

# Webbasierte Anwendungen

Web-Ueb-01 (15.04.2024)

Wir arbeiten in dieser Veranstaltung **nur unter Linux** und durchgängig mit dem Editor “VisualStudio Code” <https://code.visualstudio.com/>, den Sie bereits von “Programmieren 3” kennen. Er wurde unter der Leitung des großmächtigen Erich Gamma entwickelt, unterstützt **alle** in der Veranstaltung benötigten Sprachen und Formate (Python, Java, Gradle, TypeScript, JSON, XML, HTML, CSS) sowie die Frameworks, die wir im Verlauf brauchen werden.

Die Praktikumsstunden sind für die aktive Bearbeitung des aktuellen Übungsblattes gedacht. Die benötigte Software ist auf den Pool-PCs verfügbar. Sie können auch Ihren privaten Laptop nutzen, die betreute Übungszeit aber dann nicht mit Systemupdates und anderen Installationstätigkeiten verschwenden, sondern erkennbaren Fortschritt bei der Bearbeitung der Aufgaben machen (und sich helfen lassen, wenn es dazu Fragen gibt).

## Aufgabe 1 (Browser: Entwickler-Tools)

Webbrowser enthalten oft hilfreiche Entwicklungswerkzeuge. Starten Sie bitte den Browser `chromium-browser` und öffnen Sie die URL `http://if-schleife.de`. Öffnen Sie die **Entwicklertools** mit der Tastenkombination `strg+shift+i`. Wählen Sie den **Reiter “Netzwerkanalyse” (oder Network)** aus und laden Sie die URL erneut (normaler Browser-Reload). Sie sehen die durchgeführten **Netzwerkanfragen**. Wenn Sie einen Eintrag anklicken, erhalten Sie unter dem Reiter “Kopfzeilen” die **HTTP Anfrage- und Antwort-Headerfelder** und weitere Werte. Rufen Sie nun bitte z.B. die HSRM-Homepage `https://www.hs-rm.de` ab und sehen Sie sich an, wieviele HTTP-Anfragen durch den Abruf der einen URL nachfolgend ausgelöst wurden.

Die Browser-Entwicklertools sind nützlich, man sollte also wissen, dass es sie gibt und eine grobe Idee haben, was sie bieten. Lassen Sie die Nutzung selbständig in Ihre Bearbeitung der folgenden Aufgaben einfließen (und verbringen Sie **nicht zu viel Zeit** mit dieser Übersichtsaufgabe, die interessanten Sachen kommen nachfolgend).

## Aufgabe 2 (Spring Boot Initializr, Umgang mit Gradle in der Shell)

Bitte begeben Sie sich auf `https://start.spring.io` und erzeugen Sie sich mit dem **Spring Boot Initializr** einen Zip-Archiv-Download für ein Spring-Boot-Projekt (für Build mit **Gradle**) mit **Java 21**, **Spring Boot 3.2.x** (neuestes stabile Version, also in der Auswahl “ohne Klammern dahinter”, d.h. kein SNAPSHOT und keine “M” Milestone-Zwischenversion) und unter Verwendung der **folgenden SpringBoot-Starter Dependencies**: “Spring Web” (**nicht** *Spring Reactive Web* oder *Spring Web Services*) und “Spring Boot Devtools”. Achten Sie auf die entsprechenden Einstellungen im Initializr. Für das Feld “Group” können Sie sich einen (hoffentlich eindeutigen) Java-Paketpfad-Anfang für alle Ihre Projekte überlegen (z.B. `de.hsrm.mi.web`), der Name “Artifact” wird an “Group” angehängt und ermöglicht die Unterscheidung verschiedener Ihrer Projekte innerhalb derselben “Group”.

Verwenden Sie auf der Webseite den **“Explore”-Button** – hier können Sie sich schon einmal die auf Grundlage Ihrer Auswahl generierte Gradle Builddatei ansehen, welche die Informationen zum Bauen Ihres Projekts enthält (benötigte *Dependencies*, Java *sourceCompatibility* usw).

Mit dem **“Download”-Button** können Sie ein vorbereitetes zip-Archiv mit einem Grundgerüst für ein SpringBoot-Projekt auf Basis Ihrer Auswahl herunterladen. Entpacken Sie das Zip-Archiv bitte und sehen Sie sich die **Projektstruktur** an (Sie können dazu z.B. in einer Shell das Linux-Kommando `tree` verwenden, sofern es bei Ihnen installiert ist).

Im Wurzelverzeichnis Ihres Projekts finden Sie das Shell-Skript `gradlew` (“gradle wrapper”). Probieren Sie `./gradlew bootRun`, um das Projekt zu **compilieren** und direkt zu **starten**. Machen Sie sich bitte bei der Gelegenheit mit der **Struktur der Log-Ausgabe** von Spring Boot vertraut. Durch das Einbinden der “Spring Web”-Abhängigkeit integriert SpringBoot automatisch einen Servlet-Container (Tomcat), den Sie unter `http://localhost:8080` erreichen. Da Ihr Projekt bis auf das generierte Grundgerüst jedoch noch “leer” ist, erhalten Sie nur eine “Whitelabel Error Page”, aber immerhin spricht Ihr Tomcat schon mit Ihnen.

Brechen Sie nun bitte die laufende Anwendung ab und erzeugen Sie mit `./gradlew build` ein “stand alone” **ausführbares jar-Archiv** der Anwendung. Sie finden das jar im Unterordner `build/libs`. **Starten** Sie die Anwendung in diesem Ordner (ohne `gradlew`) direkt mit `java -jar jardateiname.jar`. Abbrechen können Sie das laufende Serverprogramm wie gewohnt mit `strg+c`.

Nachdem nun durch die verschiedenen Schritte eine Reihe generierter Dateien angelegt wurden (jar, .class-Dateien etc, bitte nochmal mit `tree` einen Überblick verschaffen), können Sie Ihr Projekt mit `./gradlew clean` **automatisch aufräumen** lassen. Sehen Sie sich bitte mit `./gradlew tasks` an, welche **weiteren Subkommandos** Gradle für Ihr Projekt anbietet.

**Hinweis:** Sollten Sie diese Übung auf einem gemeinsam genutzten Rechner wie `linux001` machen, kann es passieren, dass schon eine andere Anwendung die Portnummer 8080 auf “localhost” belegt hat (siehe oben – die 8080 aus `http://localhost:8080`). Sie erhalten dann eine Fehlermeldung und Ihre Anwendung startet nicht. Auf einer konkreten Netzwerkschnittstelle (hier das Loopback-Interface von `localhost`) kann immer **nur eine** laufende Anwendung **gleichzeitig** eine Portnummer belegen. Unter Linux können Sie sich (sofern installiert) mit dem Kommando `netstat -tplan` einen Überblick über alle aktuell belegten (TCP-)Ports verschaffen (Spalte “Local Address” – wie das im Detail funktioniert lernen Sie in der Rechnernetze-Vorlesung).

Sie können die eigene Portnummer jederzeit ändern, indem Sie in Ihrem Projekt die automatisch angelegte Konfigurationsdatei `./src/main/resources/application.properties` öffnen und einen Eintrag der Form `server.port=9090` mit der gewünschten Portnummer hinzufügen. Starten Sie danach Ihre Anwendung neu. Sie erreichen Sie danach unter `http://localhost:9090`

Wenn Sie nur für einen Start oder zum Testen die `server.port`-Property ändern wollen, geht das auch, indem Sie dem `gradlew bootRun`-Kommando eine entsprechende Kommandozeilenoption hinzufügen (Linux):  
`./gradlew bootRun --args='--server.port=9876'`

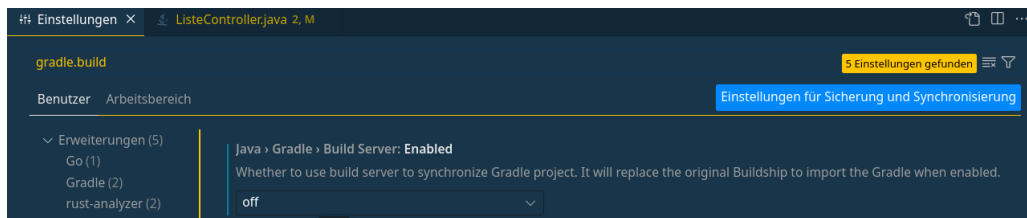
## Aufgabe 3 (Spring Boot Projekt in VSCode, einfacher Web-Controller)

Bitte **installieren** Sie folgende **Erweiterungen** in Ihrem VS Code Editor und starten Sie ihn danach neu:

- Java Extension Pack – `vscjava.vscode-java-pack`
- Spring Boot Extension Pack – `vmware.vscode-boot-dev-pack`
- Gradle for Java – `vscjava.vscode-gradle`
- Error Lens – `usernamehw.errorlens`
- Git Lens – `eamodio.gitlens`

Wer cool ist oder einfach keine Lust hat, die Extensions in der VSCode-GUI zu installieren (das kann ja jeder), kann dies in einer Shell mit `code --install-extension extensionid` tun, wobei `extensionid` die in der Aufzählung oben hinter den Namen in Typewriter-Schrift angegebenen Bezeichnungen sind.

Sie können Ihre Spring-Anwendung von VSCode aus starten (`f5` in einem Sourcecode-Fenster) und auch debuggen. Bei Code-Änderungen sollte die laufende Spring-Applikation automatisch neu gestartet werden, sofern Sie die Spring Devtools in Ihr Projekt eingebunden haben (s.o., `build.gradle`). Zur Optimierung der Durchführungsgelung sollten Sie in Ihrer VSCode-Konfiguration sicherstellen, dass die Eigenschaft `java.gradle.buildserver.enabled` auf `off` gesetzt ist. Gehen Sie dazu in die VSCode-Einstellungen (`strg+,`) und suchen Sie mithilfe des Suchfelds nach `buildserver.enabled`.



Bei der Gelegenheit können Sie auch in den Einstellungen nach dem Stichwort `telemetry` suchen und diese jeweils abschalten, wenn Sie sich nicht als Datenspender zur “Verbesserung der Benutzer- und anderer Erfahrungen” sehen.

## 0. Aufgabe

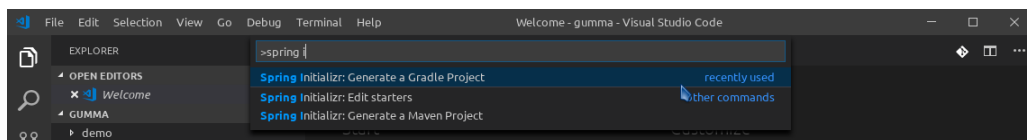
Bekanntlich ist der Body-Spaß-Index BSI eine populäre Kennzahl, bei aus den drei leicht bestimmbar Parametern *Körpergröße*  $k$  (in cm), *Schuhgröße*  $s$  und *Semesterzahl*  $z$  ein mehr oder weniger aussagekräftiger Indikator bestimmt wird. Die gebräuchliche Formel dazu lautet

$$bsi(k, s, z) = 100 \cdot \frac{k}{s \cdot z}$$

Trotz bekannter Ungenauigkeiten und gelegentlich geäußerten Zweifeln an der Wissenschaftlichkeit ist sie u.a. deshalb so beliebt, weil sie ohne komplizierte Sachen auskommt. Wir wollen nun eine erste SpringMVC-Anwendung bauen, welche Nutzern die Berechnung des individuellen BSI-Wertes anbietet.

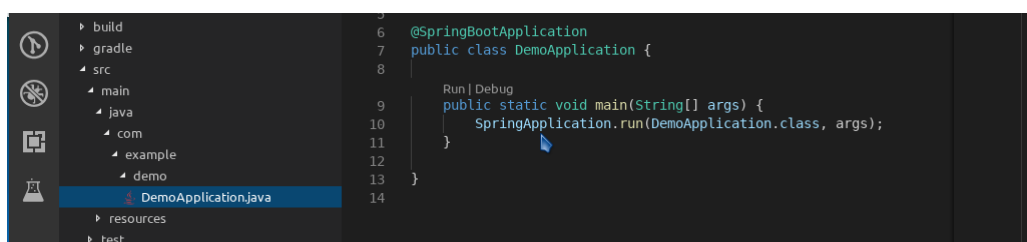
## 1. SpringBoot-Projekt anlegen

Ein wichtiges Tastaturkürzel in VSCode ist `strg` + `shift` + `p`. Es öffnet eine Eingabezeile (“**Command Palette**”), in der Sie schnell und bequem Editor-Kommandos absetzen können. Wir legen nun ein neues SpringBoot-Projekt direkt in VSCode an




Öffnen Sie dazu die Command Palette und tippen Sie `spring ini`. Der Editor bietet Ihnen alle Kommandos an, welche die eingegebenen Wörter enthalten. Wählen Sie bitte `Spring Initializr: Generate a Gradle Project` aus und beantworten Sie die nachfolgenden Fragen wahrheitsgemäß (analog zur Verwendung der Initializr-Webseite in der vorigen Aufgabe), um ein SpringBoot-Java-Projekt mit den “Dependencies” `Spring Web`, `Spring Boot DevTools` und zusätzlich `Thymeleaf` anzulegen. Zum Abschluss erscheint ein Dateidialog, mit dessen Hilfe Sie ein leeres Zielverzeichnis anlegen bzw. auswählen können, in dem die Projektvorlage erzeugt werden soll.

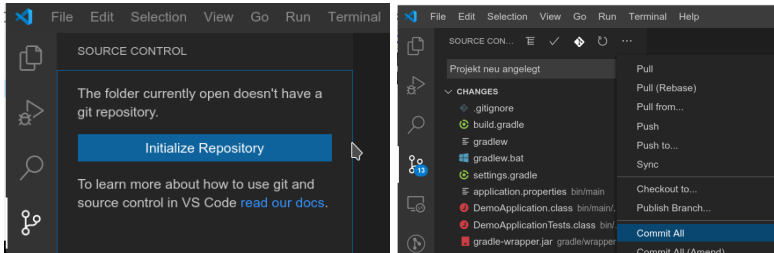
Sehen Sie sich nun innerhalb VSCode im Projekt um und öffnen Sie die Datei `...Application.java` im Editor. Sobald sich VSCode fertig initialisiert hat, sehen Sie zwei Links “Run | Debug” über der `main()`-Methode. Sie können Ihre Anwendung nun z.B. mit einem Klick auf `Debug` starten. Es ist auch möglich (und bequem), das Projekt mit der Taste `F5` zu starten (ebenfalls im Debug-Mode).




## 2. Lokales Git-Repository für Projekt initialisieren und füllen

Sobald Sie Ihr Projekt angelegt haben, sollten Sie es unter Versionskontrolle stellen und nach jedem Schritt die Änderungen committen (also eher viele Commits machen – schon zum Üben). Hierzu müssen Sie erst ein leeres Git-Repository initialisieren (“git init”). Gehen Sie dazu in die Git-Ansicht von VSCode (das ist in der linken Button-Leiste das Icon mit der merkwürdigen Gabel  oder tippen Sie `strg+shift+g`. Dort wird Ihnen dann ein blauer “Initialize Repository”-Button angeboten.

Spring Boot legt Ihnen schon eine erste `.gitignore`-Datei an, so dass in der VSCode-Git-Ansicht gleich nur “sinnvolle” Dateien angeboten werden. Sie müssen also die geänderten Dateien nicht dateiweise mit “+” stagen (also zum Commit bereitstellen), sondern können nach Eingabe eines Commit-Kommentars in das kleine Textfeld über “Changes” gleich “Commit All” aus dem Dreibommelmanü auswählen. Dasselbe können Sie (kürzer) erreichen, indem Sie nach dem Eintrag Ihres Änderungskommentars `strg+Enter` drücken.



Denken Sie daran, dass Sie nach dem Commit immer noch in der (durch den Commit geleerten) Git-Ansicht sind. Wenn es Ihnen hier zu einsam wird, können Sie mit dem Explorer-Button  oder `strg+shift+e` in die Explorer-Ansicht für Ihre Projektstruktur zurückkehren.

## 3. Ein HTML-Template hinzufügen

Bitte legen Sie im Ordner `src/main/resources/templates` ein Thymeleaf-HTML-Template `bsi.html` mit einem HTML-Formular an, das den Benutzer nach drei Eingabewerten fragt. Als action des “OK”-Submit-Buttons können Sie einen Leerstring oder den Pfad `/bsi` angeben, als Einsende-Methode des Formulars nehmen Sie bitte `POST`. Wenn der Benutzer das Formular abschickt, erfolgt also ein HTTP POST mit den Formulardaten auf den Pfad `/bsi` des Servers, von dem das Formular abgerufen wurde.

Label	<INPUT name="..." .../>
Körpergröße in cm	kgroesse
Schuhgröße	sgroesse
Semesterzahl	semzahl

Körpergröße in cm	<input type="text" value="172"/>
Schuhgröße	<input type="text" value="48"/>
Semesterzahl	<input type="text" value="17"/>
<input type="button" value="OK"/>	

## 4. Controller und HTTP-GET-Request-Handler für Pfad “/bsi” anlegen

Nun haben wir ein HTML-Formular in unserem Server-Projekt, können es aber noch nicht abrufen. Das ändert sich umgehend. Spring Boot hat bereits eine Klasse `...Application.java` unter `src/main/java/...ihrpaketpfad.../` angelegt. Beachten Sie bitte bei Imports, dass `src/main/java` selbst *nicht* Teil des Klassen-/Paketpfads ist, also so etwas wie `src.main.java` in import-Statements **nichts** zu suchen hat.

Bitte fügen Sie Ihrem Projekt eine neue Controller-Klasse `BSIController` hinzu, die unter der URL-Pfadkomponenten `/bsi` einen Handler für HTTP GET bereitstellt. Diese Methode soll nur den (“View-”)Namen der HTML-Template-Datei, also “bsi” (nicht “bsi.html”), als String zurückgeben, damit die zugehörige “bsi.html” als Template verwendet und dargestellt wird, wenn `http://localhost:8080/bsi` per HTTP GET abgerufen wird, diese URL also z.B. in die Browser-URL-Zeile eingetragen wurde.

Ihre Datei `bsi.html` soll keine einfache statische HTML-Datei sein, sondern von Spring MVC nachfolgend als **Template** für den Templateprozessor *Thymeleaf* genutzt werden können. Überprüfen Sie dazu bitte,

ob Sie im `<HTML>`-Element an das zur Markierung erwartete XML Namespace-Deklaration für `th`, also `xmlns:th="http://www.thymeleaf.org"`, gedacht haben.

Stellen Sie bitte auch sicher, dass die neue Controller-Klasse im **selben Ordner** wie die `Application`-Klasse oder einem **Unterordner davon** liegen muss, um ohne Weiteres gefunden zu werden.

## 5. Testen und verstehen

Testen Sie bitte, ob Sie **auf demselben Rechner**, auf dem Ihre Spring-Anwendung läuft, die Seite über die URL `http://localhost:8080/bsi` abrufen können.

Sobald das der Fall ist, haben Sie Ihren ersten “Spring MVC Round Trip” erfolgreich durchlaufen. Es ist wichtig, nicht nur die Gebrauchsanweisung herunterzutippen, sondern die einzelnen Schritte des Spring-MVC-Verarbeitungszyklus und wo sie sich im Code widerspiegeln zu verstehen.

Bitte sehen Sie sich dazu auch noch einmal die Übersichtsgrafik im Foliensatz an: Eine **HTTP GET-Anfrage** wurde vom **Tomcat** Servlet-Container entgegengenommen, über den Pfad `/bsi` aus der URL der zugehörigen **Requesthandler-Methode** Ihrer **BSIController**-Klasse zugeordnet, diese hat als Ergebnis den Namen einer **“View”** (hier des darzustellenden **HTML-Templates** `bsi`) zurückgegeben und Spring MVC hat daraufhin dieses mit *Thymeleaf* das (hier noch nicht weiter bearbeitete) HTML-Template an den Aufrufer zurückgeliefert, der das Formular nun in seinem Browser sieht.

Vergegenwärtigen Sie sich bitte auch, wo in der Projekt-Verzeichnishierarchie welche Teile der Anwendung hingehören.

## 6. Formulardaten entgegennehmen und verarbeiten

Wie wir gesehen haben, übermittelt der Nutzer die eingegebenen Formulardaten mit einem HTTP-POST-Request – so steht’s im Template-HTML. Fügen Sie bitte eine entsprechende weitere Handler-Methode zu Ihrer Controller-Klasse hinzu, welche die Parameter `kgroesse`, `sgroesse` und `semzahl` aus dem eingehenden Request (jeweils als `int`) entgegennimmt, daraus den BSI berechnet (auch `int`) und das Ergebnis in einem Model-Attribut `bsiwert` ablegt, das in der Antwort unter dem (dann wieder leeren) Formular ausgegeben wird (siehe Abbildung). Ergänzen Sie dazu Ihr Template `bsi.html` um eine Ausgabe von `bsiwert` unter dem OK-Button. Die Zeile darf auch (ohne konkreten Zahlenwert) beim initialen Abruf des Formulars erscheinen, muss also nicht ein-/ausgeblendet werden.

Das Diagramm zeigt den Prozess der BSI-Berechnung in zwei Schritten. Links ist ein Formular mit den Eingabefeldern 'Körpergröße in cm' (Wert: 172), 'Schuhgröße' (Wert: 48) und 'Semesterzahl' (Wert: 17). Darunter befindet sich ein 'OK' Button und der Text 'Ihr BSI-Index beträgt'. Ein Pfeil zeigt nach rechts zu einem zweiten, identischen Formular, bei dem das Ergebnis 'Ihr BSI-Index beträgt 21' unter dem 'OK' Button angezeigt wird.

## 7. Mit der Umgebung vertraut machen

Probieren Sie bitte auch aus, ob Sie **Änderungen im Sourcecode** (bei laufender Anwendung) machen können. Ändern Sie dazu z.B. das HTML-Template und/oder die BSI-Berechnung und testen Sie (nach Abspeichern) durch eine neue Abfrage im Browser, ob die Anwendung nach Codeänderung korrekt neu compiliert und gestartet wurde (bei Template-Änderungen brauchen keinen Neustart).

Setzen Sie auch einen **Breakpoint** in einer der Rechenmethoden und lassen Sie die Anwendung auf den Breakpoint laufen. Machen Sie sich dabei mit dem **Java-Debugger** in Visual Studio Code vertraut, z.B. auf <https://code.visualstudio.com/docs/editor/debugging>

Auch die **Git-Funktionalität** sollten Sie sich näher anschauen, z.B. auf <https://code.visualstudio.com/docs/editor/versioncontrol>

Auch diese “richtige” Anwendungen können Sie ohne Entwicklungsumgebung mit `./gradlew bootRun` aus der Shell starten bzw. mit `./gradlew build` ein ausführbares jar-File erzeugen, das Ihre gesamte Anwendung einschließlich Webserver enthält. Probieren Sie es ruhig mal aus.

## Aufgabe 4 (Vorbereitung Projektarbeit)

Das (bis Semesterende wöchentlich zu erweiternde) Projekt beginnt nächste Woche, heute laufen wir uns zunächst etwas warm. Bitte bestellen Sie, sofern noch nicht geschehen, daher bis **spätestens So, 21.04.2024** je (max) Zweier-Gruppe per Mail an `wolfgang.weitz@hs-rm.de` unter Angabe Ihrer **Pool-Login-Namen** ein Git-Repository auf unserem RhodeCode-Server. Darüber finden die wochenweisen Abgaben zu den Übungen über Git statt, deren fristgerechte Abgabe und vorführbares Funktionieren erforderlich ist und Grundlage für die Zwischen- und Abschlussbesprechungen sowie zum Erwerb der Praktikumsleistung erforderlich sind.

Wir verwenden für das Projekt in diesem Praktikum **bis Semesterende** standardmäßig die aktuelle “long time support”-Version von Java, das ist z.Zt. **Java 21**. Diese Version ist der augenblickliche Standard auf den Hochschulrechnern. Durch Setzen der Property `sourceCompatibility = '21'` in Ihrer `build.gradle`-Datei wird dies unterstützt. **Abgaben, die nicht unter Java 21 compilieren** bzw. in der `build.gradle` einen höheren oder unverträglichen `sourceCompatibility`-Level fordern, gelten als nicht übersetzbar und **zählen als nicht abgeben**.

- In “Softwaretechnik” gab es eine Einführung in das Versionskontrollsystem Git. Dieses benötigen wir auch für die Projektbearbeitung in diesem Modul. Falls Sie sich um Umgang mit Git unsicher fühlen, sollten Sie die Zeit bis zum Projektbeginn nutzen, um noch etwas zu wiederholen bzw. zu üben. Links zu einer Sammlung von Anleitungen finden Sie z.B. unter <https://scm.mi.hs-rm.de/git/>
- Bitte machen Sie sich auch mit der Verwendung von Git in VS Code näher vertraut. In der *Linksammlung Werkzeug: Git, HTML, VSCode* im read.MI finden Sie weitere VS Code Tutorials, speziell auch zum Thema Git.
- Die Lösungen müssen spätestens 12 Tage nach Erscheinen des zugehörigen Aufgabenblatts auf dem master-Branch Ihres Repositories vorliegen (Bewertungsgrundlage). Eigene Beiträge müssen im Git nachvollziehbar sein. Um das Autorenfeld der Commits korrekt auszufüllen, sind dazu der eigene **Klarname** (Vorname Nachname) und die **HSRM-Mailadresse** korrekt in die Konfigurationsdatei `$HOME/.gitconfig` einzutragen und für alle eigenen Commits *einheitlich* zu verwenden. Dies gilt insbesondere auch für Commits, die auf privaten Rechnern gemacht werden.
- Die im Git abgegebenen Lösungen müssen **auf den Poolrechnern unter Linux** unverändert lauffähig sein. Nach einem Auschecken des Projekts aus Ihrem Git Repository muss der Spring-Projektteil unmittelbar per `./gradlew clean bootRun` mit Java 21 übersetz- und ausführbar sein, später der Vue-Teil analog mit `npm install; npm run dev`. Wenn Sie auf eigenen Rechnern und nicht unter Linux entwickeln sollten, stellen Sie rechtzeitig durch Testen in der Hochschul-Pool umgebung sicher, dass diese Bedingungen für Ihre Abgabe erfüllt sind.
- Von allen Teammitgliedern wird ein ausgewogener, kontinuierlicher und anhand eigener Git-Commits nachvollziehbarer Anteil an der Projektarbeit erwartet (“Pair Programming” ist keine Begründung, wenn das nicht der Fall ist), d.h. beide kennen sich mit dem Projekt insgesamt aus und können es erklären bzw. auf Zuruf kleinere Anpassungen vornehmen.
- Nach größeren zusammenhängenden Projektabschnitten wird es Besprechungstermine geben, bei denen der Projektstand vorgeführt und von Ihnen erläutert wird.
- Mit der Abgabe Ihres Codes im Git (Commit) bestätigen Sie, dass Sie die eingereichten Lösungsteile vollständig selbst entwickelt haben (keine generative KI, kein Plagiat von anderen Personen etc.). Online-Doku und ergoogelte Erläuterungen/Tutorials sind natürlich ok.