LU15b – Node.js/Express-Projekt einrichten

Learning Objectives

- Sie können **Node.js** installieren und die Installation prüfen.
- Sie können in **WebStorm** ein neues Node.js-Projekt für das Backend anlegen.
- Sie verstehen die Rolle von `package.json`.
- Sie können **Express** als Bibliothek zu Ihrem Projekt hinzufügen.

Nodejs und express installieren und konfigurieren

Dieses Video veranschaulicht den Installationsprozess von Nodejs und express und zeigt das Erstellen eines Backend-Servers.

Schritt 1 – Node.js installieren

1. Öffnen Sie nodejs.org und laden Sie die **LTS-Version** von Node.js herunter.
2. Installieren Sie Node.js mit dem Installer (Standard-Optionen genügen).
3. Nach der Installation:
    1. Öffnen Sie ein Terminal (PowerShell, macOS Terminal, ...).
    2. Prüfen Sie die Versionen mit:

```
node -v
```

Wenn beide Befehle eine Versionsnummer anzeigen, sind Node.js und npm korrekt installiert.

Schritt 2 – Neues Projekt in WebStorm anlegen

1. Starten Sie **WebStorm**.
2. Wählen Sie **`New Project`**.
3. Geben Sie dem Projekt einen sinnvollen Namen statt `untitled` (z.B. `m290-backend-demo`).
4. Create Git-Repository: optional – nur aktivieren, wenn Sie wissen, was Git ist und damit arbeiten möchten.
5. Language: **JavaScript**.
6. WebStorm erstellt ein neues Projekt. Meistens wird automatisch eine Datei `package.json`und `index.js` erzeugt.

Typische Dateien:

- `index.js` → Haupt-JavaScript-Datei, in die Sie Ihren Server-Code schreiben.
- `package.json` → enthält Informationen über das Projekt (Name, Version, Scripts, Abhängigkeiten).
- `node_modules/` → wird später automatisch erstellt, sobald Sie Pakete installieren.

Fallback für andere IDEs (z.B. VS Code)

Wenn Ihre IDE **nicht automatisch** eine `package.json` erstellt, können Sie das Projekt im Terminal initialisieren:

```
npm init -y
```

In WebStorm ist dieser Schritt in der Regel **nicht nötig**, da `package.json` beim Anlegen des Projekts erzeugt wird.

Schritt 3 – Express installieren

Öffnen Sie das Terminal im Projektordner (unten links in WebStorm).

Installieren Sie Express mit:

```
npm install express
```

Dadurch passiert:

- Der Ordner `node_modules/` wird erstellt (falls noch nicht vorhanden).
- In `package.json` wird unter `dependencies` ein Eintrag für **Express** angelegt.
- Sie können nun in Ihrem Code Express verwenden.

Schritt 4 – ES-Module aktivieren (''import … from …'')

Im Unterricht verwenden wir im Beispiel die moderne `import`-Syntax:

- `import express from 'express';`

Damit Node.js diese Syntax versteht, müssen Sie in `package.json` den Typ auf `module` setzen.

Beispiel für `package.json` (auszugweise):

```
{
  "name": "m290-backend-demo",
  "version": "1.0.0",
  "type": "module",
  "main": "index.js",
```

```
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "dependencies": {
    "express": "^5.2.1"
  }
}
```

Wichtig:

- Löschen oder benennen Sie `package.json` **nicht** um – sonst funktionieren npm und Ihre Abhängigkeiten nicht mehr.

Schritt 5 – Überblick: Dateien im Projekt

- **`index.js` (oder `app.js`):**
  - Einstiegspunkt Ihrer Anwendung.
  - Hier initialisieren Sie Express, definieren Routen und starten den Server.
- **`package.json`:**
  - Enthält Scripts (z.B. `start`, `dev`) und die Liste Ihrer Bibliotheken.
  - Wird von `npm` verwendet, um die richtigen Pakete zu installieren.
- **`node_modules/`:**
  - Enthält den gesamten Code der installierten Pakete.
  - Wird von `npm` automatisch verwaltet.
  - Sie müssen (und sollten) diesen Ordner **nicht manuell bearbeiten**.

In der nächsten Seite bauen Sie nun auf dieser Basis Ihren ersten **Express-Server** auf.