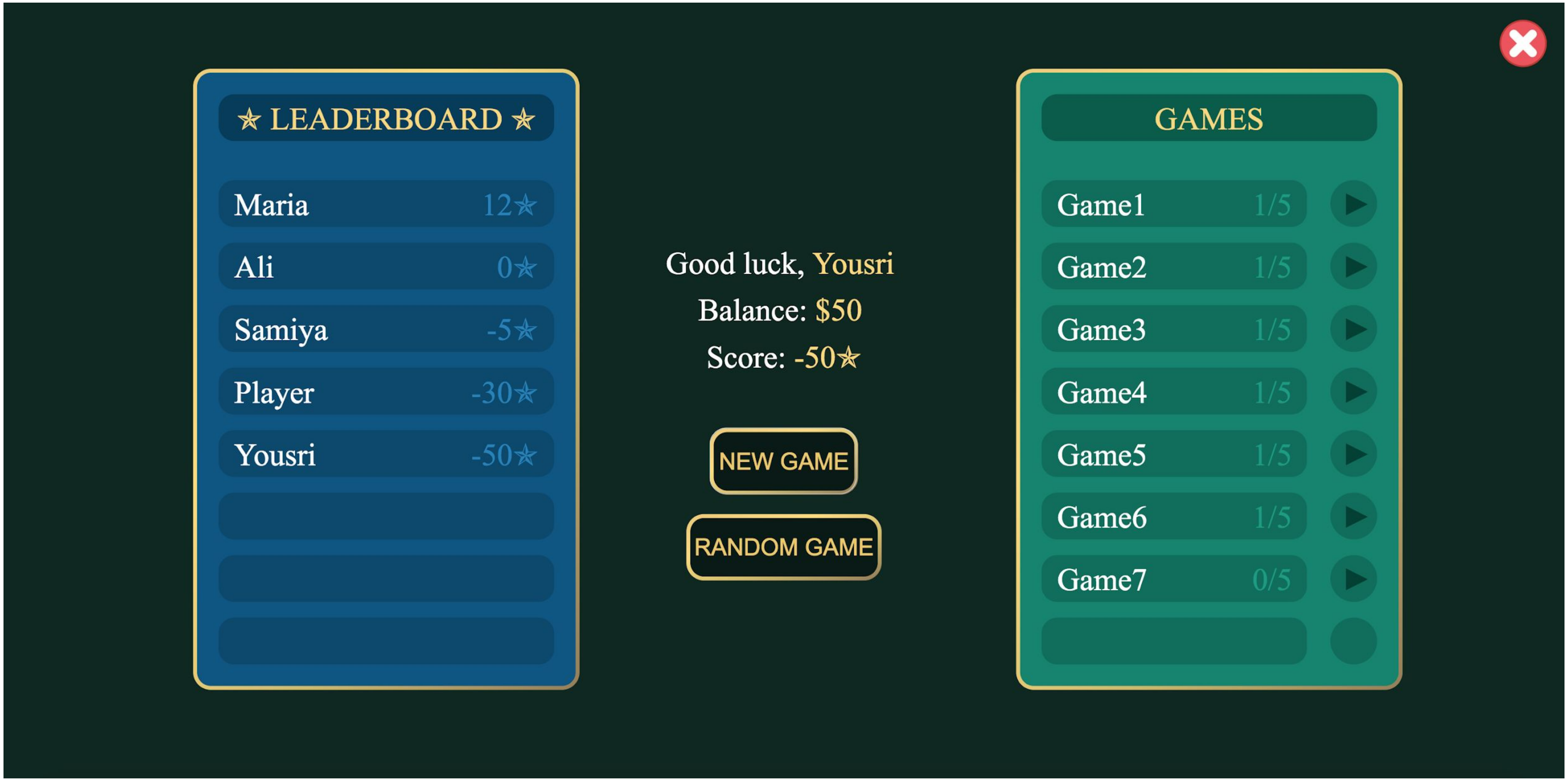




```
<!-- chip design - 50 -->
<svg id="50" x="{$chip50X}" y="{$chip50Y}" width="100%" height="100%" >
    <g transform="scale(.7)" >
        <circle cx="{$chipCenter}" cy="{$chipCenter}" r="{$chipRadius + 2}" fill="#b1017d" />
        <circle cx="{$chipCenter}" cy="{$chipCenter}" r="{$chipRadius + 2}" fill="#C92E9A" clip-path="url(#cut-off-bottom)" />
        <circle cx="{$chipCenter}" cy="{$chipCenter}" r="{$chipRadius + 0.4}" fill="#6e0369" stroke="url(#goldGradient)" stroke-width=".2" />
        <circle cx="{$chipCenter}" cy="{$chipCenter}" r="{$chipRadius}" fill="#b1017d"  />
        <circle cx="{$chipCenter}" cy="{$chipCenter}" r="{$chipRadius}" fill="#C92E9A" clip-path="url(#cut-off-bottom)" />
        <circle cx="{$chipCenter}" cy="{$chipCenter}" r="{$chipRadius + 2}" fill="#ffffff" clip-path="url(#chip-white-rectangles)"  />
        <text x="20" y="20" font-family="Arial" font-size="5" text-anchor="middle" alignment-baseline="central"
              fill="#ffffff">50</text>
    </g>
</svg>
```

# Design

## Lounge



☆ LEADERBOARD ☆

| | |
|---|---|
| Maria | 12☆ |
| Ali | 0☆ |
| Samiya | -5☆ |
| Player | -30☆ |
| Yousri | -50☆ |

Good luck, Yousri

Balance: $50

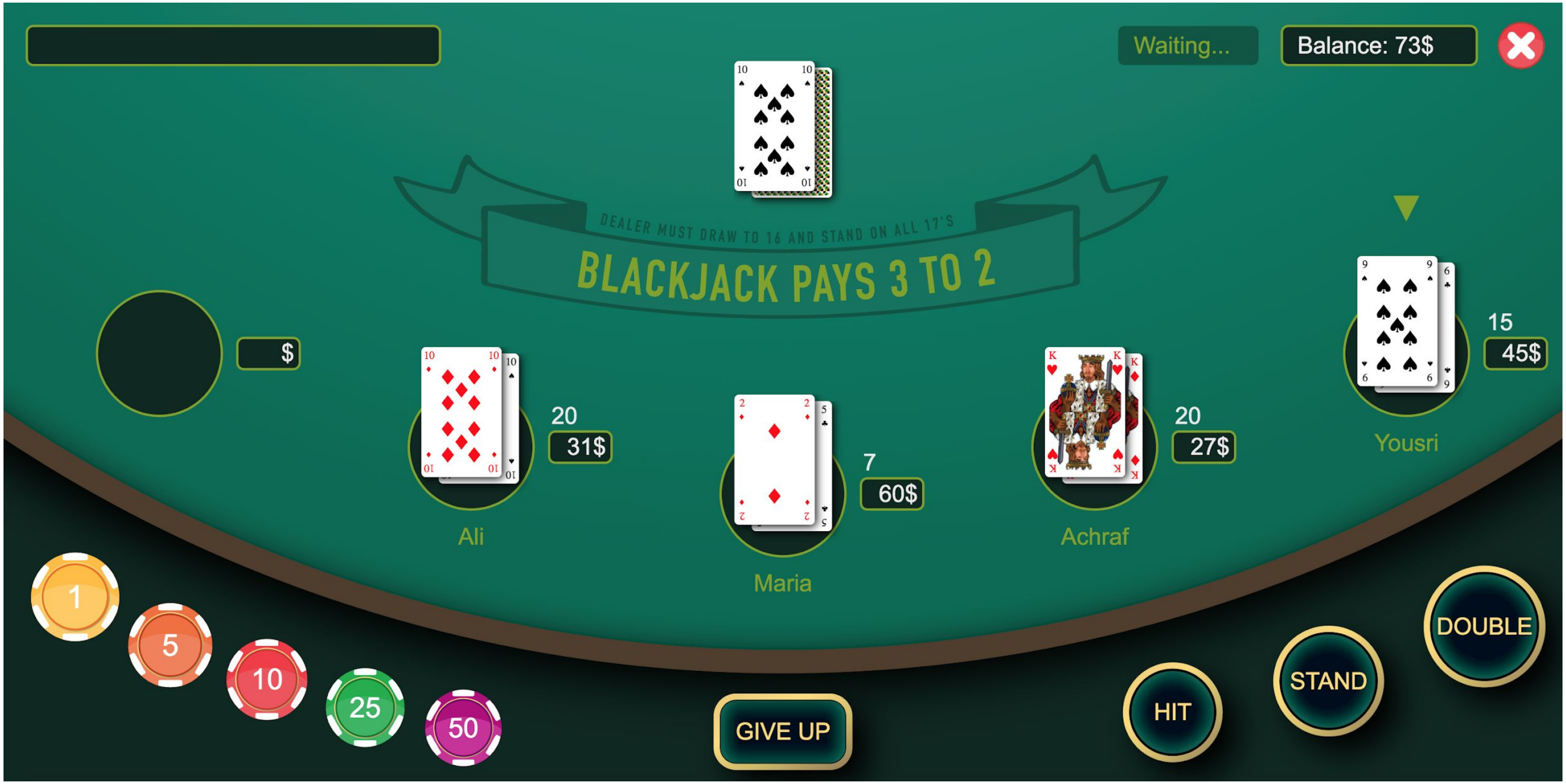
Score: -50☆

GAMES

| | |
|---|---|
| Game1 | 1/5 |
| Game2 | 1/5 |
| Game3 | 1/5 |
| Game4 | 1/5 |
| Game5 | 1/5 |
| Game6 | 1/5 |
| Game7 | 0/5 |

# Design

## Final Game



# Functionality

## Technologies: XQuery

Update constraint: return a value or make an update

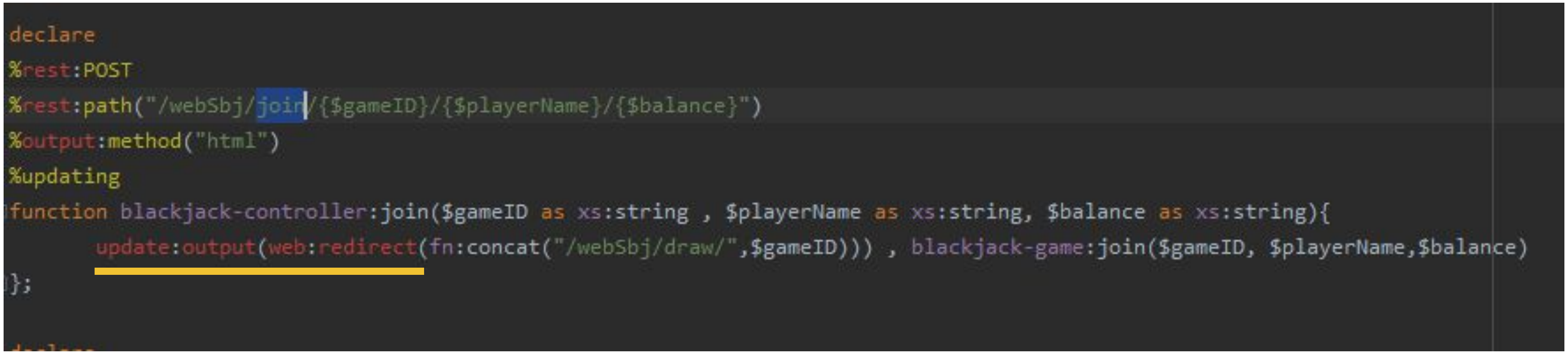
This can be solved with two steps:

1. Using HTTP redirects and the function web:redirect
2. Using the BaseX method update:output

```
update:output(web:redirect(URI))
```

# Functionality

## Technologies: XQuery



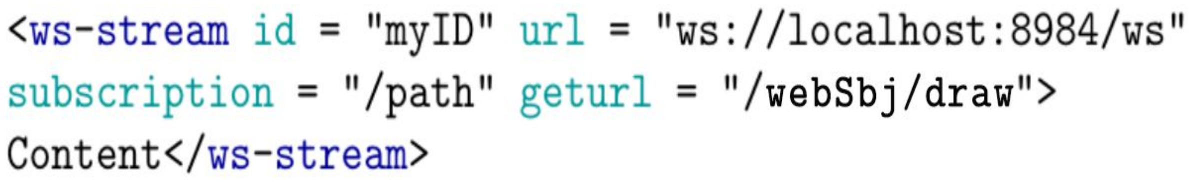
```
declare
%rest:POST
%rest:path("/webSbj/join/{$gameID}/{$playerName}/{$balance}")
%output:method("html")
%updating
function blackjack-controller:join($gameID as xs:string , $playerName as xs:string, $balance as xs:string){
       update:output(web:redirect(fn:concat("/webSbj/draw/",$gameID))) , blackjack-game:join($gameID, $playerName,$balance)
};
```

Draw function here is responsible for sending the response to the clients.

The response in draw function is sent via  webSockets and the ws-element offered by baseX stomp

# Functionality

## WS-Element

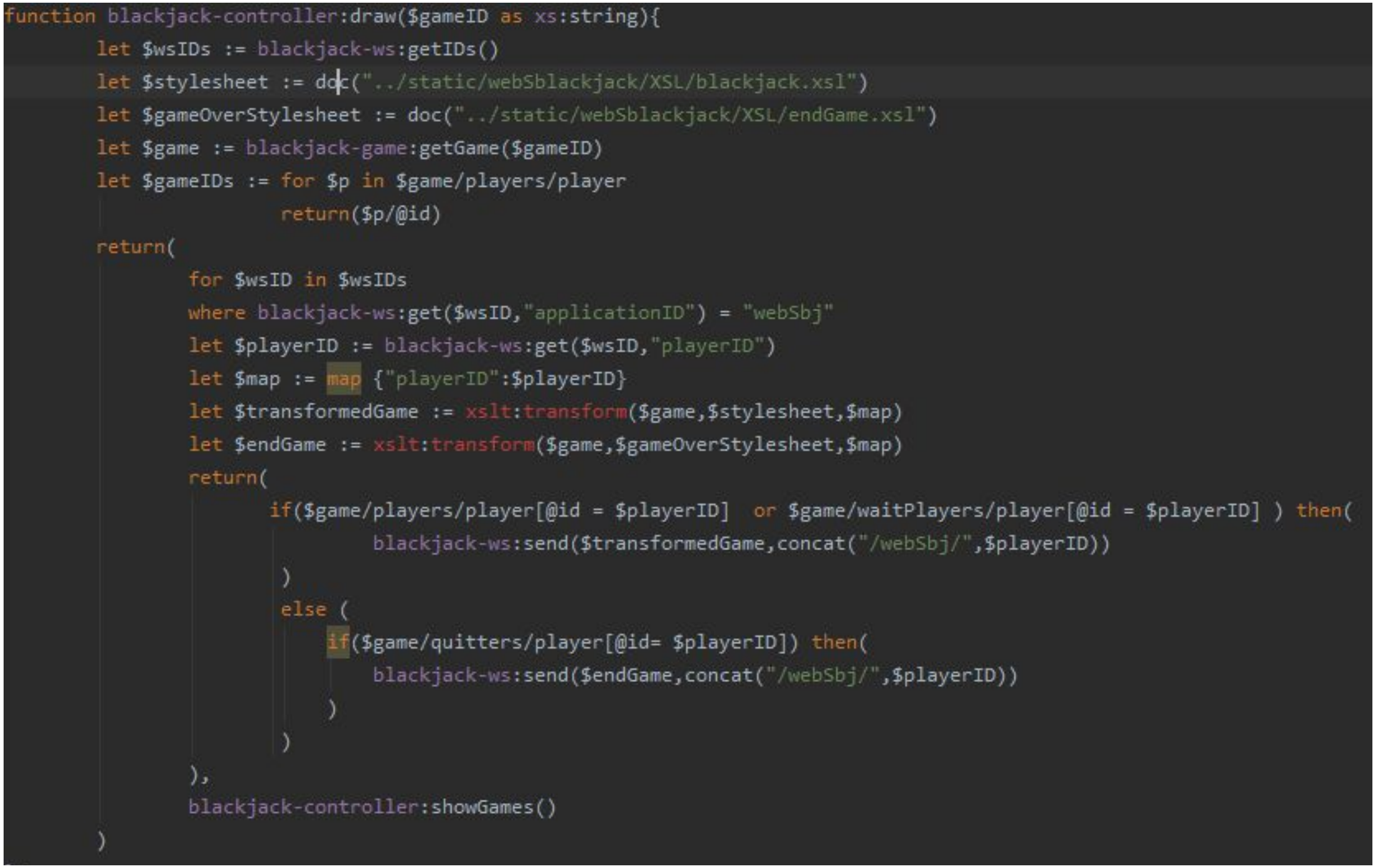


```
<ws-stream id = "myID" url = "ws://localhost:8984/ws"
subscription = "/path" geturl = "/webSbj/draw">
Content</ws-stream>
```

- **id:** identification for Websocket element
- **url:** target address for establishing the WebSocket connection
- **subscription:** defines which paths the WebSocket element will subscribe to
- **geturl:** to load first state of the WebSocket element

# Functionality

## XQuery and WebSockets



```
function blackjack-controller:draw($gameID as xs:string){
        let $wsIDs := blackjack-ws:getIDs()
        let $stylesheet := doc("../static/webSblackjack/XSL/blackjack.xsl")
        let $gameOverStylesheet := doc("../static/webSblackjack/XSL/endGame.xsl")
        let $game := blackjack-game:getGame($gameID)
        let $gameIDs := for $p in $game/players/player
                        return($p/@id)
        return(
                for $wsID in $wsIDs
                where blackjack-ws:get($wsID,"applicationID") = "webSbj"
                let $playerID := blackjack-ws:get($wsID,"playerID")
                let $map := map {"playerID":$playerID}
                let $transformedGame := xslt:transform($game,$stylesheet,$map)
                let $endGame := xslt:transform($game,$gameOverStylesheet,$map)
                return(
                       if($game/players/player[@id = $playerID]  or $game/waitPlayers/player[@id = $playerID] ) then(
                                blackjack-ws:send($transformedGame,concat("/webSbj/",$playerID))
                        )
                        else (
                            if($game/quitters/player[@id= $playerID]) then(
                                blackjack-ws:send($endGame,concat("/webSbj/",$playerID))
                            )
                        )
                ),
                blackjack-controller:showGames()
        )
```

# Demo

# Questions?