Meta Ping-Pong

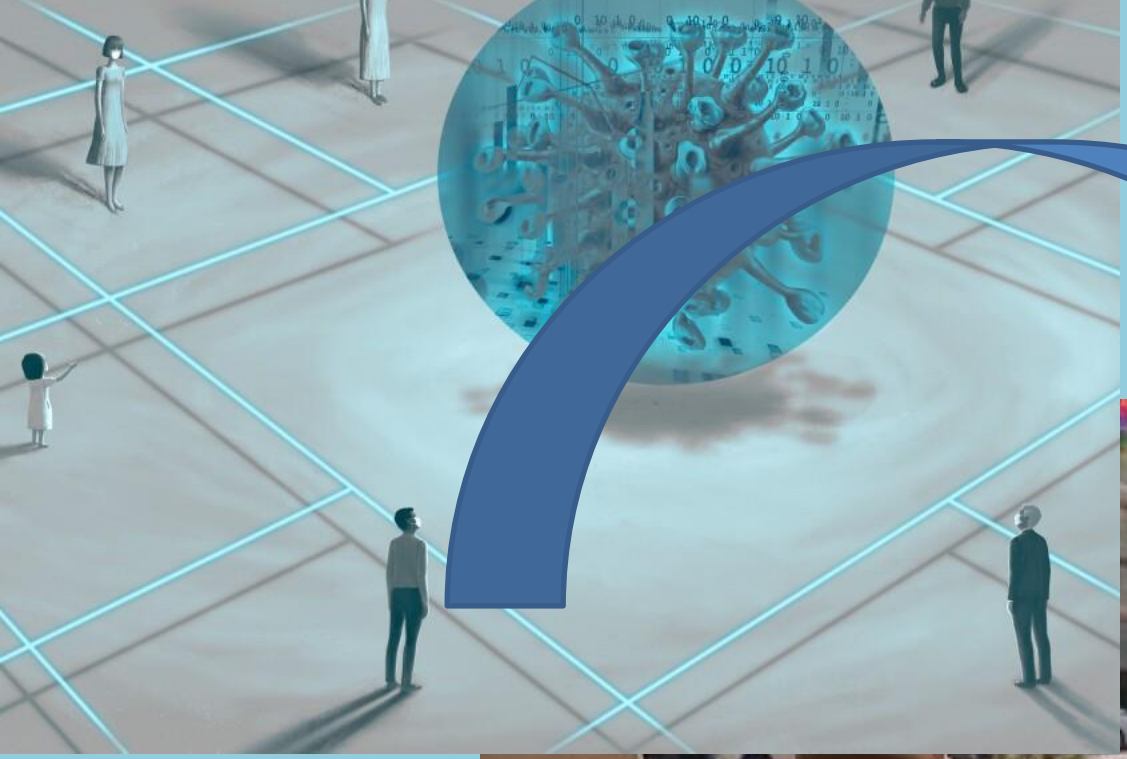
FFHS
Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig
Mitglied der SUPSI

Modul: WebE

Dozent: Heinrich Zimmermann

Präsenten: Chantale Gihara , Theologos Baxevanos

Zürich, 01.12.2021




SALESSIA

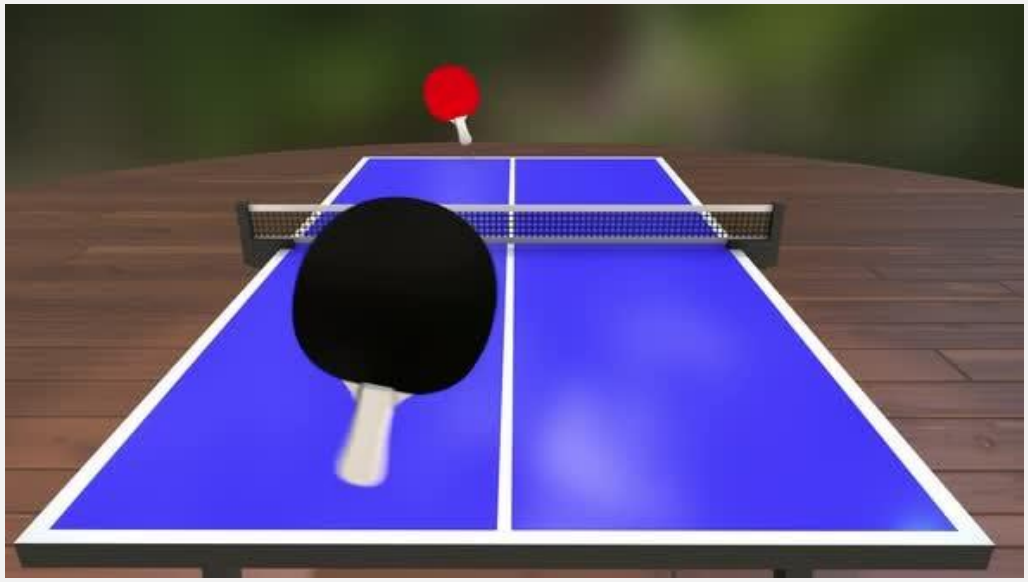
Agenda

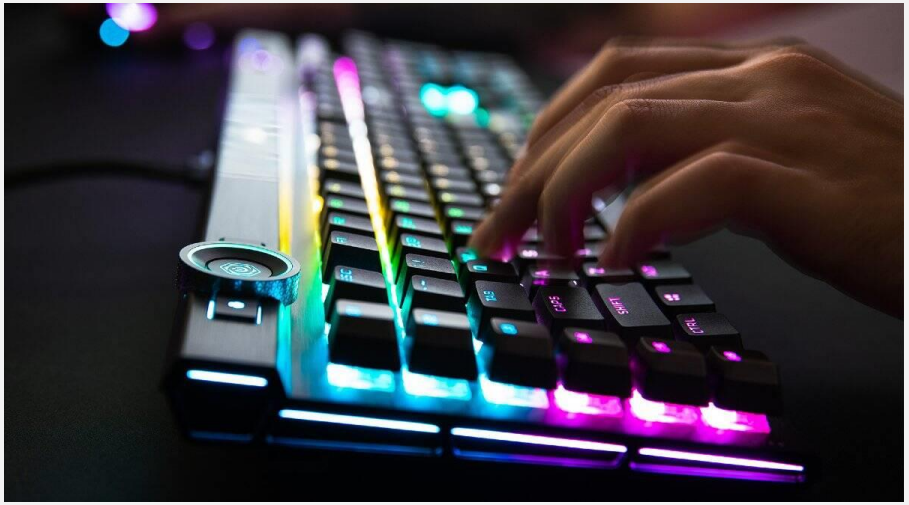
FFHS
Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig
Mitglied der SUPSI

1. Die Spielregeln
2. Die Architektur des Spiels
3. Die verwendeten Technologien
4. Die interessantesten Code Beispiele
5. Das methodische Vorgehen
6. Demo
7. Die Erkenntnisse

1.2 Spielregeln – Spiel

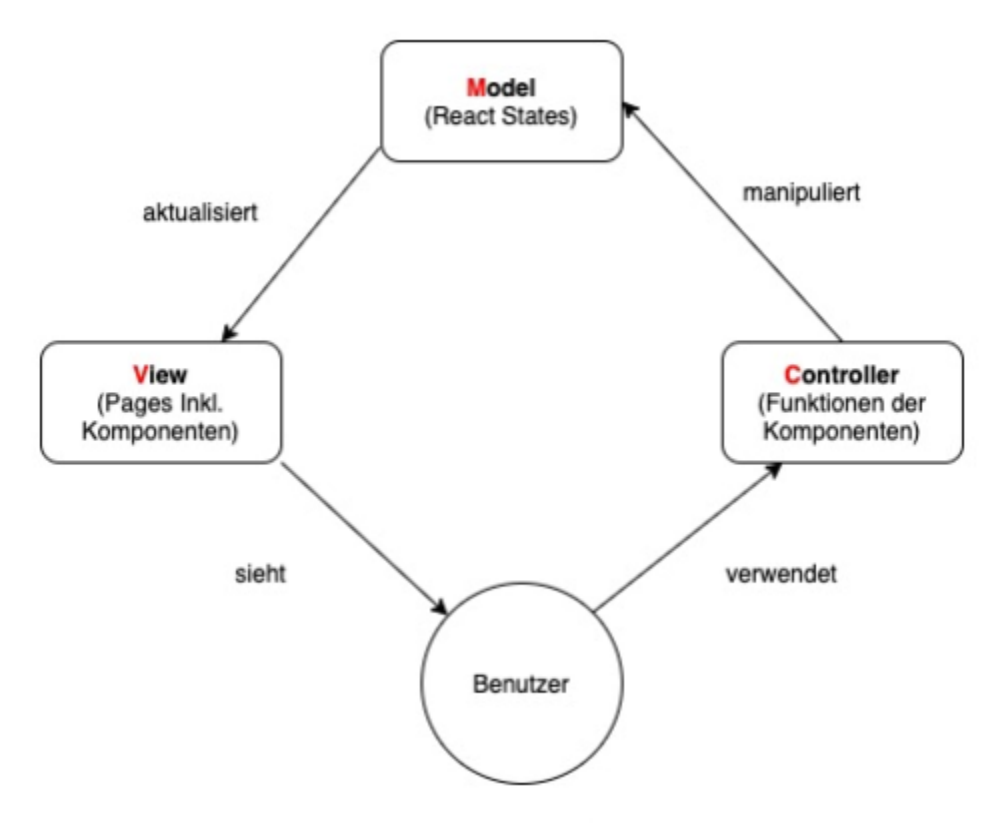
FFHS
Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig
Mitglied der SUPSI





2. Die Architektur des Spieles

- MVC Architektur



3. Die verwendeten Technologien

FFHS
Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig
Mitglied der SUPSI

- Server: Express.js
- DB: PostgreSQL
- Backend: Node.js inkl. Libraries
- Frontend: ReactJS inkl.Libraries
- Docker

4.1 Die interessantesten Code Beispiele

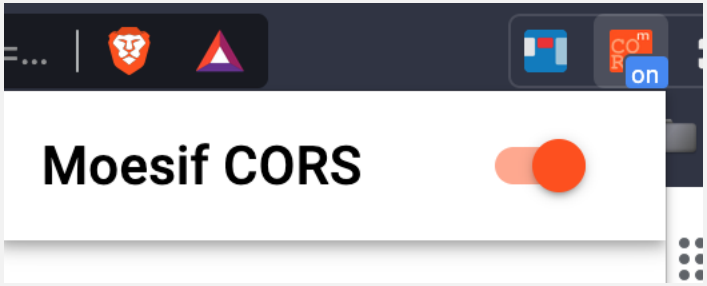

```
  // chat messages - Websockets communication
io.on('connection', socket => {
    socket.on('message', ({ name, message }) => {
      io.emit('message', { name, message })
    })
  }
server.listen(4000, () => {
    console.log('Websockets listening on Port 4000');
  });

function ChatRoom() {
  const [state, setState] = useState({ message: "", name: "" });
  const [chat, setChat] = useState([]);

  const socketRef = useRef();

  useEffect(() => {
    socketRef.current = io.connect("http://localhost:4000");
    socketRef.current.on("message", ({ name, message }) => {
      setChat([...chat, { name, message }]);
    });
    return () => socketRef.current.disconnect();
  }, [chat]);

  const onTextChange = (e) => {
    setState({ ...state, [e.target.name]: e.target.value });
  };

  const onMessageSubmit = (e) => {
    const { name, message } = state;
    socketRef.current.emit("message", { name, message });
    e.preventDefault();
    setState({ message: "", name });
  };
```

4.2 Die interessantesten Code Beispiele


```
const express = require("express");
const bodyParser = require("body-parser");
const router = express.Router();
const app = express();

// add router in express app
app.use("/",router);

//
const http = require('http');
const server = http.createServer(app);
const { Server } = require("socket.io");
const io = new Server(server);
//

const { pool } = require("./dbConfig.js");
const bcrypt = require("bcrypt");
//

// create application/json parser
const jsonParser = bodyParser.json()

const cors = require('cors');
app.use(cors());

router.post('/users/register', jsonParser, async (req, res) => {
```

4.3 Die interessantesten Code Beispiele


```
// create application/json parser
const jsonParser = bodyParser.json()

const cors = require('cors');
app.use(cors());

router.post('/users/register', jsonParser, async (req, res) => {
```



4.4 Die interessantesten Code Beispiele

```
// check if the email already exists in our db
pool.query(
    `SELECT * FROM users
        WHERE email = $1`,
    [email],
    (err, results) => {
        if (err) {
            console.log(err);
        }
        // if results.rows.length > 0  -----> account already
        if (results.rows.length > 0) {
            res.send({ emailExists: true, registered: false })
        } else {
            // if the e-mail does not already exist in the db,
            pool.query(
                `INSERT INTO users (name, email, password)
                VALUES ($1, $2, $3)
                RETURNING id, name, email,  password`,
                [nickname, email, hashedPassword], // these ar
                (err, results) => {
                    if (err) {
                        throw err;
                    }
                }
            );
            res.send( { registered: true } )
        }
    }
);
```

```
const handleSubmit = (e) => {
  e.preventDefault();
  const data = { nickname, email, password, password2 };
  console.log(data);

  fetch(`http://localhost:3001/users/register`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(data),
  })
    .then((res) => res.json())
    .then((response) => {
      setAccountExists(response.emailExists);
      setRegistered(response.registered);

      if(response.emailExists){
        setnickname("");
        setemail("");
        setpassword("");
        setpassword2("");
      }
    });
};
```

4.4 Die interessantesten Code Beispiele

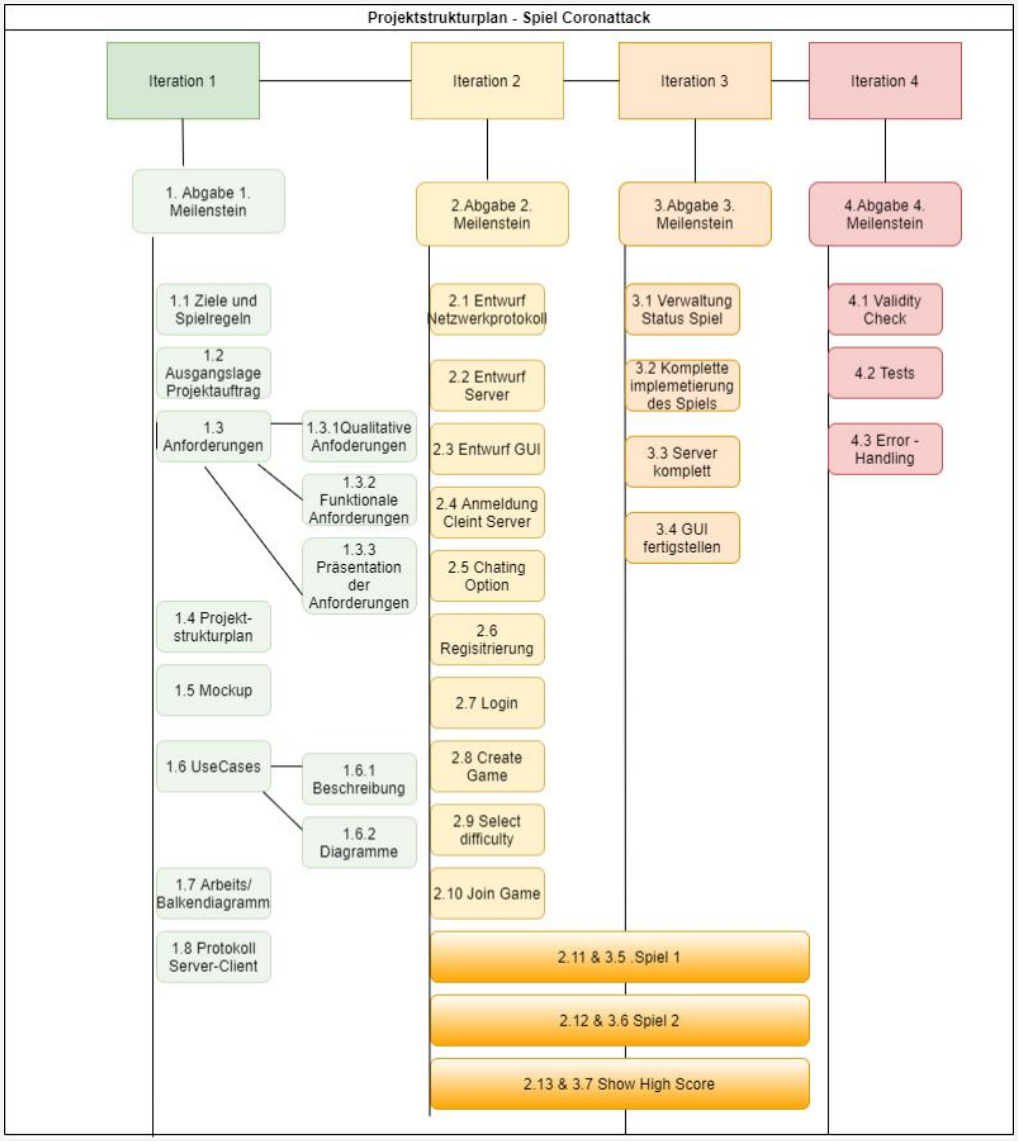
```
this.ballSpeedX = this.Height / props.difficulty;
```

```
const ctx = this.canvas.getContext("2d");
greatmerlin, 13 hours ago | 1 author (greatmerlin)
function KeyListener() {
  this.pressedKeys = [];
  this.keydown = (e) => {
    this.pressedKeys[e.keyCode] = true;
  };
  this.keyup = (e) => {
    this.pressedKeys[e.keyCode] = false;
  };
  document.addEventListener("keydown", this.keydown.bind(this));
  document.addEventListener("keyup", this.keyup.bind(this));
}
KeyListener.prototype.isPressed = function (key) {
  return this.pressedKeys[key] ? true : false;
};
KeyListener.prototype.addKeyPressListener = function (keyCode, callback) {
  document.addEventListener("keypress", (e) => {
    if (e.keyCode === keyCode) callback(e);
  });
};
const keys = new KeyListener();
```

```
// draw everything on screen
drawAll = (ctx) => {      greatmerlin, 13 hours ago • meta tranfer f
  // screen
  ctx.fillStyle = "#661177";
  ctx.fillRect(0, 0, this.Width, this.Height);
  // middle dashed line
  ctx.strokeStyle = "#fff";
  ctx.setLineDash([10]);
  ctx.beginPath();
  ctx.moveTo(this.Width / 2, 0);
  ctx.lineTo(this.Width / 2, this.Height);
  ctx.stroke();
  // score
  ctx.font = "30px Orbitron";
  ctx.fillStyle = "#888";
  ctx.fillText(this.player1Score, this.Width / 2 / 2, 100);
  ctx.fillText(this.player2Score, (this.Width / 2) * 1.5, 100);
  // 2 rects
  ctx.fillStyle = "#fff";
  ctx.fillRect(0, this.paddle1Y, 10, this.paddleWidth);
  ctx.fillRect(this.Width - 10, this.paddle2Y, 10, this.paddleWidth);
  // ball
  ctx.beginPath();
  ctx.arc(this.ballX, this.ballY, this.ballRadius, 0, Math.PI * 2);
  ctx.fill();
};
```

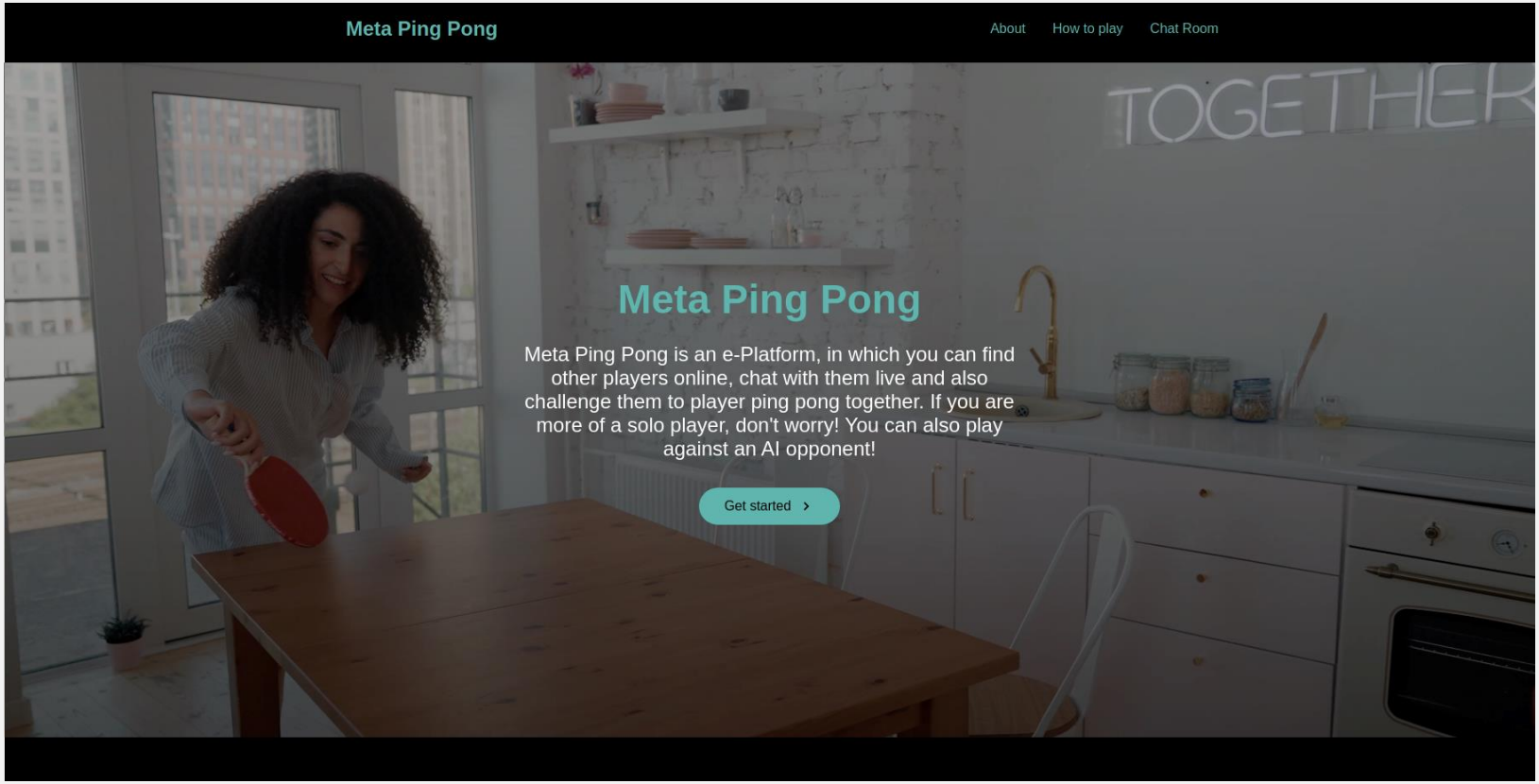
5. Das methodische Vorgehen

- Iteratives Vorgehen:



6. Demo

FFHS
Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig
Mitglied der SUPSI



7.1 Die Erkenntnisse

FFHS
Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig
Mitglied der SUPSI

- Das Spiel wurde total unterschätzt
- Unterschiedliche Entwicklungsumgebungen sind sehr relevant und in den Risiken zu beachten
- Cores – Problem, abhängig nicht nur von dem Browser sondern auch von dem OS
- Docker-Compose war nicht nur ein Vorteil, auch hier OS zu beachten!
- Server Express ist nicht der Beste in der Benutzung
- Middleware ist zu beachten
- Teamwork mit unterschiedlichen Code Fortschritt, viel Code für nichts. Mussten uns einigen.

Durch Fehlschläge lernt man am Meisten, somit Ziel Modul erreicht!!

7.2 Die Erkenntnisse

- HTML
- CSS
- JS & React, Node, Express
- Websockets
- Design Patterns

7.3 Die Erkenntnisse

- Canvas API
- Kollisionserkennung im Spiel
- DB Storage
- Client/Server
- Projekt Management

FFHS
Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig
Mitglied der SUPSI

Danke für die Aufmerksamkeit

Fragen?