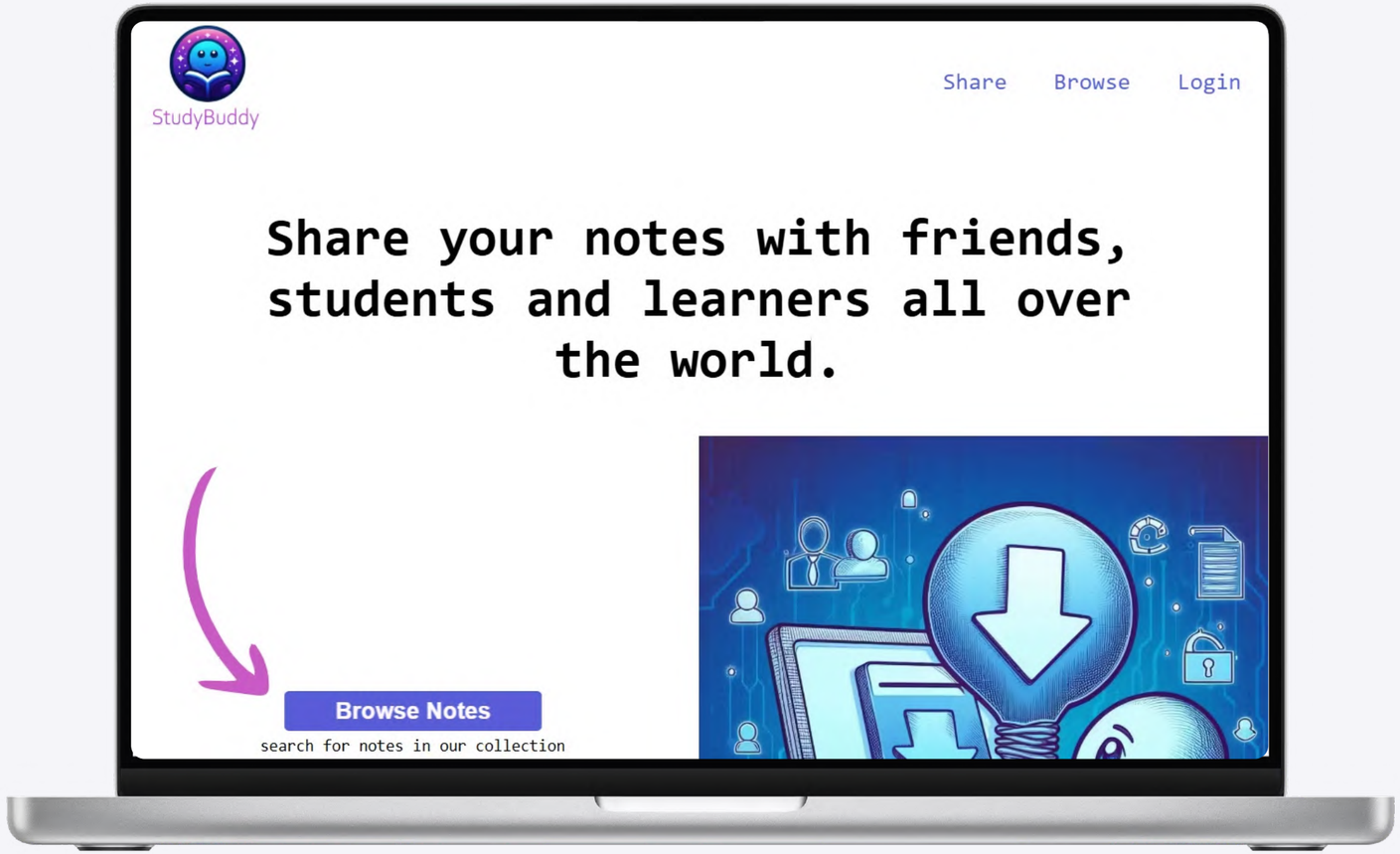
Chiara Herz, Nils Kubousek, Atussa Mehrawari, Lukas Richter

DHBW Duale Hochschule Baden-Württemberg Ravensburg

StudyBuddy

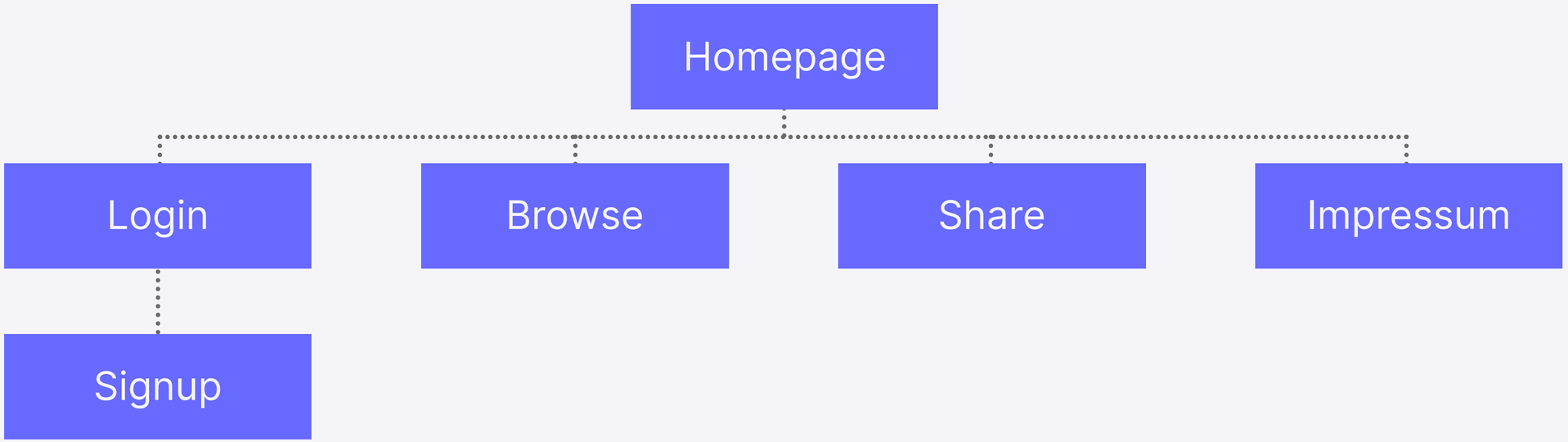
StudyBuddy



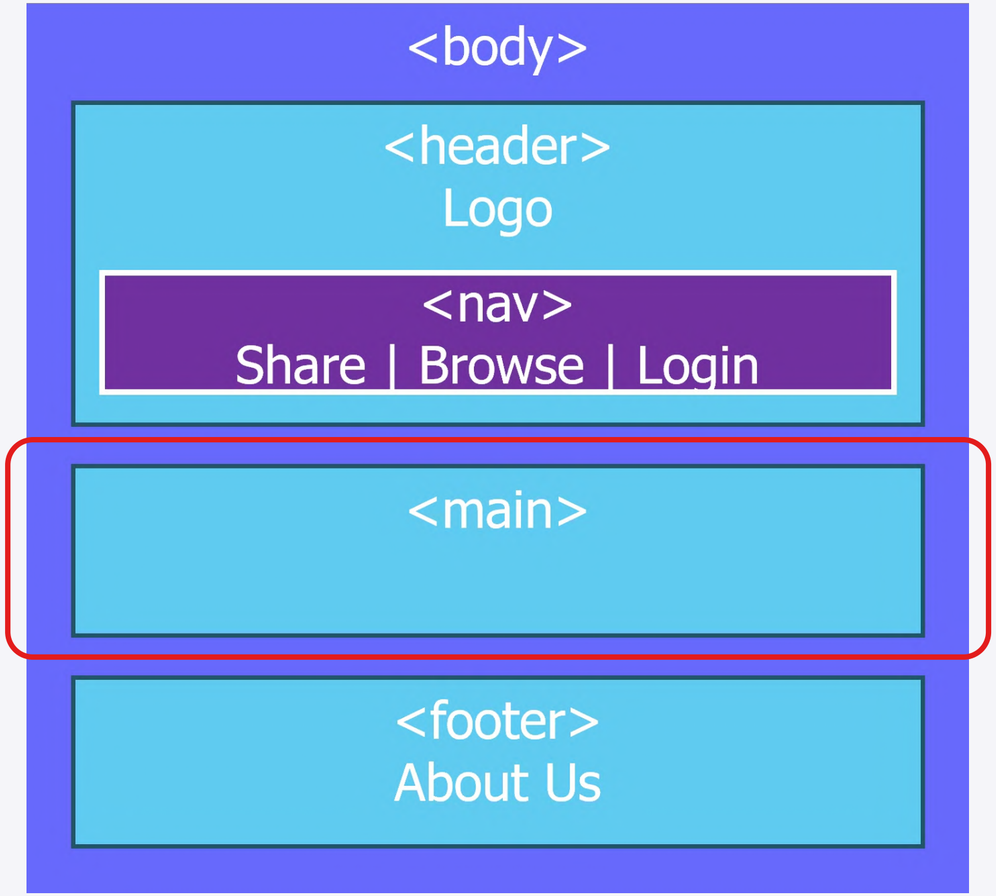
Gliederung

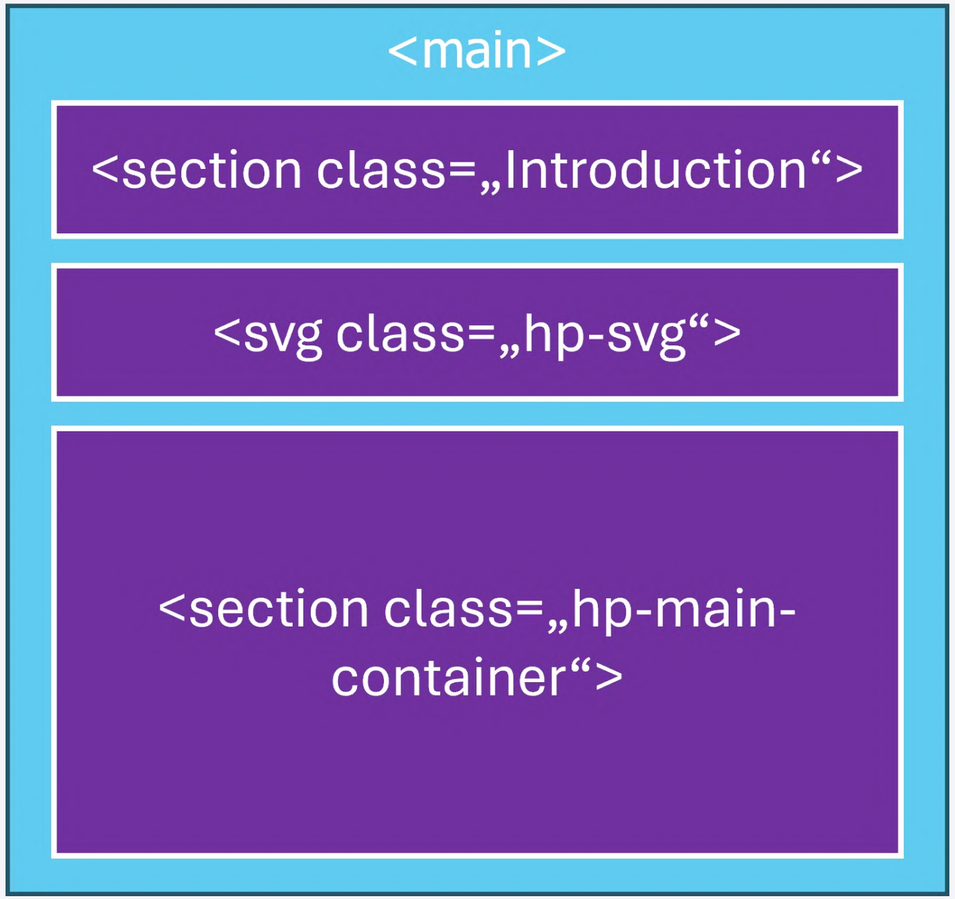
1. Struktur & Design
2. User Handling
3. Document Handling
4. Datenbank
5. Future Features

Struktur - Sitemap



Struktur - Homepage





Struktur - Homepage

Demo

Design - Homepage


```
:root{

    font-family: monospace;
    font-size: 16px;
    --ui-base-color: white;
    --ui-blue: #474ad7e6;
    […]
    --ui-font-size-big: max(1rem, 2vw)
    […]
}
```

Design - Homepage

```
.hp-main-container{

    animation: slideInRL 1s ease-in-out;
    display: flex;
    flex-flow: row wrap;
    […]
}
```

```
.hp-main-cards{
    […]
    flex: 1 1 50%;
    […]
}
```

Design - Homepage

```
.hp-main-container{

    animation: slideInRL 1s ease-in-out;
    display: flex;
    flex-flow: row wrap;
    […]
}
```

```
@media screen and (max-width:
426px){
    .hp-main-container{
        flex-flow: column wrap;
        […] }
    .hp-main-cards{
        width: 100%; }}
```

Bis 425 Pixel

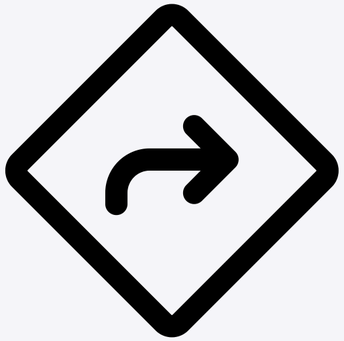
426 - 767 Pixel

Ab 768 Pixel

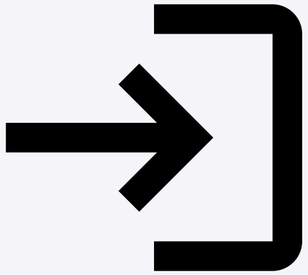
Design - Homepage

Demo

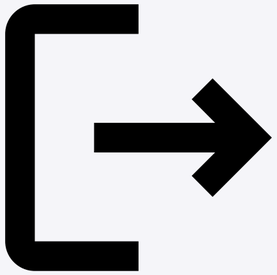
User Handling



Signup

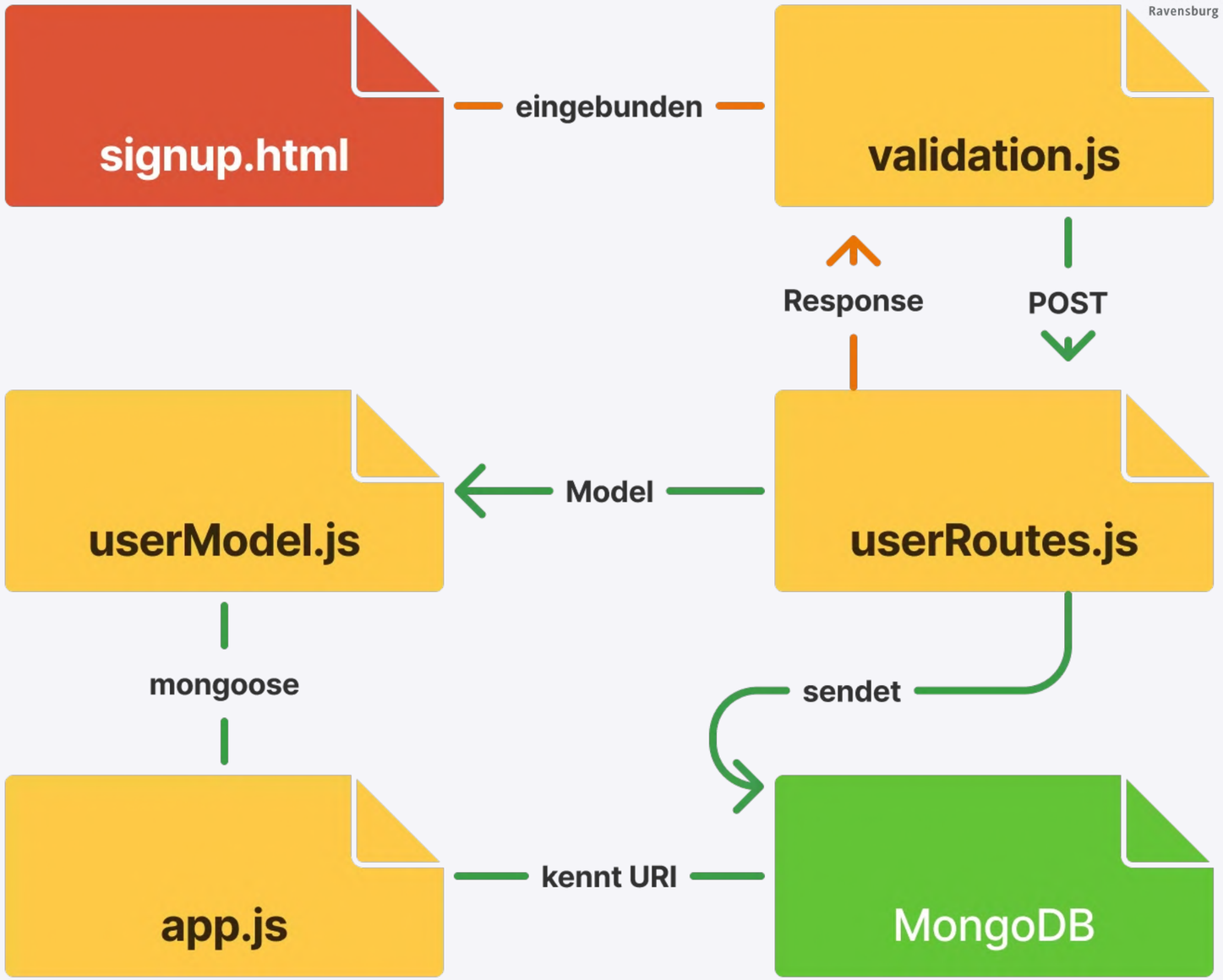


Login



Logout

Signup


signup.html
eingebunden
validation.js
Response
POST
userModel.js
Model
userRoutes.js
mongoose
sendet
app.js
kennt URI
MongoDB

Signup (validation.js)

```
signupForm.addEventListener('submit', async (e) => {
        e.preventDefault();
        const firstName = document.getElementById('firstname-input').value;
        const email     = document.getElementById('email-input').value;
        const password  = document.getElementById('password-input').value;

        try {
            const res = await fetch('/api/users/signup'), {
                method: 'POST',
                headers: { 'Content-Type': 'application/json'}
                body: JSON.stringify({ firstName, email, password })
         });
        …
}
```

DHBW Duale Hochschule Baden-Württemberg Ravensburg

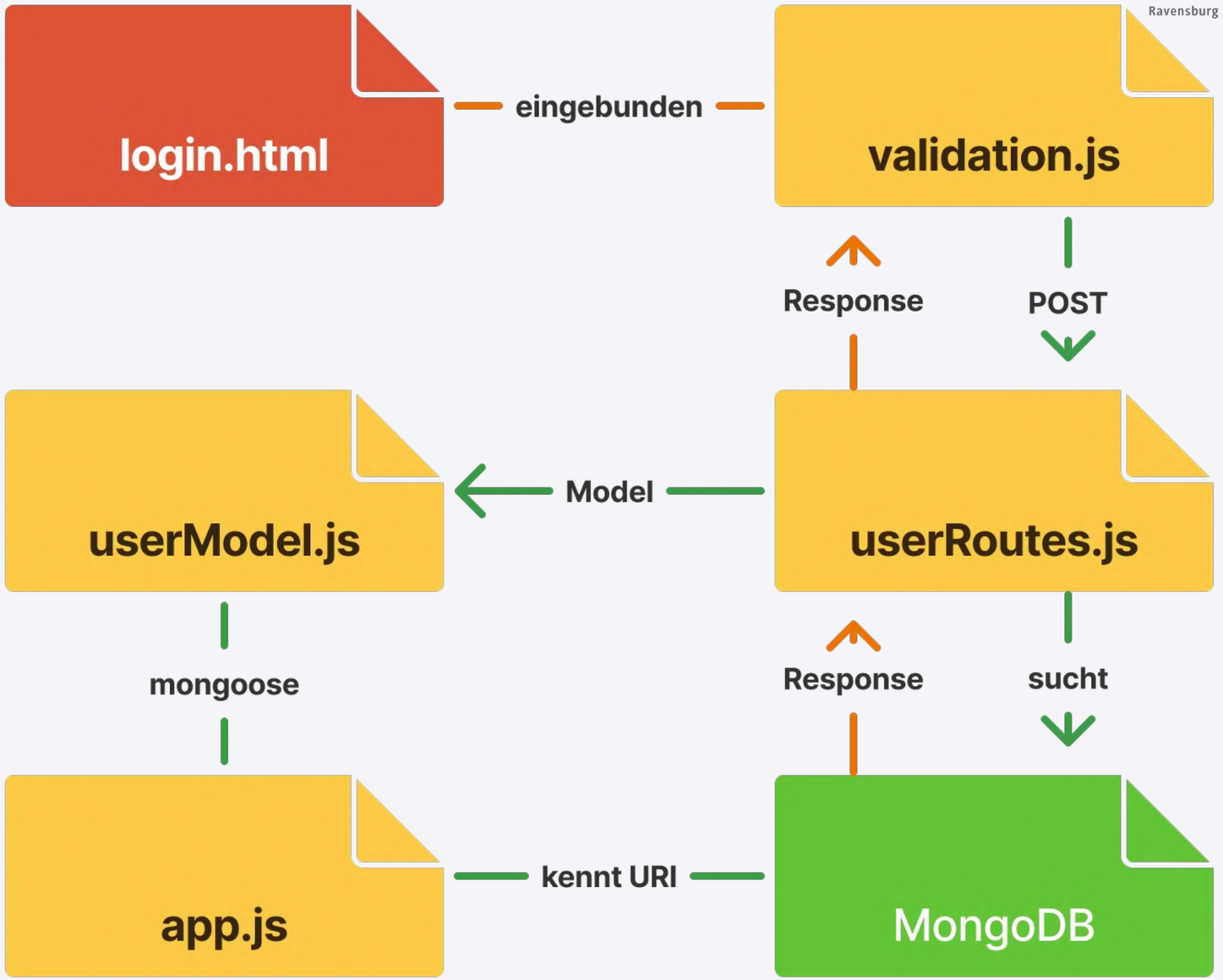
StudyBuddy

Signup (userRoutes.js)

```
const express = require("express");
const User    = require("../models/userModel");
const router  = express.Router();

router.post('/signup', async (req, res) => {
    try {
        const { firstName, email, password } = req.body;
        user = new User({
            firstName: firstName,
            email: email,
            password: password

        await user.save();
}
```

Try Pitch

Login


login.html
eingebunden
validation.js
Response
POST
userModel.js
Model
userRoutes.js
mongoose
Response
sucht
app.js
kennt URI
MongoDB

Login (validation.js)

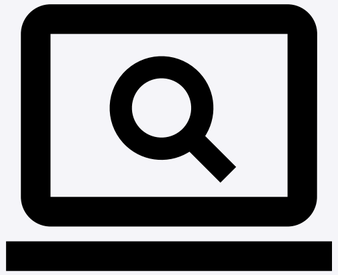
```
loginForm.addEventListener('submit', async (e) => {
        e.preventDefault();
        const email     = document.getElementById('email-input').value;
        const password  = document.getElementById('password-input').value;
        try {
            const res = await fetch('/api/users/login'), {
                method: 'POST',
                headers: { 'Content-Type': 'application/json'}
                credentials: 'include',
                body: JSON.stringify({ email, password })
         });
            const data = await res.json()
            if (res.ok) {
                localStorage.setItem("firstName", data.firstName);
}}});
```

Login (userRoutes.js)

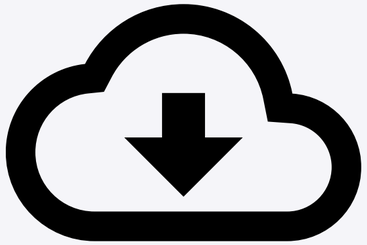
```
const express =  require("express")
const User    =  require("../models/userModel")
const router  =  express.Router();

router.post("/login", async (req, res) => {
    try {
        const { email, password } = req.body;
        const user = await User.findOne({ email });
        const isMatch = password === user.password;

        req.session.user = {
            id: user._id,
            firstName: user.firstName,
            email: user.email
}}};
```

Document Handling

Upload



Browsing

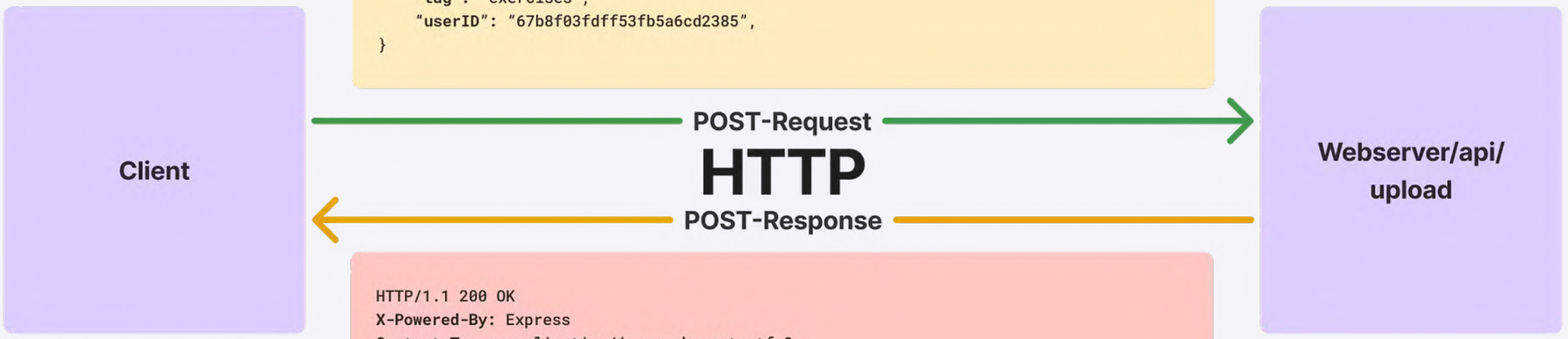


Download

Upload

```
POST /api/upload HTTP/1.1
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9,de;q=0.8,de-DE;q=0.7
Connection: keep-alive
Content-Length: 180989
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryMQRiBB4PTkE3qGDe
Cookie:
connect.sid=s%3AqqYz0JxImKbyW7ez8YzoacvjPOdHoYL5.Cf1GyHW1j8SjabHxtlbNm8GrbdXa2A84BT39kc
Ep7b4
Host: localhost:3000
Origin: http://localhost:3000
Referer: http://localhost:3000/share
```

```
{
    “docTitle”: “Mathe”,
    “description”: “Mathe, Satz des Pythagoras”,
    “uploadFile”: (binary),
    “tag”: “exercises”,
    “userID”: “67b8f03fdff53fb5a6cd2385”,
}
```




```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 36
Date: Tue, 25 Feb 2025 22:47:23 GMT
Connection: keep-alive
Keep-Alive: timeout=5
```

```
{
    "message":"Doc saved successfully"
}
```

DHBW Duale Hochschule Baden-Württemberg Ravensburg

StudyBuddy

Upload - Clientseitig

```
const response = await fetch('/api/users/status');
const status = await response.json();
if (status.loggedIn) {
    const userID = status.user.id;
    const formData = new FormData();
    formData.append('docTitle', documentTitle);
    formData.append('description', documentDescription);
    formData.append('uploadFile', uploadFile);
    formData.append('tag', tag);
    formData.append('userID', userID);
    const uploadResponse = await fetch('/api/upload', {
        method: 'POST',
        body: formData
    });
    …
```

Try Pitch

Upload - Serverseitig

```
const fileObj = req.files.uploadFile;
const fileBuffer = fileObj.data;
const fileType = fileObj.mimetype;
const originalName = fileObj.name;

const title = req.body.docTitle;
const description = req.body.description;
const tag = req.body.tag;
const userID = req.body.userID; // Datenbank-Abfrage User.findById(userID).exec()

const uploadDate = new Date();
const doc = new Doc({ userID, title, uploadDate, description, file: fileBuffer, fileType,
originalName, tag });
await doc.save();
return res.status(200).json({ message: "Doc saved successfully" });
```

Browsing


GET /browse?searchTerm=math HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9,de;q=0.8,de-DE;q=0.7
Connection: keep-alive
Cookie:
connect.sid=s%3AZFVOdSL0KNSwCFEzM97TrUAFZ7SCfHfo.REj2qPCl2Zgt2MGJRw2ODz5fc51FwiciycQ2Uz2UTO4
Host: localhost:3000
Referer: http://localhost:3000/browse
Client
GET-Request
HTTP
GET-Response
Webserver/browse
HTTP/1.1 200 OK
X-Powered-By: Express
Date: Tue, 25 Feb 2025 22:20:26 GMT
Connection: keep-alive
Keep-Alive: timeout=5
{
"numDocs": 1,
"documents": [
{
"docID": "67be43b8a357598f8174992b",
"docTitle": "Mathematics Notes",
"docDescription": "Math stuff",
"docTag": "summary",
"docAuthor": "lukas",
"docDate": "2025-02-25T22:27:04.195Z"
}
]
}

DHBW Duale Hochschule Baden-Württemberg Ravensburg

StudyBuddy

Browsing - Clientseitig

```
searchForm.addEventListener("submit", async (e) => {
    e.preventDefault();
    const formData = new FormData(searchForm);
    const searchTerm = formData.get("searchTerm");
    const tag = formData.get("tag");
    let url = `/browse?searchTerm=${encodeURIComponent(searchTerm)}`;
    if (tag) {
        url += `&tag=${encodeURIComponent(tag)}`;
    }
    try {
        const response = await fetch(url);
        const data = await response.json();
        …
    …
…
```

Try Pitch

Browsing - Serverseitig

```
const { searchTerm, tag } = req.query;
let query = {
    $or: [
        {
            title: {
                $regex: ".*" + searchTerm + ".*",
                $options: "i"
            }
        }, {
            description: {
                $regex: ".*" + searchTerm + ".*",
                $options: "i"
            }
        }
    ]
};
```

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

StudyBuddy

Browsing - Serverseitig

```
if (tag && tag.trim() !== "") {
    query.tag = { $in: tag.split(",").map(t => t.trim()) };
}
const matches = await Doc.find(query, {
    userID: 1,
    title: 1,
    uploadDate: 1,
    description: 1,
    tag: 1
}).exec();

const populatedMatches = await Doc.populate(matches, {
    path: "userID",
    model: User,
    select: "firstName"
});
```

Try Pitch

Browsing - Serverseitig

```
const documents = populatedMatches.map(doc => ({
    docID: doc._id,
    docTitle: doc.title,
    docDescription: doc.description,
    docTag: doc.tag,
    docAuthor: doc.userID.firstName,
    docDate: doc.uploadDate
}));

res.status(200).json({ numDocs: documents.length, documents });
```

Download


GET /browse/download?docID=67bb0fe4e508e2ec1369103b HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9,de;q=0.8,de-DE;q=0.7
Connection: keep-alive
Cookie: connect.sid=s%3AqqYz0JxImKbyW7ez8YzoacvjPOdHoYL5.Cf1GyHW1j8SjabHxtlbNm8GrbdXa2A84BT39kcEp7b4
Host: localhost:3000
Referer: http://localhost:3000/browse
Client
GET-Request
HTTP
GET-Response
Webserver/browse/download
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: file.fileType
Content-Disposition: attachment; filename="Mathe.pdf"
Content-Length: 1809143
Date: Tue, 25 Feb 2025 23:06:34 GMT
Connection: keep-alive
Keep-Alive: timeout=5
undefined

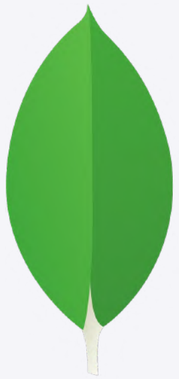
Download - Clientseitig

```
resultsSection.addEventListener("click", (e) => {
    if (e.target.classList.contains("download-btn")) {
        const docID = e.target.getAttribute("data-id");
        window.location.href = `/browse/download?docID=${docID}`;
    }
});
```

Download - Serverseitig

```
const docID = req.query.docID;

const file = await Doc.findById(docID).exec();

if (!file) {
    return res.status(400).send("The document you requested does not seem to exist");
} else {
    res.set({
        "Content-Type": "file.fileType",
        "Content-Disposition": `attachment; filename="${file.originalName}"`
    });

    res.status(200).send(file.file);
}
…
```

Datenbank



mongoDB®

noSQL - Dokumentorientierte Datenbank

Try Pitch

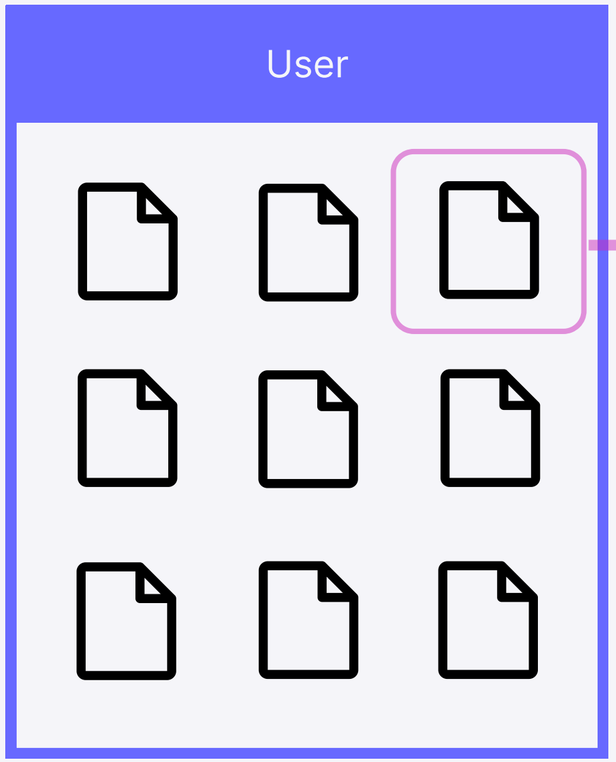
WARUM?

mongoDB®

Open Source
Flexible Schema
Dokumentorientierte Speicherung
Hohe Leistung Speichert Daten im JSON-ähnlichen (BSON - Binary JSON) Format, ermöglicht schnelle Abfrage und Abrufe
Skalierbarkeit Ideal für große Datenmengen
Einfache Nutzung JavaScript-ähnliche Syntax (JSON) Funktioniert mit JavaScript-Frameworks wie Node.js

DHBW Duale Hochschule Baden-Württemberg Ravensburg
StudyBuddy
Try Pitch

Collection & Documents




```
_id: ObjectId('67bd7c9bf5bad201a0f5e5a5')
firstName : "Atussa"
email : "atussa.mrw@yahoo.com"
password : "$2b$10$SKqOyQRYh8plzodOKcAX2ugFtRqKCHcVbA/mh2oqRZc4oVynieVPW"
__v : 0
```

Gespeichert als BSON (Binary JASON)

Verbindung zu mongoDB.

URI

```
const MONGO_URI = process.env.MONGO_URI || "mongodb://localhost:27017/StudyBuddy";
```

Verbindung JSON & mongoDB®

ODM library: Object Document Mapping library

Bibliothek für die Abbildung von Objekten auf Dokumenten

```
const mongoose = require("mongoose");
```

Wie?

Anhand von Methods, welche die Datenmodelle der Datenbank abfragen.

Funktion:

- Abfragen
- Hinzufügen
- Löschen
- Updaten

```
let user = await User.findOne({ email });
if (user) {
   return res.status(400).json({ msg: "User already exists"});
}
```

```
await user.save();
res.status(201).json({ msg: "User registered successfully"});
```

Schema & Model

```
const mongoose = require ("mongoose");

const userSchema = new mongoose.Schema({
  firstName: { type: String, required: true },
  email: { type: String, required: true, unique: true},
  password: { type: String, required: true},
});

module.exports = mongoose.model("User", userSchema);
```

Try Pitch

Schema

```
const mongoose = require ("mongoose");

const userSchema = new mongoose.Schema({
  firstName: { type: String, required: true },
  email: { type: String, required: true, unique: true},
  password: { type: String, required: true},
});

module.exports = mongoose.model("User", userSchema);
```

Schema

- **Properties**
- **Type =**

```
const mongoose = require ("mongoose");

const userSchema = new mongoose.Schema({
  firstName: { type: String, required: true },
  email: { type: String, required: true, unique: true},
  password: { type: String, required: true},
});

module.exports = mongoose.model("User", userSchema);
```

Model

Model umfasst das ganze Schema & erlaubt uns mit der Datenbank-Kollektion zu kommunizieren

```
const mongoose = require ("mongoose");

const userSchema = new mongoose.Schema({
  firstName: { type: String, required: true },
  email: { type: String, required: true, unique: true},
  password: { type: String, required: true},
});

module.exports = mongoose.model("User", userSchema);
```

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

StudyBuddy

Future Features

Like Funktion

Verbesserung der Tags

Book Marks

Filter Optimierung

Try Pitch

Vielen Dank!

StudyBuddy