Verwaltungsgericht Cottbus
Vom-Stein-Straße 27
03050 Cottbus
Telefax: 0355 4991-6499
Doppelte Ausführung
AZ: VG 8 K 518/21

Von:
Marcel Langner
[redacted]

Sehr geehrte Vorsitzende Richterin, Sehr geehrte Kammer,
ich erhielt am 05.08.2021 Ihr Schreiben vom 02.08.2021 mit Anhang der Hochschule und Bitte um eventuelle Stellungnahme. Vielen Dank für diese Möglichkeit.
Vorweg die mir nun notwendig erscheinenden Anträge, die Sie begründet in den folgenden Erläuterungen finden.

Ich beantrage, dass die an der Entwicklung beteiligten Programmierer und Verantwortlichen an Eides Statt versichern (§98 VwGO i.V.m §452 ZPO), dass keine anderen vorherigen Versionen des Quellcodes oder der Anwendung existieren. Weder in Backups noch auf anderen Systemen oder Datenträgern.

Ich beantrage die Hochschule zu verpflichten, Auskunft nach dem von mir beschriebenen Verfahren in Anlage 1 meines Schreibens vom 26.06.2021 zu erteilen (§7 (3) AIG).

Ich beantrage die Herausgabe aller ausgeführten Kommandozeilenbefehle (bzw. auch Scripte) und die genaue Beschreibung eventueller Folgeschritte, die zu der von der Hochschule bereitgestellten PDF geführt haben.

Zu A (1-4)
Ich halte es nicht für sachdienlich auf die hier gemachten Aussagen zur Eigenwerbung der Hochschule oder zu meiner Person einzugehen. Sollten Sie, wertes Gericht, dies anders sehen, bitte ich um Rückmeldung. Ich würde dann eine Richtigstellung verfassen, da gerade die von der Hochschule mir als vermutlich nachteilig angedachten Aussagen nicht den Tatsachen entsprechen. Die Argumentation der Hochschule zeigt klassische ad-hominem und Argumentum ad personam Charakteristika.
In diesem Sinne halte ich auch die Überschrift dieses Kapitels „Zum Sachverhalt“ nicht für passend. In der Sache geht es mir um Akteneinsicht nach AIG in Quellcode, der hier eine verwaltungsrechtliche Außenwirkung entfaltet.

Zu B I
Auch dieser Absatz besteht zu einem großen Teil aus Darlegungen der eigenen Ansichten der Hochschule, wie schwierig sie die Situation empfand und welche Leistung sie angesichts (angeblicher) Alternativlosigkeit vollbracht hat. Auch das finde ich in der Sache wenig hilfreich. Ich möchte mich daher in der Folge ausschließlich auf die mir für die Akteneinsicht nach AIG relevant erscheinenden Aussagen beziehen.

Die Hochschule gibt nun erstmalig in 14 an, dass die Nutzung der Anwendung nie auf Dauer angelegt war. Das widerspricht meiner Lesart nach ihrem vorherigen Ablehnungsgrund, der erst zu dieser Klage führte, wonach mit dieser Anwendung Geld verdient werden sollte/musste, also ein langfristiger Horizont durch Teilnahme am Wirtschaftsverkehr begründet wurde.

Es ist, wie auch in Verfahren VG 8 K 556/21, eine (weitere) nachgewiesene fehlerhafte Aussage. Mein Vertrauen in die Belastbarkeit der Aussagen dieser Hochschule ist inzwischen erheblich angekratzt. Wertes Gericht, was bitte soll ich denn nun noch überhaupt glauben können? Morgen wird es wieder eine neue Behauptung der Hochschule geben, die ihren vorherigen widerspricht.

Ich beantrage daher, dass die an der Entwicklung beteiligten Programmierer und Verantwortlichen an Eides Statt versichern (§98 VwGO i.V.m §452 ZPO), dass keine anderen vorherigen Versionen des Quellcodes oder der Anwendung existieren. Weder in Backups noch auf anderen Systemen oder Datenträgern.

Im Verlauf des Absatzes werden dann viele „Gründe“ aufgelistet, warum man keine Versionierung eingesetzt hat. Jeder einzelne dieser „Gründe“ sind keine Begründungen, sachfehlerhaft und eine Vermischung verschiedenster Begriffe, die so wie durch die Hochschule dargestellt nicht zusammenhängen.
Tatsächlich legen die „Gründe“ nahe, dass die Hochschule versucht mit weiteren (Schein-)Begründungen Verwirrung zu stiften und so die Veröffentlichung der Vorversionen zu verhindern.

Eine Versionierung von Quellcodedateien, wie hier als weltweiter Stand der Technik für jede Anwendungsentwicklung zu bezeichnen, erlaubt es zu bestimmten Versionen von einzelnen Quellcodedateien zurückzukehren, bzw. sich die Unterschiede dazwischen anzuschauen. Ein wenig vergleichbar mit der Funktion „Änderungen verfolgen“ in Textprogrammen (z.B. Word oder LibreOffice), wobei das technisch nicht die einzige Möglichkeit ist, Versionen von Dateien anzufertigen.
So ließe sich auch jeden Abend eine Kopie einer Datei erzeugen und dann so benennen, dass der aktuelle Tag Teil des Dateinamens wird. Ebenso könnte man alle Dateien jeden Abend in eine ZIP Datei einpacken und diese entsprechend des aktuellen Tages benennen. Auch entstehen automatisch (sozusagen ungewollt) dadurch Versionen, wenn von einem System Backups angefertigt werden, da im Backup ja die alten Quellcodedateien erhalten bleiben, selbst wenn diese im laufenden Produktivsystem überschrieben werden. Üblich ist jedoch die Verwendung von Versionierungstools, wie z.B. GIT, die sowohl online als auch ausschließlich offline verwendet werden können. Die Nutzung eines solchen Tools ist nicht mal ansatzweise zusätzlicher Aufwand, wie die Hochschule darzustellen versucht. Das Erstellen eines Versionsstandes aller Quellcodedateien einer Anwendung entspricht dem Aufruf eines einzigen Kommandos (und die Hochschule hat ja mit ihrem Quellcode PDF gezeigt, dass sie sowas kann) und dauert in Summe nicht länger als 10s. Das Versionierungstool prüft dann vollautomatisch, welche der vielen hundert Quellcodedateien sich geändert haben und erzeugt daraus vollautomatisch Kopien und erlaubt auch einen Kommentar zu diesem Versionsstand zu hinterlegen. Es handelt sich bei der Versionierung also nicht um ein zeitaufwendiges „Extra“, was mit zusätzlichem Aufwand umgesetzt/geplant werden muss, sondern um die ganze normale Arbeitsweise eines jeden professionellen Entwicklers. Diese Versionshistorie wird dann lokal in einem extra Ordner abgelegt und, sofern man das will, online gespiegelt.

14
Die Versionierung einzelner Quellcodedateien bedingt keine Definition oder Nutzung von Major/Minor Releases und auch umgekehrt gibt es keine Bedingungsbeziehung. Das sogenannte Releasemanagement definiert, wann und unter welchen Umständen eine bestimmter Stand aller Quellcodedateien einer Anwendung so stimmig miteinander ist (also möglichst fehlerfrei zusammen funktioniert), dass dieser Stand als Update öffentlich zur Verfügung gestellt wird. Häufig wird dabei die Semantische Notation verwenden, wobei eine Versionsnummer dann aus einer Haupversion, Unterversion und eines Patchlevels besteht z.B. v1.2.3. Es ist liegt in der Verantwortung der Programmierenden (bzw. den selbsgewählten Definitionen im Releasemanagement), wann welche

dieser Ziffern hochgezählt wird. Üblicherweise wird bei Fehlerbehebungen nur die letzte Ziffer hochgezählt, bei kleineren Funktionsupdates wird die mittlere Ziffer hochgezählt und große Updates spiegel sich durch hochzählen der ersten Ziffer wieder.
Komplett unabhängig davon existieren jedoch die einzelnen Versionen jeder einzelnen Quellcodedatei dieser Anwendung. Tatsächlich hat die Hochschule ja Updates veröffentlicht, also ganz bestimmte Versionen einzelner Quellcodedateien als Update veröffentlicht. Sie beschreibt ja auch, dies nach entsprechenden Tests dieser Funktionen getan zu haben, also einen strukturierten Prozess zu durchlaufen. Sie hat lediglich keine Versionsnummer vergeben, was jedoch, entgegen der Darstellung der Hochschule auch ohne weiteren Aufwand möglich ist. Das Bereitstellen, oder vielleicht besser, visualisieren einer Versionsnummer kann wie die Versionsnummer in einem Textdokument in eine einzige Quellcodedatei eingetragen werden. Eine Versionierung der einzelnen Quellcodedateien ist dafür nicht erforderlich. Der Aufwand dafür beläuft sich auf unter einer Minute. Es letztlich nur Text.
Auch OpenSource bedingt keine Notwendigkeit einer Versionierung, wie hier dargestellt und auch umgekehrt gibt es keine Bedingungsbeziehung. OpenSource Software kann auch ganz ohne Versionierung zur Verfügung gestellt oder entwickelt werden. Es handelt sich nämlich lediglich um ein Lizenzierungsmodell.

15
Der Gesamtprozess einer Softwareentwicklung wird in sogenannten Softwareentwicklungsmodellen beschrieben. Ein solches Vorgehensmodell erfordert nicht die Nutzung einer Versionierung. Es kann also nicht als Begründung dafür dienen, warum man diese nicht eingesetzt hat. Es ist schlicht etwas ganz anderes.
Dass die Hochschule hier als Beispiel gerade ein Vorgehensmodell auswählt (ohne es zu benennen), bei der besonders hoher Aufwand für Dokumentation entsteht (was mit Versionierung nichts zu tun hat), passt ihr natürlich in die Argumentation. Gerade solche Vorgehensmodelle liefern jedoch Software „von der Stange“ mit nur wenigen Anpassungen; also keine Neuentwicklungen. Im Bereich der Behörden hat sich zum Beispiel das V Modell XT etabliert, welches gerade auf Dokumentation viel Wert legt. Bei Vorgehensmodellen gibt es keinen One Size Fits All Ansatz. Vielmehr ist für jeden Fall auszuwählen, welches am effizientesten zum Ziel führen könnte.
Das Vorgehensmodell hat jedoch mit der Nutzung von Versionierung einzelner Quellcodedateien nichts zu tun, auch wenn bestimmte Vorgehensmodelle sich besonders für die Nutzung einer Versionierung anbieten.

16
Tatsächlich behauptet die Hochschule „agil“ als Vorgehensmodell gewählt zu haben. Dabei handelt es sich um einen belegten Begriff eines Vorgehensmodells. Es ist aufgrund der bisherigen Schriftverkehrs nicht eindeutig, ob sich die Hochschule hier an eine etablierte Begriffsbestimmung hält oder etwas ganz anderes meint. Agile Softwareentwicklung folgt dabei, wie auch alle anderen Vorgehensmodelle, ganz bestimmten Regeln. Besonderen Wert legt die agile Entwicklung auf viele (kleine) Iterationen funktionstüchtiger Releases. Anstatt sich also zum Anfang einmal sehr lange mit einem Kunden hinzusetzen und alles durchzusprechen, wird beim agilen Ansatz nur ein absolut minimal notwendiges Gespräch geführt, was in der Folge aber bereits zu einem funktionstüchtigen und nutzbaren Produkt führt und eingesetzt werden kann. Die Abstände zwischen solchen Iterationen betragen üblicherweise 2 Wochen (auch als Sprint bezeichnet). Am Ende steht ein neues Release bereit, was der Kunde testen und sofort Feedback geben kann.
Es gibt also gerade bei agiler Entwicklung besonders viele Releases. Aber auch diese können komplett ohne Versionierung einzelner Quellcodedateien durchgeführt werden, genauso wie mit Versionierung.
Es ist jedoch schlicht nicht glaubhaft, dass es im Falle eines nicht erkannten Programmfehlers, der erst nach dem Update im Produktivsystem auftritt, nicht die Möglichkeit geben sollte, auf das vorherige Release zurückzukehren. Also zumindest immer eine vorherige Version eines Releases existieren sollte.

Weiterhin kann eine Versionierung, neben den Versionen auf dem Produktivsystem, die ja angeblich nicht existieren auch durch die Entwickler auf den eigenen Rechnersystemen vorgenommen werden.
Auf mein Argument, dass doch Versionsstände auch schon durch die regelmäßigen Backups des Produktivsystems oder auch des Entwicklungssystems vorliegen, geht die Hochschule nicht ein.
Um es ganz deutlich zu sagen, stellt sich der Prozess, den die Hochschule bei der Entwicklung dieser Anwendung benutzt aktuell so dar: Es gibt Entwickler, die am Quellcode entwickeln, auf einem einzigen Rechnersystem, ohne Backups oder Sicherung für eine Anwendung, die die ganze Hochschule verwendet, und von diesem einem Rechnersystem wird dann ab und zu ein Stand auf das Produktivsystem überspielt, für welches auch keine Backups existieren oder die Möglichkeit, im Falle eines Problems nach dem Update auf die vorherige Version zurückzukehren. Das ist nicht glaubhaft.

Zu B II
Vorab sei gesagt, dass die Hochschule keinerlei Bezug auf mein in Anlage 1 beschriebenes Verfahren aus meinem Schreiben vom 26.06.2021 zur Akteneinsicht nimmt. Zu keinem der von mir dort genannten Punkte. Ich sehe dieses Verfahren als eine von mir bestimmte Art des Informationszuganges nach §7 (3) AIG an, gegen den die Hochschule keine wichtigen Gründe dargelegt hat.
Ich beantrage daher die Hochschule zu verpflichten, Auskunft nach dem von mir beschriebenen Verfahren in Anlage 1 meines Schreibens vom 26.06.2021 zu erteilen (§7 (3) AIG).
Die ursprünglich von mir bestimmte Art des Informationszuganges durch Zugang zum online existierenden Versionierungssystem der Hochschule, scheint ja an der (angeblich) fehlenden Nutzung zu scheitern.

18
Die Ordnerstruktur liegt direkt auf dem Datenträger vor, was unter 19 ja auch bestätigt wird. Das von mir vorgeschlagene Verfahren hätte diese erhalten. Die Ordnerstruktur ist daher, weil zur Ausführung des Programms erforderlich und mit dienstlichen Zwecken angelegt, Teil einer Akte nach AIG, weil ohne diese die Anwendung nicht funktionieren würde. Ob diese existierende Ordnerstruktur nochmal extra dokumentiert ist oder nicht, ist dafür unerheblich.
Dieses Problem tritt nur auf, weil die Hochschule diese Art der Akteneinsicht gewählt hat.

19
Ich habe nicht verlangt Kapitel oder Überschriften zu erstellen. Das ist ein „Lösungsvorschlag“ der Hochschule, für das von ihr selbst geschaffene Problem. Ich will die Akte (alle Ordner und Dateien) genau so, wie sie auf der Festplatte liegen.
Dieses Problem tritt nur auf, weil die Hochschule diese Art der Akteneinsicht gewählt hat.

20
Diese Aussage zeugt von grundsätzlicher Unkenntnis, wie Quellcode bearbeitet und gelesen wird. Anders kann ich es leider nicht ausdrücken und wirklich jeder Programmierende weltweit wird Ihnen bestätigen können, wie weltfremd dieses Argument ist. Quellcode existiert in (hierarchischen) Strukturen, die einer Modellierung entspringen. Genau das ist eines der kreativen Elemente an Softwareentwicklung. Diese Strukturen zu verflachen (hier durch Löschung) und dann alphabetisch zu sortieren ist eine Veränderung der Originalakte. Das dann „ordnen der unstrukturierten Aktenteile“ zu nennen ist, wie das Gemälde der Mona Lisa in gleich große Stücke zu schneiden und dann der durchschnittlichen Helligkeit jedes Stückes nach sortiert zu beauskunften. Das Kunstwerk und auch die Originalakte werden dadurch zerstört.
Dieses Problem tritt nur auf, weil die Hochschule diese Art der Akteneinsicht gewählt hat.

21
Hier möchte ich die Erklärung der Speicherung von Quellcode vorausschicken.

In einer Quellcodedatei existiert ausschließlich die Speicherung eines Zeichens. Also zum Beispiel der Buchstabe A. Quellcode beinhaltet jedoch keinerlei Hinweise auf die Darstellung dieses A, also die Schriftart, oder ob dieser kursiv dargestellt werden soll. Die Darstellung wird erst am Bildschirm durch das Bearbeitungsprogramm durchgeführt, welches den Quellcode anzeigt. Immer wieder aufs neue, wenn die Quellcodedatei geöffnet wird. Dort lässt sich üblicherweise einstellen, in welcher Schriftart der Programmierende den Quellcode sehen möchte. Tatsächlich haben Programmierende unterschiedliche Präferenzen, in welcher Schriftart (aber auch Farbe und Größe) sie sich Quellcode am liebsten anschauen. Die identische Quellcodedatei wird daher auf einem Rechner von unterschiedlichen Programmierenden unterschiedlich angezeigt, ihr Inhalt auf der Festplatte ist jedoch immer gleich.
Dies unterscheidet Quellcode fundamental von Textdokumenten, wie z.B. für LibreOffice oder Word. In solchen Textdokumenten ist für jeden einzelnen Buchstaben ebenso hinterlegt, in welcher Schriftart und Form (z.B. fett) dieser angezeigt werden soll. Dadurch sieht ein solches Textdokument dann auf jedem Rechnersystem gleich aus, was ja auch bezweckt wird.

Einrückungen sind ein wichtiges Strukturelement von Programmiersprachen. Es gibt sogar Sprachen, bei denen Einrückungen auch Bedeutung für den Ablauf des Programms haben und nicht nur für die Lesbarkeit.
Aufgrund dieser Wichtigkeit ist es ein weltweiter Standard, dass zur Darstellung von Quellcode eine sogenannte `MonoSpace` Schriftart verwendet wird. Es ist eines der ersten Dinge die Studierende auch an dieser Hochschule lernen, dass Quellcode nur in einer solchen Schriftart dargestellt werden soll. Bei einer solchen Schriftart wird ein recht breites M in der gleichen Breite dargestellt wie in dünnes I.
Beispiel ohne MonoSpace Schriftart:

MMMM
IIII

Beispiel mit MonoSpace Schriftart:

```
MMMM
IIII
```

Eine solche MonoSpace Schriftart sorgt also dafür, dass die Strukturierung der Einrückungen nicht von der Wahl der Buchstaben selbst, sondern nur von deren Anzahl abhängig ist.

Hier hat die Hochschule zur Darstellung im PDF keine MonoSpace Schriftart verwendet. Damit sind Einrückungen nicht so dargestellt, wie sie ein Programmierender sehen würde, der natürlich immer eine MonoSpace Schriftart zur Anzeige verwenden würde. Es handelt sich also nicht um die Originalakte, da ja eine Interpretation der vorliegenden Zeichen durch die Wahl einer Schriftart stattgefunden hat, die es in der Originalakte jedoch nicht gibt.
Da die Originalakte keinen Verweis auf eine bestimmte Schriftart enthält, ist das abdrucken bzw. übertragen einer Quellcodedatei mithilfe einer Schriftart in ein anderes Dokument, regelmäßig eine Veränderung der Originalakte und somit ein ganz neues Dokument und keine Kopie. Das AIG sieht explizit die Einsicht durch elektronische Übermittlung der Originaldokumente vor.

Die Hochschule hat weiterhin kein PDF erstellt, in welchem Text abgespeichert wurde; also z.B. Buchstabe A mit der Definition in welcher Schriftart dieser dargestellt werden soll.
Tatsächlich befindet sich auf jeder Seite ein Bild, welches Text darstellt. Vergleichbar mit einem Bild von einem KFZ. Auf einem solchen Bild ist natürlich auch Text abgebildet, nämlich mindestens das Nummernschild. Letztlich ist es aber ein Bild und kein Textdokument.
Einrückungen lassen sich in Quellcode mit Leerzeichen oder mit sogenannten Tabstops (vergleichbar wie bei einem Textdokument) erreichen. Ein Tabstop ist jedoch EIN Zeichen in der Quellcodedatei. Erst das Darstellungsprogramm entscheidet (das lässt sich auch konfigurieren), für eine wie starke visuelle Einrückung dieses eine Zeichen sorgen soll. Bei einem als Bild dargestellten Text lässt sich jedoch nicht ermitteln, ob es sich um ein bestimmte Anzahl Leerzeichen oder nur um einen Tabstop handelt. Ob Leerzeichen oder Tabstop, der Bereich wird schlicht weiß dargestellt.

Aufgrund der beschränkten Seitenbreite finden auch Zeilenumbrüche statt, die im Quellcode (vermutlich) nicht vorhanden sind (erstmalig auf Seite 4 des Quellcode PDF, und dann immer häufiger). Auch dies stellt eine Veränderung der Originalakte dar und ist gerade nicht üblich, wie von der Hochschule behauptet.
Diese Probleme treten nur auf, weil die Hochschule diese Art der Akteneinsicht gewählt hat.

22
Wie zuvor erläutert, leidet die Lesbarkeit darunter, dass die Hochschule eben keinen Text in das PDF eingefügt hat, sondern Bilder, die Text darstellen. Und Bilder benötigen mehr Speicherplatz, als Text. Ein durch die Hochschule selbstverschuldetes Problem ihrer Wahl dieser Akteneinsicht.
Im Übrigen hätte auch der Übersendung einer DVD mit besserer Auflösung nichts im Wege gestanden.
Die „Begründung“ alles in eine einzige PDF Datei zu exportieren erschließt sich mir nur, wenn die Hochschule die technische Fähigkeit zur Erstellungen einer gepackten Datei (z.B. ZIP Format) nicht besitzt. Das kann ich aber ausschließen.
Diese Probleme treten nur auf, weil die Hochschule diese Art der Akteneinsicht gewählt hat.

23
Ich beantrage die Herausgabe aller ausgeführten Kommandozeilenbefehle (bzw. auch Scripte) und die genaue Beschreibung eventueller Folgeschritte, die zu der von der Hochschule bereitgestellten PDF geführt haben.
Ich zweifle fundamental am Wahrheitsgehalt der Behauptung, die darzustellen versucht, es würde sich um einen einfachen Weg handeln. Nur so lässt sich das Ermessen prüfen, warum die Hochschule keine andere Form der Auskunft gewählt hat. Ebenso gehe ich von Fehlern in dem Prozess aus (siehe Bemerkung zu 24). Sobald mir die Kommandozeilenbefehle vorliegen, kann ich zweifelsfrei nachweisen, ob diese vollständig sind und ob diese auch die Quellcode PDF erzeugt haben können.
Im Bereich der Kommandozeilenbefehle ist es üblich für einen Vorgang nacheinander mehrere dieser Befehle zu nutzen, wobei das Ergebnis der Bearbeitung eines Befehls dann die Eingabe eines folgenden Befehls darstellt. Dieses Konzept ergibt sich aus der grundlegenden Idee von (Linux) Kommandozeilenbefehlen, dass diese möglichst nur eine einzige Aufgabe erledigen, diese eine Aufgabe aber besonders gut und fehlerfrei. Somit kann durch die Kombination mehrerer Befehle dann recht flexibel vieles erreicht werden. Es ist ebenso nicht unüblich, dass dann dort auch mal 10 Befehle nacheinander ausgeführt werden, damit am Ende das herauskommt, was man haben wollte. Damit man diese Menge an Befehlen nicht immer wieder von Hand eingeben muss, lassen sich diese Befehle in einer Datei sammeln und dann „in einem Rutsch“ ausführen. Eine solche Datei, auch Script (oder Batchdatei) genannt, ist natürlich auch ein Programm und damit selbst geschriebene Software. Ich sehe dieses als von meinem Antrag zur Herausgabe erfasst an.
Und um es gleich vorweg zu nehmen, in Linuxsystemen wird eine Historie über die ausgeführten Kommandozeilenbefehle geführt. Sie sind also als Information ermittelbar und die Hochschule kann sich nicht darauf berufen eine solche Information würde nicht vorliegen.

24
Dem PDF fehlen sämtliche Bilder der Anwendung. Wobei ich hier natürlich auch keinen Abdruck möchte, sondern die Originalakte. Bei Bilddateien gelten in etwa die gleichen Erklärungen, wie ich zuvor für Quellcodedateien ausführte. Die Abbildung einer Bilddatei ist bereits eine Veränderung der Originalakte und keine exakte Kopie. Nur die digitale Kopie der Bilddatei selbst ist mit der Originalakte identisch.

Dem PDF fehlen sämtliche Ordnerstrukturen, von denen jedoch auch die Hochschule angibt, dass diese existieren.

Dem PDF fehlen sämtliche Konfigurationsdateien. Dies sind Dateien, die über die Versionen der verwendeten Bibliotheken Auskunft geben und welche Bibliotheken überhaupt verwendet wurden. Nur mit den passenden Versionen lässt sich die Funktionsweise des Programms ermitteln (und die passende Version überhaupt erst herunterladen), da zwischen verschiedenen Versionen auch Verhaltensunterschiede auftreten. Dies ist auch deswegen für mich spannend, weil die erste Schwachstelle der Hochschule nur auf eine solche veraltete Bibliothek zurückzuführen war und ich natürlich wissen möchte, wie das kritische Problem behoben wurde, bzw. ob überhaupt eine Aktualisierung vorgenommen wurde.

Dem PDF fehlen auch noch weitere Ressourcen, die lediglich aus dem Netzwerk der Hochschule erreichbar sind. Hier mindestens eine Liste von nicht erlaubten Emailadressen, die bei der Registrierung nicht verwendet werden dürfen. Das eine solche Liste existiert, konnte ich bereits aus dem mir übermittelten PDF ermitteln.

Dem PDF fehlen alle Metainformationen der Quellcodedateien. Das sind hier beispielhaft die Zeitstempel der Quellcodedateien oder Zugriffsberechtigungen.

Nach meinem derzeitigen Kenntnisstand sind einige abgedruckte Quellcodedateien in dem PDF „zu leer“. Diese sollten mehr Code enthalten. Ich kann nicht wissen, ob diese Daten gelöscht wurden, oder ein Fehler bei der Übertragung aufgrund fehlerhafter Nutzung/Verwendung der Kommandozeilenbefehle stattfand. Bisher hat die Hochschule auch nicht bekannt gegeben, ob sie überhaupt Daten gelöscht/geschwärzt hat, sondern lediglich, dass sie der Meinung ist, vollständig Auskunft erteilt zu haben. Das gibt mir natürlich nur wenig effektive Rechtsmittel an die Hand, gegen ungerechtfertigte Schwärzungen vorzugehen, wenn diese nicht ersichtlich sind, sondern aus der Originalakte entfernt wurden.
Weiterhin verweise ich auf Anlage 1 meines Schreibens vom 26.06.2021, was für mich zu Quellcode einer Anwendung gehört (und auch für die bpb).

Ebenso konnte ich jene Programmteile nicht finden, die für den Zugriff auf die Nutzerdaten bei einer Anfrage durch das Gesundheitsamt genutzt werden.

Im Übrigen möchte ich auch darauf hinweisen, dass vermutlich alle von der Hochschule verwendeten OpenSource Bibliotheken eine Lizenz aufweisen, die diese verpflichtet darauf hinzuweisen, welche Bibliotheken sie nutzt und diese auch bei der Übermittlung ihres darauf aufbauenden Programms verpflichtend mit zu übermitteln und zwar in genau der Version, die sie verwendet. In diesem Sinne könnte § 6 (4) AIG anwendbar sein (sofern denn klar wäre, welche Bibliotheken in welcher Version genutzt wurden), jedoch ist die Hochschule aufgrund von Lizenzbedingungen sowieso zur Weitergabe verpflichtet. Ich schätze, dass der Quellcode, den die Hochschule wirklich selbst entwickelt hat, unter Einbeziehung der von der Hochschule genutzten bereits vorhandenen OpenSource Software, auf die sie sich stützt, unter 10% Anteil hat.
Diese Probleme treten nur auf, weil die Hochschule diese Art der Akteneinsicht gewählt hat.

25
Für eine Einsicht nach AIG, sehe ich es nicht als erheblich an, ob die Anwendung als webbasiertes Programm erstellt wurde, oder ob mehrere Personen diese erstellt haben, oder ob diese lange verwendet wird, oder ob es verschieden Versionen davon gibt.
Dafür sehe ich keine Rechtsgrundlage.
Aber auch fachlich ist die Begründung in sich nicht stimmig. So wird insinuiert, dass es einen Zusammenhang zwischen einen webbasierten Programm gibt und der Programmierung in Co-Arbeit. Beide Sachen haben absolut nichts miteinander zu tun. Man kann ein webbasiertes Programm allein

erstellen oder mit anderen zusammen. Ebenso kann man jede andere Art (nicht webbasiert) von Programm sowohl allein, als auch mit anderen zusammen erstellen.
Pikanterweise ist die Anwendung aber zumindest auch eine webbasierte Anwendung, jedoch nicht ausschließlich. Die Hochschule in ihrem Widerspruchsbescheid selbst die Formulierung „Web-App“.

26
Sofern die Hochschule angibt vollständig Auskunft erteilt zu haben, stellt sich die Frage, warum so viele Dokumente fehlen, obwohl ihr mein Verfahrensvorschlag und meine Ansichten, was zu Quellcode alles dazugehört ja bekannt sind. Ich kann nicht erkennen, dass sie sich sachlich mit genau diesen Argumenten auseinandergesetzt hat.

Fazit
Erneut habe ich versucht wirklich jedes Argument der Hochschule zu widerlegen. Jedes einzelne war fehlerhaft und kontextverfälschend.
Schaut man sich die Formulierungen der Hochschule aus ihrem Widerspruchsbescheid vom 06.04.2021 an, so fällt auf, dass diese letztlich verhindern möchte, dass andere den Quellcode in einer Art erhalten, dass dieser für diese von Nutzen ist (Hervorhebung von mir):

„Von einer Wettbewerbsrelevanz muss ausgegangen werden, wenn die Programmdetails des ausschließlich von dem icampus-Team der TH Wildau entwickelten und angewandten Web-App dergestalt bekannt werden, ***dass sie auch von anderen nachvollzogen werden und damit letztlich kopiert werden können****.“*

„...da allein dadurch, dass die Anwendungs- und Programmiermerkmale den geschützten Bereich der Urheber und Entwickler bei der TH Wildau verlassen, ***die Gefahr einer anderweitigen Verwendung*** *verursachen kann.“*

„...um ein für eigene Tätigkeiten entwickeltes Computerverfahren vor ***ungerechtfertigter Benutzung durch andere*** *zu schützen.“*

Dem letzten Schreiben der Hochschule nach, scheint diese aktuell alle ihre bisherigen Ablehnungsgründe nach AIG nicht mehr aufrechtzuerhalten. Das wird aber meiner Erfahrung nach an ihrer Grundeinstellung nichts ändern. Wir Menschen ändern einmal liebgewonnene Ansichten nicht einfach so. Und auch wenn ich immer „die Hochschule“ schreibe, sind es letztlich Einzelpersonen, die hier handeln. Warum die Hochschule also dann nicht eine viel einfacher zu erstellende und zu versende ZIP Datei, wie im von mir vorgeschlagenen Verfahren nutzt, führe ich darauf zurück, hier in Überschreitung ihres Ermessens die vollständige Akteneinsicht vorsätzlich zu erschweren, bzw. durch Vorenthaltung und Veränderung von Daten unmöglich zu machen. Dass die Hochschule auch nicht auf meinen Verfahrensvorschlag eingeht, stützt diese These, da es mir sachlich in der Tat keine rechtlichen Gegenargumente zu geben scheint.

Wenn ich mich unter dieser Annahme in die Hochschule hineinversetze, dann sorge ich nämlich dafür, dass Quellcodedateien fehlen, die Ordnerstrukturen wegfallen, die Quellcodedateien nicht als Text zum Copy&Paste vorliegen, sondern als Bild und das Löschungen von Programmteilen nicht ersichtlich sind. Ebenso wähle ich die Auflösung und Schriftart so, dass es OCR Programme, die die Möglichkeit haben aus Text, welcher in Bildern steckt, wieder normalen (editierbaren) Text zu machen, besonders schwer haben. Und natürlich packe ich dann alles in eine einzige PDF Datei, damit der andere es besonders schwer hat, daraus wieder die einzelnen Quellcodedateien herauszulösen. Und für all das, werde ich entsprechende Personalressourcen einsetzen und nehme Mehrarbeit billigend in Kauf.

Die Hochschule hat also wirklich (fast) alles unternommen, um die Akteneinsicht nach AIG zu erschweren. Inklusive der Notwendigkeit einer Klageerhebung. Dafür nutzt sie die ihr unzweifelhaft (zumindest in Teilen) vorliegende Fachkenntnis. Ebenso nutzt sie diese, um mit technischen Begriffen in einer Art zu argumentieren, die inhaltlich keinen Sinn ergeben, um damit abzulenken und zu verwirren. (Ich möchte die Alternative, dass sie wirklich glaubt, was sie hier schreibt nicht wahr haben). Ich bin überzeugt davon, dass Ihnen jeder externe Sachverständige dies bestätigen kann, sofern es meine Argumentation nicht vermocht hat.

Eine Akteneinsicht nach AIG in Quellcode, kann doch in der Konsequenz nur dann vollständig sein, wenn die herausgegebenen Dokumente es dem Petenten (entsprechendes Fachwissen vorausgesetzt) erlauben, die darauf basierende Anwendung auch zur Ausführung zu bringen und so dessen Verhalten und Funktionsweise zweifelsfrei feststellen und nachweisen zu können. Nur so kann auch die Vollständigkeit überhaupt nachgewiesen werden.

Wertes Gericht, ich bemühe mich meine Ausführungen dergestalt darzulegen, dass auch technisch weniger tief in der Materie bewanderte Personen diese verstehen können. Sicherlich gelingt mir das mal besser und mal nicht, was vor allem daran liegt, dass mir Ihr Wissensstand nicht bekannt ist. Es ist mir jedoch wichtig, die (Schein-)Argumente der Hochschule zu widerlegen. Ich bitte um Nachsicht.