Inhaltsverzeichnis

[Vorwort 2](#_Toc16153649)

[Aufgabe 1 – Architekturskizzen 3](#_Toc16153650)

[A-Architektur 3](#_Toc16153651)

[T-Architektur 5](#_Toc16153652)

[TI Architektur 8](#_Toc16153653)

# Vorwort

Bei diesem Dokument handelt es sich um eine Arbeitsmappe, welche sich mit der Aufgabenstellung der Firma Capgemini befasst. Für die Bearbeitung der Aufgaben im Architekturdesign, wird ein UML-Tool benötigt. Hierfür wurde das UML-Tool Modelio verwendet, bei dem es sich um eine Open-Source Sofware handelt, welche die Standards UML2 und BPMN2 unterstützt. Zweites ist für die Bearbeitung der Aufgabenstellung jedoch nicht weiter relevant.

# Aufgabe 1 – Architekturskizzen

Die Aufgabe 1 besteht daraus, ein Architekturmodell nach der Vorgehensweise zu erstellen, wie sie uns in der Präsentation der Firma Capgemini vorgestellt worden ist. Dieses Architekturmodell beinhaltet eine A-Architektur, T-Architektur und eine TI-Architektur. Dieses Architekturmodell sollten wir für ein „Bahn 2.0“ System erstellen. Welches neben einer automatischen Sitzplatzzuweisung, auch eine Infotainmentsystem an den einzelnen Sitzplätzen zur Verfügung stellt und noch einige andere autonome Funktionen übernehmen soll.

### A-Architektur

Die A-Architektur ist der Grundbaustein für ein Architekturmodell, da sich diese mit den Funktionen und Abläufen des zukünftigen Systems befassen. Dieser Abschnitt ist somit auch Technik frei und sehr Businessorientiert.

Abbildung 1 Komponenten des Systems für die Bahn 2.0

Beim ersten Schritt der A-Architektur, welche sich mit dem analysieren der benötigten Komponenten beschäftigt, sind wir zu den folgenden sechs Subsystemen gekommen (Abbildung 1 Komponenten des Systems für die Bahn 2.0).

Das Bahn 2.0 System benötigt ein *Abrechnungssystem*, welches sich um die gesamte Verwaltung und Abwicklung von Käufen der Tickets und der Infotainmentinhalte beschäftigt. Des Weiteren wird natürlich das *Infotainmentsystem* selbst benötigt, das sich ausschließlich um die Anforderungen in dem Medien- und Entertainmentbereich kümmert, die dem Zuggast, an seinem Platz, zur Verfügung steht. Damit der Zuggast überhaupt einen Sitzplatz zugeordnet bekommt, wird natürlich auch eine *Sitzplatzverwaltung* benötigt, welche aus der Anforderung der automatischen Sitzplatzzuweisung hervorgeht. Damit der Zuggast auch genau weiß, welcher Platz ihm zugeteilt ist und wo sich dieser befindet, wird ein Subsystem, welches speziell für das Informieren des Zuggastest dient, gebraucht. Dieses Subsystem wird in den *Infoterminals* laufen. Damit die anderen Systeme überhaupt wissen, auf welchen Zuggast sie ihre GUI anpassen müssen, braucht man natürlich noch eine *Profilverwaltung* in der alle wichtigen Daten über den Zuggast abgefragt und aktualisiert werden können. Zu guter Letzt wäre da noch das *RFID-System*, das dazu dient, zu erkennen ob Beispielweise ein Zuggast gerade den Zug betritt oder sich an seinen zugewiesenen Sitzplatz setzt.

Im zweiten Schritt, der Erstellung der A-Architektur, wird geprüft, mit welchen umliegenden Systemen das zukünftige System interagiert (welches der Einfachheit, im weiteren Dokument immer als „Bahn 2.0 System“ bezeichnet wird). Hier wird sowohl geschaut welche Systeme wir benötigen damit das Bahn 2.0 System lauffähig ist, also Abhängigkeiten unserer Seite aus und welche Fremdsysteme auf unser System zugreifen. Außerdem wird in diesem Abschnitt festgelegt, welche User das System haben wird.

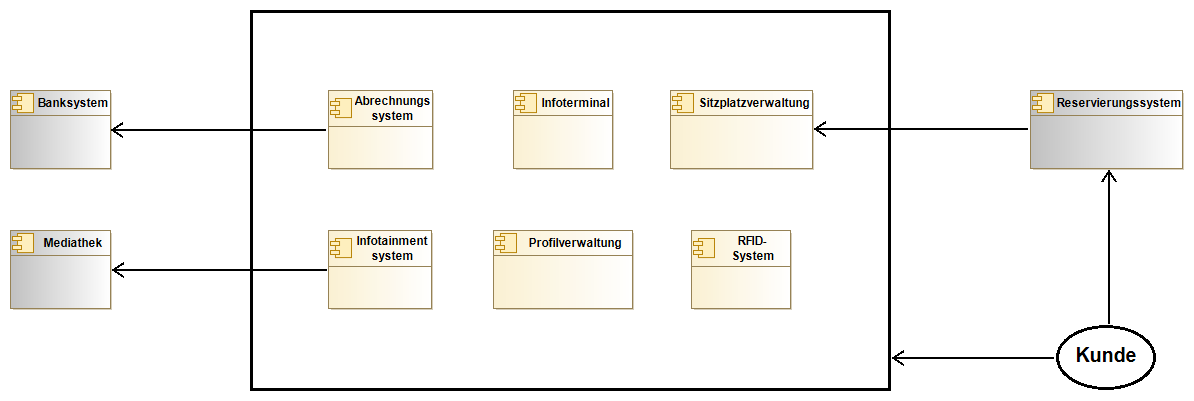


Abbildung 2 A-Architektur mit Fremdsystemen und Usern

Wie man aus Abbildung 2 entnehmen kann, wird das Bahn 2.0 System nur zwei Abhängigkeiten mit anderen Systemen haben. Dort wäre zum einen das *Banksystem*, welches als Abhängigkeit für das Abrechnungssystem dient. Diese Abhängigkeit entsteht dadurch, dass das System Überweisungstransaktionen anstoßen muss. Dies kann das System nur tun, wenn es eine Anbindung zu einer Schnittstelle einer Bank hat. Hier könnte beispielweise eine Verknüpfung mit einer Schnittstelle der Hausbank der Bahn sein, welche durch eine Einzugsermächtigung des Zuggastes, von einer externen Bank, auf das Konto der Gesellschaft, Geld buchen möchte. Des Weiteren ist eine Abhängigkeit von dem Infotainmentsystem zu einem System, welches wir zum besseren Verständnis als *Mediathek* bezeichnen. Dieses Fremdsystem bietet die benötigten Filme und Serien zur Verfügung, welche dem Zuggast angeboten werden können. Beispielweise könnte hier ein Fremdsystem wie Netflix oder Maxdome hinter verborgen sein, sollte die Bahn mit einem Drittanbieter einen Vertrag abgeschlossen haben. Jedoch ist es auch möglich, dass die Bahn selbst eine Mediathek auf einem Server zur Verfügung stellt. Grundsätzlich sollten die großen Mengen an Daten, nicht direkt im Infotainmentsystem hinterlegt sein.

Da trotz der vielen Vorteile des einzuführenden Bahn 2.0 System, es dem Zuggast trotzdem möglich sein soll, ein Ticket zu kaufen, welches dem Zuggast ein Sitzplatz im Zug reserviert (nicht eine freie Wahl eines Sitzplatzes), wird natürlich eine Schnittstelle nach außen benötigt. Dies wird mit der Verbindung von *Reservierungssystem* zu Sitzplatzverwaltung dargestellt. Hinter diesem Fremdsystem würde sich das System zum Reservieren an einem Bahnschalter und Internet verbergen.

Den Anforderungen entnehmen wir nur einen User des ganzen Systems. Dies wäre der Zuggast, welcher in der Abbildung 2 als *Kunde* bezeichnet wird. Dieser User interagiert sowohl mit dem Bahn 2.0 System als auch mit dem Fremdsystem, welches auf unser System zugreift.

Grundsätzlich würden natürlich noch vereinzelt andere User mit dem System interagieren, wie Beispielsweise ein Administrator, der dafür sorgt, dass die Software einwandfrei installiert und gewartet wird oder ein Techniker, der Beispielweise Hardware austauschen muss, die für das System benötigt wird ( siehe später Abschnitt T-Architektur). Jedoch sind dies vereinzelte Interaktionen, da sich die A-Architektur jedoch nur mit Power-Usern befasst, wurden diese User nicht mit in die A-Architektur aufgenommen.

### T-Architektur

Im zweiten Teil der Erstellung einer Architekturskizze wird die sogenannte T-Architektur erstellt. Diese befasst sich mit den Abhängigkeiten innerhalb des Systems und welche Geräte später mit dem System interagieren.

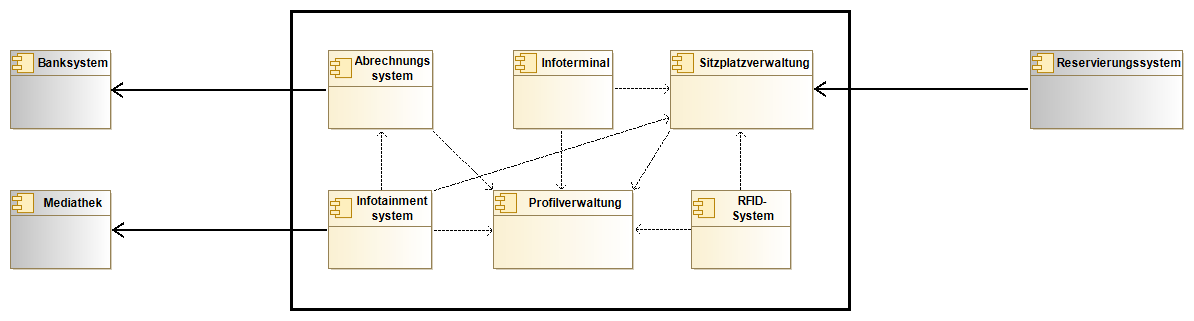


Abbildung 3 Abhängigkeiten innerhalb des Bahn 2.0 Systems

Als ersten Schritt, haben wir die Abhängigkeiten innerhalb des Systems dargestellt. Diese sind in Abbildung 3 zu sehen. Im nachstehenden Abschnitt werden die einzelnen Abhängigkeiten erläutert.

Betrachten wir zunächst das Infoterminal. Dies hat eine Abhängigkeit an die beiden Komponenten Profilverwaltung und Sitzplatzverwaltung. Diese resultieren aus der Anforderung, dass beim Einsteigen in den Zug, das Ticket gescannt wird und dem Zuggast sein zugeteilter Sitz angezeigt wird. Da außerdem der Sitzplatz auch angezeigt wird, muss ein Sitzplan geladen werden. Dieser wird aus der Sitzplatzverwaltung geladen. Des Weiteren hat der Zuggast die Möglichkeit, sich seinen aktuellen Kontostand und historische Daten (z.B. vergangene Fahrten) auf dem Infoterminal anzeigen zu lassen. Dafür müssen diese Daten aus der Profilverwaltung gezogen werden.

Als Nächstes betrachten wir die Abhängigkeit zwischen dem Abrechnungssystem und der Profilverwaltung. Diese entsteht durch das Abwickeln einer Kauftransaktion. Wird so eine Transaktion vom Infotainmentsystem angestoßen, werden zum weiteren Bearbeiten die Kontodaten bzw. das Guthaben vom Nutzer benötigt. Sonst weiß das Banksystem nicht, von welchem Konto das Geld eingezogen werden muss. Alternativ muss natürlich das Guthaben des jeweiligen Zuggastes im Profil angepasst werden.

Auch das Infotainmentsystem benötigt die Profilverwaltung, da es sonst nicht möglich wäre, dem Zuggast eine, für ihn zugeschnittene, Empfehlung anzuzeigen. Sollte der Zuggast dann etwas Kostenpflichtiges auswählen, muss das Infotainmentsystem natürlich auch die Möglichkeit haben, diese Aktivität dem Abrechnungssystem mitzuteilen. Dadurch entsteht die Dependance zu dieser Komponente.

Die Sitzplatzverwaltung ist für das organisieren der Sitzplätze innerhalb des Zuges verantwortlich. Um diese Funktion zu erfüllen, braucht sie, neben einer Abhängigkeit zu einer Datenbank (dies wird in einem späteren Abschnitt erläutert), die Profilverwaltung. Diese wird benötigt, da hinterlegt werden muss, welcher Zuggast, sich auf welchem Platz befindet. Hierfür braucht die Sitzplatzverwaltung wenigstens eine ID, am besten auch den Namen. Auch bei einer Reservierung braucht die Sitzplatzverwaltung diese relevanten Daten.

Das RFID-System benötigt sowohl die Sitzplatzverwaltung als auch die Profilverwaltung. Dies geht daraus hervor, weil das RFID-System erkennt, ob ein Zuggast ein- oder aussteigt. Um der Sitzplatzverwaltung aber überhaupt mitteilen zu können, welcher Zuggast gerade ein- beziehungsweise ausgestiegen ist, muss das RFID-System erst einmal erkennen, um welche Person es sich handelt. Dies erfolgt dann über die RFID. Sonst wüsste die Sitzplatzverwaltung nicht, wann ein Sitz frei bzw. belegt ist.

Zu guter Letzt bleibt noch die Profilverwaltung übrig. Diese hat zu keiner anderen Komponente eine Abhängigkeit. Somit würde man bei einem Deployen des Systems beispielsweise folgende Reihenfolge haben:

1. Profilverwaltung
2. Sitzplatzverwaltung
3. Abrechnungssystem
4. RFID-System
5. Infoterminal
6. Infotainmentsystem

Der nächste Abschnitt beschäftigt sich mit der Einführung einer Datenbank und UI (User Interface) in das System. Da Datenbanken in der heutigen Zeit eine gute und solide Möglichkeit sind, Daten zu speichern, haben wir uns für ein gängiges RDBMS (relationales Datenbankmanagementsystem) entschieden, wie in Abbildung 4 deutlich zu sehen ist. Eine Speicherung der Daten in einer Datenbank macht auch dahingehend einen Sinn, weil Millionen von Daten erzeugt werden, da für jede Person ein Profil angelegt wird. Dies allein erzeugt nicht die vielen Daten, doch das Stichwort hier ist „Historisierung“. Es soll dem Zuggast auch die Möglichkeit geben, seine früheren Fahrten an zu schauen. Dann werden noch Daten benötigt, welche für die Sitzplatzzuordnung benötigt werden und noch einige Daten mehr.

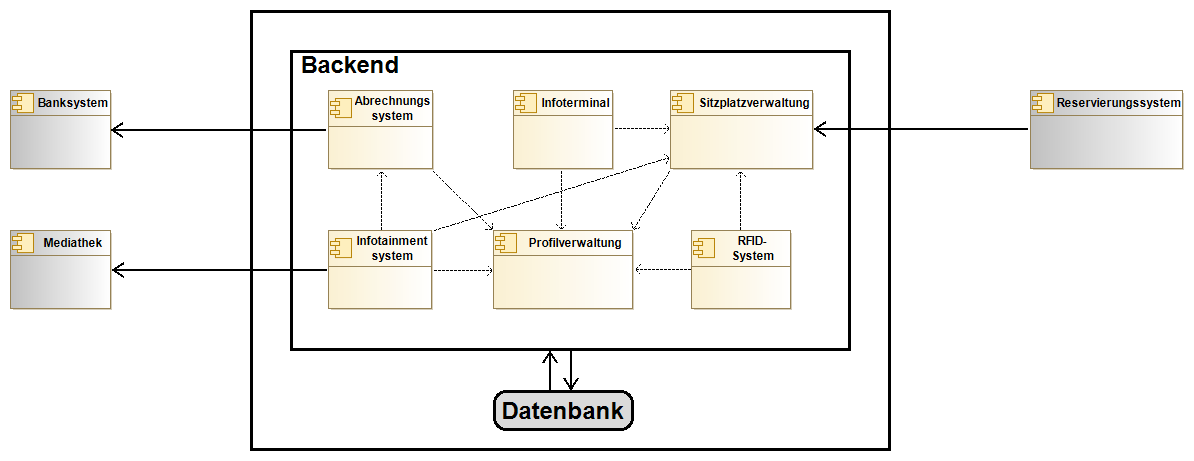


Abbildung 4 Einführung einer zentralen Datenbank für das System

Eine Einführung von Embedded Datenbanken, wie beispielsweise HSQLDB, wäre an dieser Stelle auch zu empfehlen, da es einige Daten gibt, die relativ häufig geladen werden und somit immer erreichbar sein müssen. Ein Beispiel hierfür ist der Sitzplan des jeweiligen Zugmodels. Dieser sollte in der Sitzplatzverwaltung selbst vorhanden sein, damit diese nicht immer in der zentralen Datenbank nachschauen muss, wie viele Plätze und Waggons der Zug hat.

Nach genauerer Analyse stellten wir fest, dass es für das ganze Bahn 2.0 System, keinen Sinn machen würde, ein UI zu erstellen. Natürlich hätten Komponenten, wie das Infoterminal und Infotainment-system, eine UI, da der Zuggast direkt mit diesen Komponenten interagiert, jedoch wären diese in den Komponenten selbst vorhanden. Da unser System als Ganzes jedoch nicht genutzt wird, sondern zu verschiedenen Zeitpunkten nur vereinzelte Komponenten, sind wir zu diesem Entschluss gekommen.

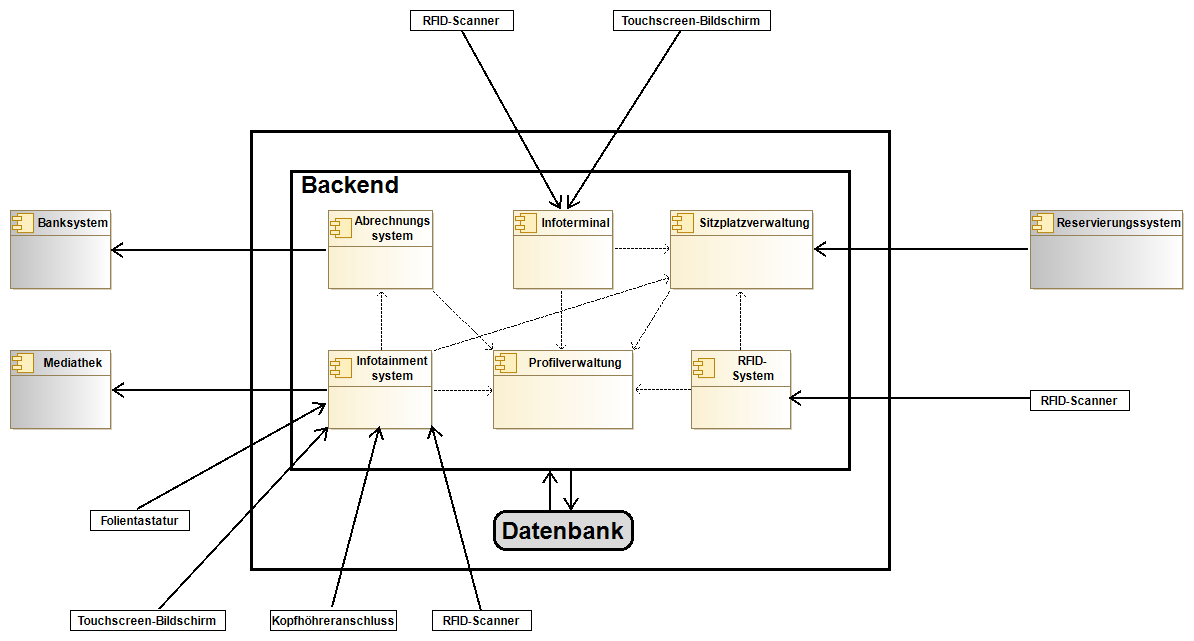


Abbildung 5 Benötigte Hardware für das Bahn 2.0 System

Der letzte Abschnitt der T-Architektur befasst sich mit der Festlegung der Hardware, welche dem System als Input dienen. In Abbildung 5 ist zu erkennen, dass mehrere Hardware mit dem System interagieren.

Zum einen wären da die RFID-Scanner. Diese sind die wichtigsten Komponenten im Zug, da durch den Scanner erkannt werden kann, wo sich ein Zuggast gerade befindet. Also ob er gerade an der Schleuse, einem Infoterminal oder an seinem Platz ist. Ohne diese Scanner, würde das System kaum bis gar nicht funktionieren. Wie in der Abbildung 5 zu sehen ist, haben wir drei Scanner bzw. handelt es sich hierbei eher um „Scanner-Typen“. Das bedeutet, dass es die Scanner für das RFID-System gibt, welche prüfen, ob ein Zuggast gerade in den Zug einsteigt oder aussteigt. Als Nächstes wären da die Scanner, welche auf das Infoterminal zugreifen, damit dieses erkennt, ob und welcher Zuggast gerade vor dem Terminal steht. Zu guter Letzt, gibt es die Scanner für das Infotainmentsystem. Diese sorgen dafür, dass das System weiß, das sich der Zuggast auf einen Platz gesetzt hat. Daher entfällt auch ein Scanner für die Sitzplatzverwaltung, da über das Infotainmentsystem eine Nachricht an die Sitzplatzverwaltung geschickt wird, dass sich der Zuggast gesetzt hat. Das gleiche gilt natürlich auch für das Verlassen des Platzes. Zusammenfassend kann man also sagen, dass die Anzahl der Scanner im Zug, die Summe der Schleusen, Infoterminals und Sitzplätze sind.

Sowohl das Infotainmentsystem als auch das Infoterminal brauchen natürlich eine Möglichkeit, die programmierte UI darzustellen. Dies wird über Touchscreens erfolgen. Diese sind gleichzeitig eine Input und Output Möglichkeit für das System. Das Infotainmentsystem benötigt weiterhin einen Kopfhöreranschluss, welcher dazu dient, damit der Zuggast seine mitgebrachten Kopfhörer anschließen kann. Eine Anforderung das Bluetooth-Kopfhörer angeschlossen werden können, gibt es nicht. Es ist trotzdem sinnvoll, diesen Punkt im Hinterkopf zu haben, da in der heutigen Zeit, die meisten Menschen eher auf ein Funkgerät setzen als auf ein Kabelgerät (Stichwort: Erweiterbarkeit des Systems in Zukunft). Abschließend wird noch eine Folientastatur benötigt. Diese geht aus den Anforderungen des Kunden hervor. Dies ist auch sinnvoll, da es dem Zuggast dann freisteht, auf dem Touchscreen die Eingabe zu tätigen, was bei einer kleinen Displaygröße schwierig werden könnte, oder ob er auf einer, ihm vertraute Eingabemöglichkeit, Folientastatur den Input an das System übergibt.

TI Architektur

Die TI-Architektur, welche auch Architektur der technischen Infrastruktur genannt wird, dient zur Darstellung und Festlegung, auf was für Software das System läuft.

Für das Bahn 2.0 System, haben wir uns für drei eigenständige Laufzeitumgebungen entschieden (Abbildung 6). Wir haben zum einen eine Virtuelle Maschine (VM), auf der ein Datenbankserver läuft. Diese befindet sich nicht innerhalb des Zuges, sondern beispielsweise in einer Zentrale der Bahn. Dort befinden sich alle wichtigen Daten, wie z.B. die Informationen des Zuggastes. Unserer Meinung nach, wäre es nicht sinnvoll, ein größeres Datenbanksystem im Zug selbst laufen zu lassen. Da es der Bahn möglich ist, zu erkennen, ob ein Zuggast gerade einsteigt oder den Zug verlässt, werden die Daten beim Betreten des Zuges temporär im Zug gespeichert. Beim Verlassen werden diese Daten dann wieder gelöscht. Dies hat den zum einen Vorteil, dass die Daten sehr schnell geladen werden können, da der Zug sich gerade an einem Bahnhof befindet. Außerdem ist die Datenbank nur dann ausgelastet, wenn Züge in einem Bahnhof sind. Somit greifen nicht alle Züge gleichzeitig auf die Datenbank zu.



Abbildung 6 Infrastruktur des Bahn 2.0 Systems

Alle relevanten Systeme, die für die Funktionen im Zug selbst verantwortlich sind, benötigen eine VM mit einem laufenden Applikationsserver. Auf diesem werden dann die folgenden Komponenten deployt: Infoterminal, Sitzplatzverwaltung, Infotainmentsystem, RFID-System. Dies sind alles Komponenten, die nicht auf einem externen Server (also außerhalb der Bahn) laufen sollten, da sich ihre Funktion, immer an einen speziellen Gast anpasst, somit muss die Sitzplatzverwaltung nicht mehrere Gigabyte an Daten gleichzeitig verarbeiten(Statistik der DB einbauen), sondern nur die, die wichtig für den aktuellen Zug (uns seine Gäste) sind.

Abschließend bleiben noch Systeme bzw. Komponenten, die nicht im Zug selbst laufen müssen, da diese seltener benutzt werden. Diese können somit in einer Zentrale der Bahn laufen. Da diese Komponenten jedoch unterschiedliche Domänen abdecken, werden für die Komponenten Profilverwaltung und Abrechnungssystem jeweils eine unterschiedliche VM und Applikationsserver benötigt. Grundsätzlich könnten diese Systeme jedoch auf derselben Hardware laufen.  
Der Vollständigkeit, haben die Mediathek-Komponente mit in diese Architektur aufgenommen. Da aus den Anforderungen nicht ersichtlich ist, ob diese über einen Drittanbieter bereitgestellt wird oder ob es sich um eine Mediathek der Bahn handeln wird. Ist erstes der Fall, wird diese Komponente aus der Abbildung 6 entfernt. Sollte jedoch Zweites der Fall sein, wird dieses System, welches größtenteils aus Video-Dateien bestehen wird, auf einer VM installiert sein. Da sich hinter dieser Komponente nicht sehr viel Logik und Komplexität verbergen wird, braucht man an dieser Stelle keinen Applikationsserver, da hier nur Daten abgefragt werden. Diese Menge an Daten sollte aber nicht dauerhaft in einem Zug gespeichert sein, da dies sehr Ressourcen aufwendig wäre.

## Aufgabe 2 - Nicht funktionale Anforderungen

### Nicht funktionale Anforderungen

Die nicht funktionalen Anforderungen (NFA) beziehen sich darauf, im Gegensatz zu den funktionalen Anforderungen, wie das System seine Leistung erbringen soll.[[1]](#footnote-1) Die funktionalen Anforderungen beschreiben gewünschte Funktionalitäten (was soll das System tun / können) eines Systems bzw. des Produktes.[[2]](#footnote-2) Deshalb werden die NFAs oft vom Auftragsgeber vergessen, außer sie haben in der Vergangenheit in diesem Bereich schlechte Erfahrungen gemacht (z.B. zu geringe Verfügbarkeit oder schlechte Qualität). Zusätzlich wirken sich die nicht funktionalen Anforderungen auf die gesamte Architektur aus.

Jedoch stellt sich die Frage, wie man die Qualität einer Software bestimmt. Dafür wurde unteranderem die ISO-Norm 9126 erstellt. Es wird zwischen sechs Kriterien unterschieden. (Abbildung 7)

1. Anforderungen an die **Benutzbarkeit** des Systems (Welchen Aufwand fordert der Einsatz der Software von den Benutzern und wie wird die Software von diesen beurteilt?)
2. Anforderungen an die **Änderbarkeit** des Systems (Welchen Aufwand erfordert die Durchführung vorgegebener Änderungen an die Software?

Abbildung 7 Softwarequalität nach ISO-Norm 9126  
(Quelle: Capgemini Präsentation vom 18.06.2019, Handout.pptx, Seite 54)

1. Anforderungen an die **Effizienz** des Systems (Wie liegt das Verhältnis zwischen Leistungsniveau der Software und den eingesetzten Betriebsmitteln?
2. Anforderungen an die **Funktionalität** des Systems (Inwieweit besitzt die Software die geforderten Funktionen?)
3. Anforderungen an die **Übertragbarkeit** des Systems (Wie leicht lässt sich die Software in eine andere Umgebung übertragen?)
4. Anforderungen an die **Zuverlässigkeit** des Systems (Kann die Software ein bestimmtes Leistungsniveau unter bestimmten Bedingungen über einen bestimmten Zeitraum aufrechterhalten?)[[3]](#footnote-3)

Nichtsdestotrotz gibt es Probleme zwischen den einzelnen Merkmalen. Denn in der Regel ist es nicht möglich alle Qualitätskriterien gleichermaßen zu verbessern. Es gibt unteranderem Zielkonflikte der unterschiedlichen Kriterien. Dementsprechend ist es immer notwendig bei der Qualitätssicherung einer Software Schwerpunkte zu setzen und diese zu priorisieren.

Neben den beiden oben genannten Typen gibt es einen weiteren Typen, nämlich die Randbedingungen. Typische Randbedingungen sind eine Obergrenze von Kosten oder einzuhaltende Termine für den Abschluss des Projektes. [[4]](#footnote-4)

1. https://www.uni-marburg.de/fb12/arbeitsgruppen/swt/lehre/files/est1415/EST141028.pdf [↑](#footnote-ref-1)
2. http://www.anforderungsmanagement.ch/in\_depth\_vertiefung/funktionale\_nicht\_funktionale\_anforderungen/index.html [↑](#footnote-ref-2)
3. https://de.wikipedia.org/wiki/ISO/IEC\_9126 [↑](#footnote-ref-3)
4. https://de.wikipedia.org/wiki/Anforderung\_(Informatik) [↑](#footnote-ref-4)