Aufgabe 1: Client-Server Architekturen (11 Punkte)

a.) Was ist ein Web Stack? (6 Punkte)

Geben Sie eine Definition und nennen Sie drei Beispiele für Web Stacks.

Antwort:

Ein Web Solution Stack ist eine Beschreibung einer Sammlung von Software, die eine Web-Lösung zu implementieren und Betriebssystem betreiben! (✓)

Beispiele:

LAMP: Linux, APache, MySQL, PHP ✓

MEAN: MongoDB, Express, Angular JS, node.js ✓

LYME: Linux, YAML, Mnesia, Erlang ✓

6 P

x benötigt werden, um

b.) Welche Kriterien können relevant sein für die Auswahl eines bestimmten WebStacks für Ihre Projekte? (5 Punkte)

Denken Sie nicht an ein bestimmtes Projekt. Nennen Sie ~~Sie~~ mindestens zwei (allgemeine) Kriterien, die Sie für Ihre Entscheidung zwischen verschiedenen WebStacks für wichtig halten. Begründen Sie kurz warum (oder wann) diese relevant sind.

Antwort:

→ das Kriterium heißt eher "SPA oder sehr dynamische Seite"

- Client-Framework ja/nein! Wenn ja, dann auf jeden Fall MEAN o.ä., da dieses Web Stack Angular beinhaltet. (✓)
- Unterstützung älterer Browser: ~~Sie~~ LAMP oder LYME, da manche ältere Browser oder allgemein ~~Bro~~ Clients ohne JavaScript sonst die Seite nicht rendern können. (✓)

5 P

Aufgabe 2: REST APIs (25 Punkte)

Nehmen Sie an, Sie sollen eine REST-API-Schnittstelle entwerfen für Partyzusagen von Personen.

a.) Ihre REST-Schnittstelle wird beispielsweise die Ressourcensammlung „persons" unterstützen. Daneben werden noch weitere Ressourcen benötigt, um Partyzusagen von Personen zu unterstützen (7 Punkte).

Wählen Sie **eine weitere** sinnvolle Ressource aus und geben Sie für diese Ressource in der folgenden Tabelle an, welche Ressourcen-URLs Sie haben und welche HTTP-Methoden Sie darauf unterstützen, damit es sich um eine REST-Schnittstelle handelt. (Ergänzen Sie in der Tabelle Zeilen oder lassen Sie welche leer, je nachdem, welche Ressource Sie gewählt haben.)

Ressourcen-URL	HTTP-Methoden (ggf. Komma-getrennt)
/party/ ✓	GET ✓, POST ✓, PUT ~~f~~
/party/123 ✓	~~POST~~, PUT ✓, PATCH ✓, GET ✓, DELETE ✓
/party/123/~~persons~~persons	GET, POST, PUT
/party/123/persons/456	PUT, PATCH, GET, DELETE

6 P

b.) Ihre Auftraggeberin möchte, dass Sie die REST-API entsprechend der üblichen REST-Richtlinien so erweitern, dass die ✓ Schnittstelle verschiedene Repräsentationen (Formate) unterstützt.
Erläutern Sie mindestens zwei Möglichkeiten, wie ein Client Ihrer REST-API die Repräsentation(en) angeben könnte. Begründen Sie anschließend kurz, für welche Variante Sie sich entscheiden würden. (8 Punkte).

Antwort:

1. In der URL: http://mezserver.my/api/persons.json ✓

2. Im Body: { returnData: "json", data: {...}} ✓

3. im Header accept: "appl/json" ...

Ich würde mich für die 1. Möglichkeit entscheiden, weil das nicht nur die gängigere Form ist, sondern auch seltsam aussieht, bspw. Per json nach ~~dinnen~~ XML-Daten zu fragen. Bei der 2. Möglichkeit wäre auch nicht* ~~klar~~, in welchem Format die Anfrage kommen muss.

8 P

*ersichtlich

c.) Nehmen Sie an, die REST-Schnittstelle für „persons" wurde durch jemand anderen bereits implementiert.
Auf eine HTTP GET-Anfrage an die URL `http://me2server.my/persons.json` zum Abrufen aller „persons" liefert die REST-Schnittstelle folgende Daten im JSON-Format als Antwort:

```
{
    "items": [{
        "name": "Susan",
        "age": 23,
        "visitedParties": 12,
        "id": 1212,
        "href": "http://me2server.my/persons/1212"
        },
        {
        "name": "Max",
        "age": 24,
        "visitedParties": 1,
        "id": 1313,
        "href": "http://me2server.my/persons/1313"
    }],
    "href": "http://me2server.my/persons"
};
```

Begründen Sie anhand der vorliegenden Daten, um welches REST-Level es sich handelt. (10 Punkte)

Antwort:

Es handelt sich hier um REST-Level 3, da folgende Features vorhanden sind:

- Grundlegende HTTP-Operation im Header ("GET"), nicht im Body ✓
- ~~Mehrere URL~~ Mehrere URLs für verschiedene Ressourcen, anstatt einer URL für alle Ressourcen ✓
- HATEOAS-kompatibel, da weitere mögliche Ressourcen-Abfragen mit 'href' angegeben wurden. ✓

HATEOAS ist nicht komplett.

√ Repräsentation über URL = Level 2 maximal

√ keine Daten im Body

5 P

Aufgabe 3: REST in node.js (12 Punkte)

Ergänzen Sie folgenden JavaScript-Code einer node.js (+ express) Server-Anwendung so, dass der gestartete Webserver beim Senden einer HTTP POST Anfrage an die URL `http://localhost:3000/api/courses/123` immer das gleiche (fest einprogrammierte) JSON-Objekt der Form { error: 'POST not allowed' } zurück sendet. Außerdem soll der HTTP-Statuscode 405 (`Method not allowed`) in der Antwort gesetzt sein. (12 Punkte)
Versuchen Sie Ihren Pseudocode so nah wie möglich an der korrekten Syntax zu orientieren.

```
var express = require('express');

var app = express();
app.post('/api/courses/123', function (req, res, next) {
  res.json({ error: "POST not allowed" }).end();
  next();
});
app.use('/', function (req, res, next) {
  res.status(405).end();
});
app.listen(3000);
```

! nicht wenn next() noch das response-Objekt nutzt für Antwort-Daten!

10 P

Aufgabe 4: Debugging (6 Punkte)

Beim Debugging nehmen Sie die Rolle des Detektivs ein, müssen also systematisch vorgehen um einen Bug zu finden und zu beheben. Sortieren Sie die folgenden sechs (6) Stufen nach der korrekten zeitlichen Reihenfolge. Tragen Sie rechts in der Tabelle die Zahlen 1-6 für die richtige Reihenfolge ein (6Punkte)

Stufen	Reihenfolge (1-6)
Beheben durch Analysieren	5 ✓
Bug beschreiben	1 ✓
Isolieren (Minimized Example)	4 ✓
Dokumentation und Test	6 ✓
Stabile Reproduzierbarkeit herstellen	2 ✓
Hypothesen formulieren	3 ✓

4P

Aufgabe 5: Modularisierung (10 Punkte)

Für die Modularisierung von JavaScript-Code (ECMAScript 5) haben sich zwei wesentliche Konzepte etabliert: AMD und CommonJS.

a.) Geben Sie zu folgenden Aussagen an, ob diese jeweils zu AMD oder zu CommonJS passen (jeweils zwei).
Achtung: Zwei Aussagen passen entweder zu beiden oder keinem. Diese beiden bitte als „beide/keine" ankreuzen. (6 Punkte)

Antwort:

AMD	CommonJS	Beide/ keine	Aussage
☐	☐	☒	…erlaubt das Laden benötigter Module, wenn möglich, unabhängig voneinander
☐	☒	☐	…erlaubt das Laden benötigter Module nacheinander
☒	☐	☐	…erlaubt die Definition mehrerer Module in einer Datei zusammengefasst.
■	☐	☒	…benutzt anonyme Funktionen zur Isolierung von Modul-Namensräumen.
☐	☒	☐	…wird hauptsächlich für server-seitige Modularisierung benutzt.
☒	☐	☐	…beachtet beim Laden die Abhängigkeiten, so dass benötigte Module immer verfügbar sind.

4P

b.) Zu welchem der beiden Ansätze (AMD oder CommonJS) gehört der folgende Code-Baustein? Begründen Sie kurz Ihre Antwort. (4 Punkte)

```
'use strict';
//...
var proto = require('./application');
var req = require('./request');
var res = require('./response');
//...
module.exports = createApplication;
exports = module.exports;
//...
```

Antwort:

Dieser Code-Baustein gehört zu commonJS, da im Code „exports" vorhanden ist. Genauso verwendet man Module in node.js, die auch auf Grundlage von commonJS spezifiziert wurden.

4P

Aufgabe 6: NoSQL (10 Punkte)

Zur Anbindung einer NoSQL mongoDB in Ihrer node.js Programmierung einer REST-API gibt es mehrere Alternativen.

Welche Unterscheidungsmerkmale (Vor-/Nachteile) sehen Sie beim Einsatz von mongoDB mit Modul mongojs im Vergleich zu mongoDB mit Modul mongoose? (10 Punkte)

Nennen Sie mindestens vier Unterschiede. Begründen Sie **kurz** je Unterschied warum Sie in diesem Punkt mongojs oder mongoose bevorzugen würden (oder warum es für Sie egal ist).

Antwort:

- Kleines Projekt: eher MongoJs
  - einfacher zu bedienen sicher?
  - einfache Schemata ?
  - Weniger Code ? für was ?

- großes Projekt: eher Mongoose
  - sehr detaillierte / genaue Nutzung von Features der MongoDB
  - komplexe Schemata ✓①
  - Joins / Relationen möglich ✓②
  - Validierung von Daten ✓③

3P

Aufgabe 7: Backbone.js (14 Punkte)

a.) Im Folgenden sind Ausschnitte von JavaScript-Quellcode einer Backbone-Anwendung zu sehen. Geben Sie für jeden Code-Ausschnitt kurz an, ob es sich um einen Teil einer Collection, eines Models, eines Router oder eines Views in Backbone.js handelt. (5 Punkte)

Code-Abschnitt 1:

```
11: …
12: … : {
13:       '': 'main',
14:       '/image/:id/edit': 'editImage'
15: },
16: editImage: function(id){
17:       …
18: }
19: …
```

Code-Abschnitt 2:

```
21: …
22: idAttribute: '_id',
23: defaults: {
24:       serverPath: '',
25:       creator: null,
26:       timestamp: ''
27: },
28: …
```

Code-Abschnitt 3:

```
31: …
32: model: ImageModel,
33: url: '/Images',
34: …
```

Antwort:

3: View
1: Model
2: Router

1: Router
2: Collection
3: View

1: Router ✓
2: Model ✓
3: Collection ✓

5P

b.) Backbone.js erlaubt einige Variationen bei der Erstellung von Views. Im Unterricht wurden diese Enhancer, `Creator` und `Delegator` Views genannt.
Erläutern Sie, für welche Zwecke dieser drei Varianten eingesetzt werden und geben Sie dabei an, welche der möglichen Attribute von BackboneView dabei verwendet oder nicht verwendet werden. Eine Liste möglicher Attribute ist für Sie aufgelistet. (9 Punkte)

Mögliche Attribute können Sie dem Code-Beispiel entnehmen:

```
41: var FiktiverView = Backbone.View.extend({
42:     el:…,
43:     tagName: …,
44:     className: …,
45:     id: …,
46:     template: …,
47:     model: …,
48:     collection: …,
49:     events: …,
50:     initialize: …,
51:     render: …,
52: });
```

Antwort:

OP

Aufgabe 8: Idempotenz (5 Punkte)

Bei REST-Schnittstellen werden verschiedene Operationen mittels HTTP durchgeführt. Erläutern Sie was eine idempotente Operation ist. Geben Sie dann an, welche der HTTP Methoden idempotent sein müssen, welche es sein können und welche es nicht sind. (6 Punkte)

Antwort:

Idempotent bedeutet, dass ein Datensatz ersetzt und nicht erweitert wird. f

Idempotenz-Eigenschaft	HTTP-Methoden (ggf. Komma-getrennt)
MUSS sein	DELETE, GET
KANN sein	PUT f √PATCH
Ist NICHT vorhanden	POST, PATCH f

3 P

Aufgabe 9: Mobile Cross-Plattform Entwicklung (6 Punkte)

Nehmen Sie an, Sie wollen mit Ihrem Wissen zu Webtechnologien, wie HTML, CSS und JavaScript eine Anwendung (App) für mobile Betriebssysteme implementieren. Welche Kriterien wägen Sie ab, bei der Entscheidung zwischen den folgenden beiden Alternativen (I) und (II)? Nennen Sie drei Kriterien und jeweils ob dies ein Vor-/ oder Nachteil für (I) oder (II) ist. (6 Punkte)

(I) **Hybride Lösung** (bspw. mit PhoneGap)
oder
(II) **Laufzeit-Interpretern** (bspw. mit Titanium)

Antwort:

Geschwindigkeit: Titanium (II): Vorteil, PhoneGap (I): Nachteil
Code-Wiederverwendbarkeit: PhoneGap (I): Vorteil, Titanium (II): Nachteil
Sensoren: Titanium (II): leichter Vorteil, Phonegap (I): leichter Nachteil

6 P