

(12) NACH DEM VERTRAG ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES
PATENTWESENS (PCT) VERÖFFENTLICHTE INTERNATIONALE ANMELDUNG

(19) Weltorganisation für geistiges
Eigentum

Internationales Büro

(43) Internationales
Veröffentlichungsdatum
2. April 2015 (02.04.2015)



(10) Internationale Veröffentlichungsnummer
WO 2015/043640 A1

(51) Internationale Patentklassifikation:
G06F 19/00 (2011.01)

(21) Internationales Aktenzeichen: PCT/EP2013/070097

(22) Internationales Anmeldedatum:
26. September 2013 (26.09.2013)

(25) Einreichungssprache: Deutsch

(26) Veröffentlichungssprache: Deutsch

(71) Anmelder: **SIEMENS AKTIENGESELLSCHAFT**
[DE/DE]; Wittelsbacherplatz 2, 80333 München (DE).

(72) Erfinder: **DORN, Karlheinz**; Erlenstr. 29, 90562
Kalchreuth (DE).

(81) Bestimmungsstaaten (soweit nicht anders angegeben, für
jede verfügbare nationale Schutzrechtsart): AE, AG, AL,
AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW,
BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK,
DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM,
GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KN, KP,

KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD,
ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI,
NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU,
RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH,
TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA,
ZM, ZW.

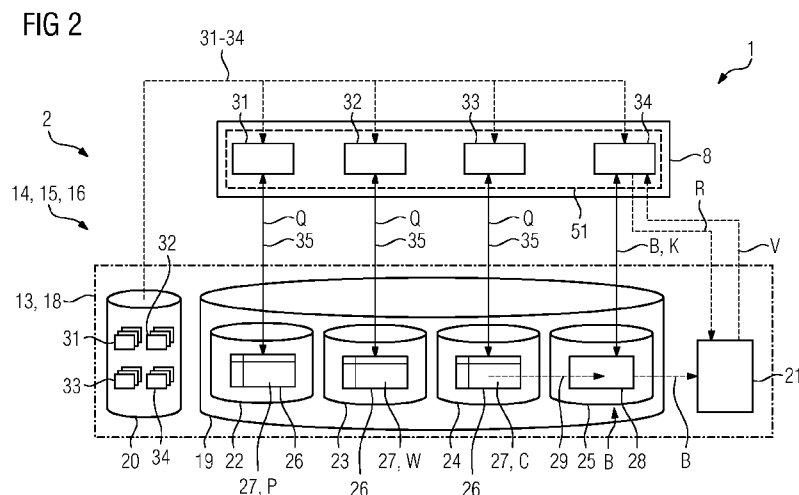
(84) Bestimmungsstaaten (soweit nicht anders angegeben, für
jede verfügbare regionale Schutzrechtsart): ARIPO (BW,
GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ,
TZ, UG, ZM, ZW), eurasisches (AM, AZ, BY, KG, KZ,
RU, TJ, TM), europäisches (AL, AT, BE, BG, CH, CY,
CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT,
LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE,
SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA,
GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Veröffentlicht:

— mit internationalem Recherchenbericht (Artikel 21 Absatz
3)

(54) Title: METHOD AND SYSTEM FOR MANAGING AND EDITING DATA OF A MEDICAL DEVICE

(54) Bezeichnung : VERFAHREN UND SYSTEM ZUR VERWALTUNG UND BEARBEITUNG VON DATEN EINER
MEDIZINISCHEN EINRICHTUNG



(57) Abstract: The invention relates to a method and to a system for managing and editing data of a medical device (2, 2'). According to the invention, separate memory areas (22-25) in a cloud storage (19) of a public cloud (13) are provided for at least two different types of medical data sets (P, W, C, B). The data sets (P, W, C, B) stored in a particular memory area (22-25) are each held available in such a way that said data sets (P, W, C, B) can be directly accessed from the Internet (16) if a subscription (18, 18') associated with the memory area (22-25) exists. In addition, at least one specifically associated application (31-34, 94, 95) for running on a user device (8, 9, 11, 12) is provided for each memory area (22-25), said application being designed to retrieve and/or display the data sets (P, W, C, B) contained in the associated memory area (22-25).

(57) Zusammenfassung:

[Fortsetzung auf der nächsten Seite]



Es werden ein Verfahren und ein System zur Verwaltung und Bearbeitung von Daten einer medizinischen Einrichtung (2,2') angegeben. Dabei werden für mindestens zwei unterschiedliche Arten von medizinischen Datensätzen (P,W,C,B) jeweils separate Speicherbereiche (22-25) in einem Cloud Storage (19) einer Public Cloud (13) zur Verfügung gestellt. Die in dem jeweiligen Speicherbereich (22-25) gespeicherten Datensätze (P,W,C,B) werden jeweils derart vorgehalten, dass ein unmittelbarer Zugriff auf diese Datensätze (P,W,C,B) aus dem Internet (16) bei Vorliegen einer dem Speicherbereich (22-25) zugeordneten Subscription (18, 18') möglich ist. Zudem wird für jeden Speicherbereich (22-25) mindestens eine spezifisch zugeordnete Applikation (31-34, 94, 95) zum Ablauf auf einem Nutzer-Gerät (8, 9, 11, 12) zur Verfügung gestellt, die zum Abruf und/oder zur Darstellung der in dem zugeordneten Speicherbereich (22-25) enthaltenen Datensätze (P,W,C,B) eingerichtet ist.

Beschreibung

Verfahren und System zur Verwaltung und Bearbeitung von Daten einer medizinischen Einrichtung

5

Die Erfindung bezieht sich auf ein Verfahren zur Verwaltung und Bearbeitung von Daten einer medizinischen Einrichtung. Die Erfindung bezieht sich des Weiteren auf ein System zur Durchführung dieses Verfahrens.

10

Unter dem Begriff „Verwaltung“ wird hier und im Folgenden insbesondere die Archivierung der Daten (das heißt die Einstellung der Daten in einen persistenten Datenspeicher), die Wiedergabe (Anzeige) und das Löschen der Daten aus dem Datenspeicher sowie das Sortieren und Auffinden bestimmter Daten aus dem Datenspeicher nach Maßgabe vorgegebener Kriterien (Browsing) verstanden.

15

Unter dem Begriff „Bearbeitung“ wird hier und im Folgenden insbesondere die Änderung (Editierung / Aufbereitung) der Daten verstanden.

20

Die zu verwaltenden und zu bearbeitenden Daten der medizinischen Einrichtung umfassen insbesondere Patientendaten, Aufgaben (Works/Tasks) oder Aufgabenlisten (Worklists) für das Personal der medizinischen Einrichtung und medizinische Bilddaten.

25

Die Verwaltung solcher medizinischer Daten erfolgt in zunehmendem Maße computergestützt durch Server-Systeme, insbesondere sogenannte Informationssysteme. Ein Informationssystem umfasst regelmäßig

30

- einen oder mehrere Datenspeicher, beispielsweise in Form eines Storage Area Network (SAN),
- 35 - einen oder mehrere zugehörige Datenserver,
- mindestens eine relationale Datenbank, die in einem Datenbankserver implementiert ist,

- sowie in der Regel einen oder mehrere weitere Server, in denen Methoden für den Datenbankzugriff und die Datenbearbeitung implementiert sind.

5 Für die unterschiedlichen Datenarten haben sich im medizinischen Bereich unterschiedliche Informationssysteme etabliert. So werden im Umfeld einer medizinischen Einrichtung wie zum Beispiel einer Klinik regelmäßig ein „Krankenhaus-Informationssystem“ (Hospital Information System, HIS) für die Verwaltung und Bearbeitung der Patientendaten und ein „Radiologie-Informationssystem“ (RIS) für die Terminplanung radiologischer Untersuchungen, die Unterstützung der Befundung medizinischer Bilddaten und die Dokumentation der Befunde eingesetzt. Daneben umfasst die IT-Struktur eines Krankenhauses in
10 der Regel ein sogenanntes „Picture Archiving and Communication System“ (PACS) zur Archivierung und Übertragung von medizinischen Bilddaten auf Basis des DICOM-Standards sowie ein „Advanced Visualisation(AV)-System“, das servergestützte Funktionen zur Visualisierung von Volumendaten, insbesondere dynamisches Volume Rendering, zur Verfügung stellt.
15
20

Die vorstehend bezeichneten Server-Systeme sind dabei in der Regel parallel zueinander vorhanden. Dies erfordert einen hohen Beschaffungs- und Wartungsaufwand, der insbesondere für
25 kleine medizinische Einrichtungen oder sonstige Einrichtung mit vergleichsweise geringen Finanzvolumen kaum zu bewältigen ist.

Die vorstehend beschriebene, komplexe IT-Struktur einer modernen medizinischen Einrichtung weist zudem nur vergleichsweise schlechte Skalierungseigenschaften auf. Eine Anpassung einer solchen IT-Struktur an größere Änderungen des zu verarbeitenden und zu archivierenden Datenvolumens und/oder der erforderlichen Rechenleistung ist somit meist nur mit sehr
30
35 hohem Aufwand möglich.

Als Nutzer- oder Endgeräte (klassischerweise als Clients bezeichnet) einer solchen IT-Struktur werden überwiegende Per-

sonal Computer (PCs) verwendet, wobei diese PCs oft als sogenannte „Thin Clients“ ausgebildet sind, die einen Großteil der erforderlichen Rechenleistung von einem angeschlossenen Server beziehen.

5

Als Alternative zu herkömmlichen Client-Server-Architekturen haben sich in den vergangenen Jahren zunehmend sogenannte Cloud-Lösungen etabliert. Als „Cloud“ („Rechnerwolke“) wird dabei eine Datenverarbeitungseinrichtung verstanden, die von einem von dem Nutzer unabhängigen Cloud-Betreiber („Cloud Vendor“) zur Verfügung gestellt und betrieben wird. Der „Cloud Vendor“ stellt hierbei einer Vielzahl von Nutzern die Hardware und gegebenenfalls die Software der Cloud im Rahmen eines Nutzungsvertrags (Subscription) als Dienst zur Verfügung. Je nach dem Umfang der zur Verfügung gestellten Dienste unterscheidet man zwischen

- einem als „Infrastructure as a Service“ (IaaS) bezeichneten Nutzungsmuster, bei dem dem Nutzer lediglich Computerhardware (Rechner, Netzwerke und Speicher) der Cloud zur Verfügung gestellt wird, während der Nutzer für die in der Cloud betriebene Software in vollem Umfang selbst verantwortlich ist,
- einem als „Platform as a Service“ (PaaS) beschriebenen Nutzungsmuster, bei dem dem Nutzer aus der Cloud die Computerhardware zusammen mit einer darauf aufbauenden Programmierungs- und Laufzeitumgebung angeboten wird, so dass der Nutzer nur für die in dieser Programmierungs- und Laufzeitumgebung implementierte Anwendungssoftware (Applikationen) selbst verantwortlich ist, sowie
- einem als „Software as a Service“ (SaaS) bezeichneten Nutzungsmuster, bei dem dem Nutzer darüber hinaus auch bestimmte Anwendungssoftware aus der Cloud zur Verfügung gestellt wird.

35

Je nach dem Nutzerkreis, an den die jeweilige Cloud adressiert ist, unterscheidet man des Weiteren zwischen

- einer sogenannten Public Cloud, deren Dienste von jedem in Anspruch genommen werden können, und
- einer sogenannten Privat Cloud, die nur Nutzern einer bestimmten Organisation, insbesondere eines bestimmten Konzerns zugänglich ist.

Für jeden Nutzer einer Public Cloud sind die Zugriffsberechtigungen auf bestimmte Hardware- und Softwarebestandteile der Cloud durch die dem Nutzer zugeordnete Subscription geregelt.

Public Clouds sind hierdurch regelmäßig „mandantenfähig“ (multi-tenant). Dies bezeichnet die Fähigkeit, Daten, Benutzerverwaltung und Rechenoperationen für Nutzer mit verschiedener Subscription strikt getrennt zu halten. Ein Nutzer der Public Cloud kann also in die Daten, Benutzerverwaltung und Rechenoperationen eines anderen Nutzers mit unterschiedlicher Subscription keinen Einblick nehmen und diese Daten auch nicht beeinflussen.

Der Erfindung liegt die Aufgabe zugrunde, ein Verfahren und ein System zur Verwaltung und Bearbeitung von Daten einer medizinischen Einrichtung anzugeben, das besonders flexibel einsetzbar, insbesondere besonders skalierbar ist.

Bezüglich des Verfahrens wird diese Aufgabe erfindungsgemäß gelöst durch die Merkmale des Anspruchs 1. Bezüglich des zugehörigen Systems wird die Aufgabe erfindungsgemäß gelöst durch die Merkmale des Anspruchs 9. Vorteilhafte und teils für sich gesehen erfinderische Ausgestaltungen und Weiterentwicklungen der Erfindung sind in den Unteransprüchen und der nachfolgenden Beschreibung dargelegt.

Erfindungsgemäß werden für mindestens zwei unterschiedliche Arten von Daten der medizinischen Einrichtung jeweils separate Speicherbereiche in einem Cloud Storage (also dem persistenten Speicher) einer Public Cloud zur Verfügung gestellt, wobei die in dem jeweiligen Speicherbereich gespeicherten Datensätze derart vorgehalten werden, dass ein unmittelbarer Zugriff auf diese Datensätze aus dem Internet mög-

lich ist, wenn eine dem Speicherbereich zugeordnete Subscription vorliegt (wenn also das zugreifende Gerät nach Maßgabe der Subscription zum Zugriff berechtigt ist).

5 Diese Zugriffsmöglichkeit ist dabei insbesondere derart gestaltet, dass sie mit dem Internet-Standard REST kompatibel ist. REST (Representational State Transfer) bezeichnet dabei ein Programmierparadigma für Internet-Anwendungen, das folgenden Prinzipien genügt:

- 10 - Jedem angebotenen Dienst ist eine eindeutige Adresse (Uniform Resource Locator, kurz: URL) zugewiesen.
- Der unter einer Adresse zugängliche Dienst kann unter-
15 schiedliche Erscheinungsformen (Repräsentationen) haben.
- Es wird ein zustandsloses (stateless) Client-Server-Protokoll verwendet, das es ermöglicht, verschiedene Anfragen zwischen Clients und Servern stets unabhängig
20 voneinander zu behandeln.
- Auf alle Ressourcen muss eine vorgegebene Anzahl von Operationen (nämlich insbesondere die vier Operationen GET, POST, PUT, DELETE) mit definierten Eigenschaften
25 anwendbar sein.
- Sowohl für Anwendungsinformationen als auch für Zustandsveränderungen wird multimedialer Hypertext (Hypermedia) benutzt.

30 Die Repräsentation erfolgt dabei vorzugsweise gemäß den Internet-Standards XML (Extensible Markup Language), JSON (JavaScript Object Notation) und/oder ATOM. ATOM ist hierbei als Überbegriff verwendet für das Atom Syndication Format (ASF),
35 das ein XML-Format für den Austausch von Nachrichten darstellt, sowie das ATOM Publishing Protocol (APP), das eine Programmierschnittstelle zur Erzeugung und Bearbeitung von Webinhalten darstellt.

Bei der medizinischen Einrichtung, der jeweils eine Subscription zugeordnet ist, handelt es sich in dem vorrangig anvisierten Anwendungsfall um eine Klinik. Allerdings ist im Rahmen der Erfindung auch denkbar, dass kleinere Einheiten wie zum Beispiel Arztpraxen, einzelne Abteilungen einer Klinik oder sogar einzelne Patienten zur Verwaltung der jeweils eigenen medizinischen Daten eine eigene Subscription erwerben und somit das Verfahren bzw. System in vollem Umfang nützen können.

Das System ist hierbei durch die Nutzung der Public Cloud in dem vorstehend erläuterten Sinne „mandantenfähig“ (multi-tenant). Daten, Benutzerverwaltung und Rechenoperationen werden für Nutzer mit verschiedener Subscription somit strikt getrennt voneinander gehalten, so dass ein Einblick in oder eine Beeinflussung von Daten fremder Subscriptions ausgeschlossen ist.

Im Rahmen des erfindungsgemäßen Verfahrens und Systems wird des Weiteren für jeden Speicherbereich eine Applikation zum Ablauf auf einem Nutzer- oder Endgerät (nachfolgend als „Device“ bezeichnet) zur Verfügung gestellt, die zum Abruf und/oder zur Darstellung der in dem zugeordneten Speicherbereich enthaltenen Datensätze eingerichtet ist.

Jede Applikation ist dabei genau einem der Speicherbereiche spezifisch zugeordnet und implementiert somit selektiv und ausschließlich diejenigen Funktionen für die Verwaltung und Bearbeitung der in dem jeweiligen Speicherbereich enthaltenen Datensätze.

Die verschiedenen Arten von Daten der Einrichtung umfassen insbesondere

35

- Patientendaten, die persönliche und medizinische Angaben zu einem bestimmten Patienten der Einrichtung umfassen,

- insbesondere Angaben zum Namen, der Adresse, dem Alter, dem Geschlecht, früheren Diagnosen und Behandlungen,
- Aufgaben (Works/Tasks) und Aufgabenlisten (Worklists), die von einem oder mehreren medizinischen Nutzern (Ärzten, medizinischen Assistenzkräften, Pflegern etc.) der medizinischen Einrichtung durchzuführen sind,
 - medizinische Bilddaten, insbesondere zweidimensionale Bilder, zweidimensionale Bildsequenzen (Videos), Volumendaten (z.B. Computertomogramme) und/oder zeitliche Volumendaten (4D-Daten), aber auch Keyimages, Photos, Dokumente und dergleichen, sowie
 - Kontextinformation zu medizinischen Bilddaten, insbesondere Verweise auf den Speicherort von Bilddatensätzen sowie optional Meta-Daten zu Bilddatensätzen wie z.B. die der jeweiligen Bildaufnahme zugrundeliegenden Aufnahmeparameter.

Als „Keyimage“ wird hierbei ein Schnappschuss, d.h. ein (bearbeiteter oder unbearbeiteter) Ausschnitt aus einem medizinischen Bild- oder Volumendatensatz bezeichnet, der bei der Befundung dieser Daten zur Darstellung eines medizinischen Befundes abgeleitet wird. Solche Keyimages werden typischerweise als Bestandteil eines medizinischen Berichts (Reports) verwendet.

Jede dieser verschiedenen Datenarten wird nachfolgend auch als „Ressource“ bezeichnet. Der dieser Ressource jeweils zugewiesene Speicherbereich des Cloud Storage wird nachfolgend auch als „(Ressource-)Hub“ bezeichnet. In Zuordnung zu den oben beschriebenen Ressourcen werden im Rahmen des Verfahrens und Systems vorzugsweise mindestens zwei der nachfolgend aufgeführten (Ressource-)Hubs im Cloud Storage zur Verfügung gestellt:

- Ein Patienten-Speicherbereich (Patient-Hub), in dem jeder (Patienten-)Datensatz Patientendaten eines bestimmten Patienten enthält,

- ein Aufgabenlisten-Speicherbereich (Worklist-Hub), in dem jeder Datensatz eine Aufgabe (Work/Task) oder Aufgabenliste (Worklist) enthält,
- einen Bilddaten-Speicherbereich (Image-Hub), in dem jeder (Bild-)Datensatz medizinische Bild- oder Volumendaten oder eine sonstige Bild-, Ton-, Graphik- oder Textinformation (Photos, Videos, Dokumente) enthält und/oder
- einen Kontext-Speicherbereich (Context-Hub), in dem jeder (Kontext-)Datensatz Kontextinformation zu einem Bilddatensatz enthält, insbesondere einen Verweis auf den Speicherort des jeweiligen Bilddatensatzes im Image-Hub sowie optional Meta-Daten zu dem Bilddatensatz.

Jeder der vorstehend beschriebenen Hubs ist zusammen mit der ihm spezifisch zugeordneten Applikation selbstständig (d.h. unabhängig von den anderen Hubs und den diesen jeweils zugeordneten Applikationen) betreibbar. Somit ist keiner der vorstehend beschriebenen Hubs und der zugeordneten Applikationen zur Durchführung des erfindungsgemäßen Verfahrens oder im Rahmen des erfindungsgemäßen Systems zwingend notwendig. Jeder der vorstehend beschriebenen Hubs erweitert aber zusammen mit der zugeordneten Applikation die Funktionalität des Verfahrens bzw. Systems in vorteilhafter Weise.

Die Erfindung geht von der Überlegung aus, dass aus Gründen einer verbesserten Skalierbarkeit für die Datenverwaltung und -bearbeitung in einer medizinischen Einrichtung ein Wechsel von einer klassischen Client-Server-Lösung hin zu einer cloud-basierten Lösung sinnvoll ist. Erkanntermaßen werden durch eine bloße Übertragung der klassischerweise gebräuchlichen Server-Systeme in die Cloud die eingangs beschriebenen Probleme aber nicht in befriedigender Weise gelöst. So wird durch die Verlagerung der herkömmlicherweise in einer medizinischen Einrichtung verwendeten Server in die Cloud die Komplexität der IT-Struktur nicht vereinfacht, sondern eher erhöht, zumal die Server in einer Cloud regelmäßig in virtualisierter Form betrieben werden, und somit zu der unverändert komplexen Serverstruktur noch die Virtualisierungs-Software

hinzu käme. Zudem würden erkanntermaßen bei einer solchen Lösung hohe Kosten für die in der Cloud erbrachte Serverleistung (Cloud Virtual Server) anfallen, die die Vorteile einer Cloud-Nutzung weitgehend aufwiegen würden. Ferner müssten die
5 virtualisiert in der Cloud betriebenen Server noch immer durch die Einrichtung selbst administriert werden, so dass der damit verbundene Aufwand bestehen bleiben würde. Auch hinsichtlich Skalierbarkeit und Ausfallsicherheit brächte eine solche Lösung keine wesentliche Verbesserung.

10

Eine solche Verbesserung wird allerdings durch die erfinderische Idee erzielt, die medizinischen Daten derart in dem Public Cloud Storage zu hinterlegen, dass auf diese Daten - insbesondere in REST-kompatibler Weise - über das Internet zugegriffen werden kann. Dies ermöglicht es, zumindest einen
15 Großteil der zur Durchführung des Verfahrens erforderlichen Rechenleistung in den an der Verfahrensdurchführung beteiligten Devices selbst zu erbringen. Somit wird im Zuge des Verfahrens ausschließlich oder zumindest nahezu ausschließlich der - vergleichsweise preisgünstige - Cloud Storage der Public Cloud in Anspruch genommen, während die wesentlich teurere Rechenleistung der Public Cloud (Cloud-Compute-Dienste)
20 nicht oder nur in geringem Maße in Anspruch genommen wird. Die ausschließliche oder zumindest nahezu ausschließliche Zuordnung der Rechenleistung zu den Devices trägt auch zu einer wesentlich verbesserten Skalierbarkeit des Verfahrens und des zugeordneten Systems bei, da die cloud-seitig implementierten Strukturen an einen veränderten Bedarf an Rechenleistung nicht angepasst werden müssen. Insbesondere müssen keine virtuellen Server in der Cloud geschaffen und betreut werden.
25 Vielmehr werden die optional in Anspruch genommenen Cloud-Compute-Dienste unmittelbar von dem Cloud-Provider im Sinne eines PaaS-Konzepts betreut und zur Verfügung gestellt.

30

35 Die Verteilung der verschiedenen Ressourcen auf verschiedene, vollständig voneinander separierte Ressource-Hubs, und die Zurverfügungstellung der jeweils diesen Hubs spezifisch zugeordneten Applikationen ermöglicht dabei einen besonders fle-

xiblen Einsatz des Verfahrens und des zugehörigen Systems, zumal verschiedene medizinische Einrichtungen die im Rahmen ihrer jeweiligen Subscription enthaltenen Hubs frei auswählen können, ohne gegenseitige Abhängigkeiten berücksichtigen zu müssen. Hierdurch sind für die medizinische Einrichtung wiederum die Kosten für den Betrieb des Verfahrens und des zugeordneten Systems flexibel regulierbar.

Durch die Nutzung der Public Cloud zur Archivierung und Vorhaltung der Daten profitiert die medizinische Einrichtung dabei von den üblichen Vorteilen eines Public Cloud Storage, nämlich einer nahezu unbegrenzten Skalierbarkeit (das heißt der Möglichkeit, nahezu beliebige Datenmengen ohne qualitative Änderungen der IT-Struktur effizient zu verwalten), eine weltweite Verfügbarkeit der Daten, eine äußerst hohe Sicherheit gegen Ausfall und Manipulation sowie eine nahezu vollständige Reduzierung des einrichtungsinternen Beschaffungs- und Wartungsaufwandes.

Zusammenfassend wird durch die Erfindung eine im medizinischen Bereich vom Üblichen vollständig abweichende IT-Struktur angegeben, die es ermöglicht, die Vorteile einer Public Cloud gegenüber einer herkömmlichen Client-Server-Lösung effizient zu nutzen, ohne dafür nennenswerte Nachteile in Kauf nehmen zu müssen.

Als Public Cloud wird vorzugsweise der „Windows Azure“-Dienst der Fa. Microsoft genutzt. Andere Public Clouds können im Rahmen der Erfindung aber auch verwendet werden. Insbesondere können auch mehrere Publics Clouds von gegebenenfalls unterschiedlichen „Cloud Vendors“ parallel (gleichzeitig) zur Verwaltung der Daten der Einrichtung herangezogen werden.

Bei den an der Verfahrensdurchführung beteiligten Devices der Einrichtung kann es sich in beliebiger Zusammensetzung um Desktop-PCs, aber auch um mobile Geräte wie Notebooks, Tablet-Computer und Smartphones handeln.

Vorzugsweise sind die Applikationen dazu eingerichtet, auf die in dem jeweils zugeordneten Hub enthaltenen Datensätze unmittelbar zuzugreifen. Zusätzlich oder alternativ sind die Applikationen vorzugsweise dazu eingerichtet, die in dem je-
5 weils zugeordneten Hub enthaltenen Datensätze unmittelbar zu erzeugen, zu editieren und zu löschen. Der Begriff „unmittelbar“ wird hier und im Folgenden jeweils in dem Sinne verwendet, dass die Applikation die jeweilige Aktion selbstständig unter ausschließlicher Nutzung des Cloud Storage ausführt,
10 ohne dass hierfür - von der reinen Speicherverwaltung (Cloud Storage) unabhängige - Rechenleistung der Public Cloud (Cloud-Compute-Dienste) in Anspruch genommen wird.

Abweichend ist im Rahmen der Erfindung aber vorzugsweise mindestens eine der Applikationen in zwei Betriebsmodi betreib-
15 bar, nämlich

- in einem „Device-Only-Modus“, in dem die Applikation ausschließlich auf einem Device abläuft, unmittelbar auf
20 den Cloud Storage zugreift und keine Cloud-Compute-Dienste der Public Cloud in Anspruch nimmt, sowie
- in einem „Cloud-Service-Modus“, in dem die Applikation zum Zugriff und/oder zur Darstellung der in dem Hub enthaltenen Datensätze zusätzlich Cloud-Compute-Dienste der
25 Public Cloud heranzieht.

Dieser - optional zuschaltbare - Cloud-Service-Modus wird insbesondere bei einer sogenannten 3D-Viewing-Applikation vorgesehen, die zur - besonders rechenleistungsintensiven -
30 Darstellung von gerenderten Ansichten (Szenen) von Volumendaten dient. Die Rechenleistung wird dabei von dem Cloud Vendor insbesondere im Rahmen eines PaaS-Nutzungsschemas angeboten, so dass für das Betriebssystem, Treiber, einen Applikationsserver (App-Store) und dergleichen keine separaten Adminis-
35 trationskosten anfallen.

Im Rahmen der Erfindung kann vorgesehen sein, dass die Auswahl zwischen dem Device-Only-Modus und dem Cloud-Service-Modus

- 5 - entweder automatisch durch die Applikation vorgenommen wird (beispielsweise in Abhängigkeit der Qualität der Netzverbindung und/oder der Rechenkapazität des Device),
- interaktiv durch den medizinischen Nutzer bestimmt wird, oder
- 10 - durch Voreinstellung (Konfigurationsdaten) der Applikation festgelegt wird.

Vorzugsweise ist innerhalb mindestens eines Hubs ein Tabellenspeicher (Table Storage) angelegt. Hierunter wird eine -
15 beispielsweise im Rahmen des „Windows Azure“-Dienstes der Firma Microsoft standardmäßig angebotene - Form der Datenspeicherung verstanden, in der Dateneinheiten (Data Entities) zusammen mit einer Index-Variable, die jede Dateneinheit eindeutig kennzeichnet, als Tabelleneinträge strukturiert abgelegt werden. Bei dem „Windows Azure“-Dienst setzt sich diese
20 Index-Variable beispielsweise aus einem sogenannten PartitionKey sowie einem sogenannten RowKey zusammen. Der PartitionKey bezeichnet hierbei eine physikalische Partition des Cloud Storage, in der der jeweilige Tabelleneintrag abgelegt wird,
25 so dass Tabelleneinträge mit verschiedenem PartitionKey in verschiedenen physikalischen Partitionen abgelegt werden.

Innerhalb des Table Storage ist jedem der in dem jeweiligen Hub gespeicherten Datensätze ein Tabelleneintrag (nachfolgend
30 als „Tweet“ bezeichnet) zugeordnet. Jeder Tweet kann hierbei den zugeordneten Datensatz selbst enthalten. So ist vorzugsweise jeder Satz von Patientendaten des Patient-Hubs vorzugsweise als Tweet des zugehörigen Tabellenspeichers hinterlegt. Alternativ hierzu (insbesondere bei Datensätzen mit großen
35 und/oder nicht-alphanumerischen Datenblöcken wie z.B. medizinischen Bildern) ist der Datensatz selbst tabellenextern (außerhalb eines Table Storage) hinterlegt, beispielsweise in einem sogenannten Blob-Speicher (Blob Storage) der Cloud Sto-

rage. Der dem Datensatz zugeordnete Tweet enthält in diesem Fall eine Angabe zu dem tabellenexternen Speicherort, beispielsweise in Form eines sogenannten „Uniform Resource Identifier“ (URI). Wiederum alternativ hierzu können bestimmte Datenblöcke eines Datensatzes im Rahmen des Tweets in dem Tabellenspeicher abgelegt sein, während mindestens ein weiterer Datenblock desselben Datensatzes tabellenextern gespeichert ist. So sind beispielsweise im Falle von medizinischen Bilddatensätzen zweckmäßigerweise die eigentlichen Bilddaten in einem Blob-Storage des Image-Hubs gespeichert, während die dazugehörigen Metadaten (z.B. Aufnahmeparameter und Angaben zum Patienten) im Rahmen der zugehörigen Kontextinformation im Table Storage des Context-Hubs hinterlegt sind. Als Blob-Storage, wie er ebenfalls im Rahmen des „Windows Azure“ Dienstes der Firma Microsoft standardmäßig angeboten wird, wird eine Form der Datenspeicherung bezeichnet, bei der eine unstrukturierte Speicherung von Datensätzen erfolgt.

Verfahrensgemäß kann hierbei der Inhalt des Tabellenspeichers von der dem Hub zugeordneten Applikation unmittelbar (ohne Inanspruchnahme von Rechenleistung der Public Cloud, also Cloud-Compute-Diensten) durchsucht werden. Als Ergebnis dieser Suche werden hierbei ausgewählte Tweets in Form einer Ergebnisliste extrahiert, die von der Applikation angezeigt wird. Eine solche Ergebnisliste ist nachfolgend als „Feed“ bezeichnet.

Die vorstehend beschriebene Ausgestaltung der Applikationen und des von diesem durchgeführten Verfahrens ermöglicht hierbei ein effektives „Browsing“, d.h. eine effektive Sortierung und Auffindung der in dem jeweiligen Hub gespeicherten Daten, ohne hierfür Rechenleistung der Public Cloud, also Cloud-Compute-Dienste in Anspruch zu nehmen.

In einer besonders bevorzugten Ausgestaltungsvariante des Verfahrens wird durch mindestens eine der Applikationen der von dieser aus dem zugehörigen Hub extrahierte Feed „externalisiert“, das heißt aus dem Public Cloud Storage in einen lo-

kalen Speicher des Device heruntergeladen, so dass dieser Feed und die darin enthaltenen Tweets durch die Applikation auch ohne Netzanbindung mit der Public Cloud angezeigt, bearbeitet und/oder an eine andere Instanz derselben oder einer
5 anderen Applikation übermittelt werden können.

Durch diesen Austauschmechanismus (auch als „Exchangeboard“ bezeichnet) wird eine besonders effektive und flexible Kommunikation zwischen verschiedenen Applikationsinstanzen ermöglicht, die wiederum ohne Rechenleistung der Public Cloud, also Cloud-Compute-Dienste auskommt. Beispielsweise kann auf diese Weise mittels einer dem Context-Hub zugeordneten Browsing-Applikation der URI eines anzuschauenden Bildes ermittelt und an eine davon separate Viewing-Applikation des
10 Image-Hub übermittelt werden, so dass die Viewing-Applikation dann selbständig anhand des URI auf das gesuchte Bild zugreifen kann. Dieser Austausch kann ohne Weiteres auch zwischen verschiedenen Devices erfolgen, beispielsweise indem der dem Bilddatensatz zugeordnete Tweet oder/oder die darin enthaltene URI per Email, Instant-Message oder Twitter versandt werden.
15 20

In einer weiteren vorteilhaften Weiterentwicklung des Verfahrens und des Systems sind auch Modalitäten der medizinischen Einrichtung (das heißt, die bildgebenden Geräte wie z.B. Computertomographen, Magnetresonanztomographen, etc.) zur unmittelbaren Kommunikation mit dem Cloud Storage ertüchtigt. Im Zuge des Verfahrens wird somit durch eine medizinische Modalität ein Datensatz, der patientenspezifische Bilddaten enthält, erzeugt und unmittelbar in dem zugeordneten Speicherbereich (insbesondere dem Image-Hub) des Cloud Storage abgelegt. Diese Funktionalität zur direkten Kommunikation mit dem Public Cloud Storage ist insbesondere in einem Speichertreiber implementiert, der der Steuersoftware der Modalität im
25 30 35 Rahmen einer Anwender-Programmier-Schnittstelle (Application Programming Interface, kurz: API) zur Verfügung gestellt wird. Der Speichertreiber kann unmittelbar in die Steuersoftware der jeweiligen Modalität integriert sein. Alternativ

hierzu bildet der Speichertreiber einen eigenständigen Programmbestandteil, auf den die Steuersoftware der Modalität nach Art eines Druckertreibers zugreift. Die vorstehend beschriebene Ausbildung einer Modalität zum unmittelbaren Laden
5 der von der Modalität erzeugten Bilddatensätze in einen zugeordneten Speicherbereich (Image-Hub) eines Public Cloud Storage wird als eigenständige Erfindung angesehen, die auch ohne die übrigen Merkmale vorstehend beschriebenen Verfahrens bzw. Systems vorteilhaft einsetzbar ist.

10

Das erfindungsgemäße System zur Verwaltung und Bearbeitung von Daten einer medizinischen Einrichtung ist schaltungs- oder programmtechnisch zur Durchführung des vorstehend beschriebenen Verfahrens eingerichtet. Es umfasst hierzu mindestens zwei (Ressource-)Hubs der vorstehend beschriebenen
15 Art zur Speicherung mindestens zweier Ressourcen (d.h. zweier unterschiedlicher Arten von medizinischen Datensätzen), wobei jeder Hub ausschließlich den Datensätzen einer bestimmten Ressource zugewiesen ist. Die Daten sind hierbei in dem jeweiligen Hub derart vorgehalten, dass auf diese Daten aus dem Internet unmittelbar zugegriffen werden kann, sofern eine dem Hub zugeordnete Subscription vorliegt. Jedem Hub ist hierbei mindestens eine Applikation zugeordnet, die selektiv zum Ab-
20 ruf und zur Darstellung der in dem zugeordneten Hub enthaltenen Datensätze eingerichtet ist. Die vorstehend im Zusammenhang mit dem Verfahren beschriebenen Ausgestaltungsvarianten und die damit verbundenen Vorteile sind sinngemäß auf das erfindungsgemäße System zu übertragen.

25

30

Im engeren Sinne umfasst das erfindungsgemäße System ausschließlich die genannten Speicherbereiche (Hubs), die zugeordneten Applikationen sowie optional den vorstehend beschriebenen Speichertreiber. Das System ist in diesem Sinne ausschließlich aus Softwarekomponenten gebildet. In einem

35

weiteren Sinne umfasst das erfindungsgemäße System zusätzlich den Cloud Storage, bei dem es sich um einen Hardware-Speicher mit einer darauf laufenden, cloud-internen Verwaltungssoftware handelt, sowie mindestens ein Device, das mit dem Public

Cloud Storage über das Internet verbunden ist. Bei dem Device handelt es sich insbesondere um einen Personal-Computer (PC), ein Notebook, einen Tablet-Computer oder ein Smartphone.

- 5 Nachfolgend werden Ausführungsbeispiele der Erfindung anhand einer Zeichnung näher erläutert. Darin zeigen:

- 10 FIG 1 in einem schematischen Blockschaltbild ein System zur Verwaltung und Bearbeitung von Daten einer medizinischen Einrichtung, mit mehreren Modalitäten und Nutzer- oder Endgeräten (Devices) sowie mit einer mit den Modalitäten und Devices über das Internet verbundenen Public Cloud,
- 15 FIG 2 in einem schematischen Blockschaltbild den softwaretechnischen Aufbau des Systems, wonach das System für verschiedene Arten von Daten der medizinischen Einrichtung in einem Cloud Storage der Public Cloud jeweils spezifisch zugeordnete Speicherbereiche (Hubs) aufweist, nämlich einen Patient-Hub für Patientendatensätze, einen Worklist-Hub für Aufgaben, einen Context-Hub für Kontextdatensätze und einen Image-Hub für Bilddatensätze, und wobei das System auf jedem Device für jeden Hub eine zugeordnete Applikation zum Zugriff auf die in diesem Hub gespeicherten Datensätze enthält, wobei in den drei erstgenannten Hubs jeweils ein Tabellenspeicher (Table Storage) angelegt ist, in dem zu jedem Datensatz des Hubs ein zugeordneter Tabelleneintrag (Tweet) enthalten ist, und wobei jede Applikation dazu eingerichtet, den Tabellenspeicher des zugehörigen Hubs zu durchsuchen (Browsing) und eine Ergebnisliste (Feed) ausgewählter Tweets zu extrahieren und anzuzeigen, und wobei die dem Image-Hub zugeordnete Applikation zur Anzeige (Viewing) der Bilddatensätze eingerichtet ist,
- 20
- 25
- 30
- 35

FIG 3 in schematischer Darstellung eine Seite einer Benutzeroberfläche einer zum Ablauf auf einem Smartphone vorgesehenen Applikation, die zur Anzeige eines Feeds des jeweils zugeordneten Hubs dient,

5

FIG 4 in Darstellung gemäß FIG 3 eine weitere Seite der Benutzeroberfläche der dortigen Applikation, die zur Anzeige eines Tweets des zugehörigen Hubs dient,

10

FIG 5 in Darstellung gemäß FIG 3 eine Seite der Benutzeroberfläche der dem Image-Hub zugeordneten Applikationen, die zur Darstellung der Bildinformation eines Bilddatensatzes dient,

15

FIG 6 in Darstellung gemäß FIG 3 eine zum Ablauf auf einem Personal-Computer, Notebook oder Tablet-Computer vorgesehene Variante der dortigen Applikation, bei der mehrere Seiten der Benutzeroberfläche gleichzeitig darstellbar sind, und

20

FIG 7 in Darstellung gemäß FIG 6 eine weitere Seite der Benutzeroberfläche, die zur Darstellung der Bildinformation eines Bilddatensatzes dient,

25

FIG 8 in einem schematischen Blockschaltbild eine alternative Ausführung des Systems, bei dem die dem Image-Hub zugeordnete Viewing-Applikation getrennte Module zum Anzeigen von 2D-Bilddaten und Volumendaten enthält,

30

FIG 9 in Darstellung gemäß FIG 8 eine weitere Ausführungsform des Systems, wobei dem Image-Hub zwei selbstständige Applikationen zugewiesen sind, nämlich eine 2D-Viewing-Applikation zum Anzeigen von 2D-Bildern und eine 3D-Viewing-Applikation zur visuellen Darstellung von Volumendaten, und

35

FIG 10 in Darstellung gemäß FIG 8 eine weitere Ausführungsform des Systems, bei der eine Modalität mit einem Speichertreiber zum unmittelbaren Laden von Bilddaten in den Image-Hub des Cloud Storage ausgestattet ist.

Einander entsprechende Teile und Größen sind in allen Figuren stets mit gleichen Bezugszeichen versehen.

FIG 1 zeigt in grober schematischer Vereinfachung ein System 1 zur Verwaltung und Bearbeitung von medizinischen Daten einer medizinischen Einrichtung 2, bei der es sich beispielsweise um eine Klinik handelt.

Hardwareseitig umfasst das System 1 eine Anzahl von Modalitäten 3, also bildgebenden, medizinischen Untersuchungsgeräten der Einrichtung 2. So umfasst das System 1 in der beispielhaften Darstellung gemäß FIG 1 einen Computertomographen 4, ein C-Bogengerät 5 und einen Magnetresonanztomographen 6. Jeder Modalität 3 ist ein (Steuer- und Auswerte-)Rechner 7 zugeordnet.

Weiterhin umfasst das System 1 hardwareseitig eine Anzahl von Nutzer- oder Endgeräten (nachfolgend Devices 8) der Einrichtung 2, die zur Anzeige und Bearbeitung von Daten dienen. In dem vereinfachten Beispiel gemäß FIG 1 umfassen die Devices 8 einen Personalcomputer 9 mit angeschlossenem Bildschirm 10, einen Tablet-Computer 11 sowie ein Smartphone 12.

Als außerhalb der Einrichtung 2 angeordneten Bestandteil umfasst das System 1 eine Public Cloud 13. Als Public Cloud 13 wird im Rahmen des System 1 beispielsweise der von der Firma Microsoft unter der Bezeichnung „Windows Azure“ angebotene Dienst genutzt. Abweichend hiervon können im Rahmen der Erfindung allerdings auch eine andere Public Cloud oder eine Kombination mehrerer Public Clouds (gegebenenfalls auch von unterschiedlichen Anbietern) als Public Cloud 13 herangezogen werden.

Die Public Cloud 13 ist über ein (Datenübertragungs-)Netz 14 mit den einrichtungsinternen Komponenten des Systems 1, also den Modalitäten 3 und Devices 8 verbunden. Diese Netzverbindung ist innerhalb der Einrichtung 2 durch ein Intranet 15 der Einrichtung 2 gebildet, das beispielweise als sogenanntes Local Area Network (LAN) auf Basis kabelgebundener Ethernet-Technologie und/oder als kabelloses Wireless Local Area Network (WLAN) aufgebaut ist. Außerhalb der Einrichtung 2 ist das Netz 14 durch das Internet 16 gebildet. An der Schnittstelle zwischen dem Intranet 15 und Internet 16 ist in üblicher Weise eine Firewall 17 angeordnet.

Die von der Public Cloud 13 im Rahmen des Systems 1 zur Verfügung gestellten Dienste sind durch einen als Subscription 18 bezeichneten Nutzungsvertrag festgelegt. Die Subscription 18 regelt dabei, welche Hardware- und Softwarebestandteile der Public Cloud 13 den einrichtungsinternen Komponenten des Systems 1 zugänglich sind. Mit dem Begriff Subscription 18 wird daher nachfolgend derjenige Teil der Public Cloud 13 bezeichnet, der der Einrichtung 2 im Rahmen des Systems 1 exklusiv zugeordnet ist. Andere Bereiche der Public Cloud 13 können – wie in FIG 1 angedeutet – im Rahmen weiterer Subscriptions 18' weiteren medizinischen Einrichtungen 2' zugewiesen sein. Jede Einrichtung 2,2' hat hierbei ausschließlich Zugang zu den ihr gemäß ihrer Subscription 18,18' zugewiesenen Daten und Diensten, nicht aber zu den Daten und Diensten anderer Einrichtungen 2 bzw. 2'. Die Public Cloud 13 ist in diesem Sinne „mandantenfähig“ (multi-tenant).

Gemäß der Subscription 18 sind dem System 1 der Einrichtung 2 innerhalb der Public Cloud 13

- ein nachfolgend als Cloud Storage 19 bezeichneter Datenspeicher,
- ein nachfolgend als App Store 20 bezeichneter Speicher für Applikationen (Anwendungssoftware) sowie
- Cloud-Compute-Dienste 21

zur Verfügung gestellt. Der Cloud Storage 19 dient zur persistenten Speicherung der Daten der Einrichtung 2. Der App Store 20 stellt Applikationen, Treiber und sonstige Peripheriesoftware wie beispielsweise Konfigurationsdateien und Vorlagen (Templates) zur Verfügung, die von den einrichtungsin-
5 ternen Komponenten des Systems 1, also den Modalitäten 3 und den Devices 8 heruntergeladen werden können und im Betrieb des Systems 1 auf den Modalitäten 3 (genauer gesagt den zuge-
10 ordneten Rechnern 7) bzw. auf den Devices 8 ablaufen. Die Cloud-Compute-Dienste 21 werden im Rahmen des Systems lediglich optional für alle Rechenoperationen herangezogen, die nicht auf den Modalitäten 3 oder den Devices 8 selbst vorgenommen werden. Letzteres betrifft insbesondere die rechen-
15 leistungsintensive Aufbereitung von 3D-Bilddaten (Volumendaten) der jeweiligen Modalität 3 für die Speicherung (Preprocessing) und/oder die Ableitung von gerenderten Ansichten V (Szenen oder Scene-Graphs) für die zweidimensionale Visualisierung (Bildsynthese, Volume Rendering) solcher Volumenda-
20 ten.

Die im Rahmen der Einrichtung 2 zu verwaltenden und zu bearbeitenden Daten umfassen gemäß FIG 2 Patientendatensätze P, Aufgaben W, Kontextdatensätze C und Bilddatensätze B, wobei
25 diese Daten im Einzelnen jeweils die vorstehend näher erläuterten Inhalte haben. Bilddatensätze B können hierbei medizinische Bilddaten oder Volumendaten oder eine sonstige Bild-, Ton-, Graphik- oder Textinformation (Photos, Videos, Dokumente) enthalten.

30

Gemäß FIG 2 ist für jede dieser verschiedenen Datenarten (Ressourcen) innerhalb des Cloud Storage 19 jeweils ein separater Speicherbereich angelegt, der nachfolgend als (Ressource-)Hub bezeichnet ist. Im Einzelnen sind in dem Cloud Storage 19 somit
35

- ein (Patient-)Hub 22 zur Speicherung der Patientendatensätze P,

- ein (Worklist-)Hub 23 zur Speicherung der Aufgaben W,
 - ein (Context-)Hub 24 zur Speicherung der Kontextdatensätze C sowie
 - ein (Image-)Hub 25 zur Speicherung der Bilddatensätze B
- 5 angelegt.

Jeder der Hubs 22-24 enthält jeweils einen Tabellenspeicher (nachfolgend als Table Storage 26 bezeichnet).

- 10 Im Rahmen des Systems 1 wird für jeden gespeicherten Datensatz in dem Table Storage 26 des jeweils zugeordneten Hubs 22-24 ein Tabelleneintrag angelegt, der nachfolgend als Tweet 27 bezeichnet ist.
- 15 Bei Datensätzen, die ihrer Natur nach aus vergleichsweise wenigen, diskreten Angaben bestehen, ist dabei der gesamte Datensatz als Tweet 27 im Table Storage 26 des jeweiligen Hubs 22-24 gespeichert. Dies betrifft im Beispiel gemäß FIG 2 die Patientendatensätze P, die Aufgaben W, sowie die Kontextdatensätze C. Anstelle des Table Storage 26 kann der Worklist-Hub 23 zur Speicherung der Aufgaben W nach einem First-in-first-out-Prinzip allerdings auch einem sogenannten Queue-Storage aufweisen.
- 20 Zur Speicherung der Bilddatensätze B, die in der Regel im Wesentlichen aus einem größeren und meist nicht-alphanumerischen Datenblock gebildet sind, ist im Image Hub 25 anstelle eines Table-Storage 26 ein sogenannter Blob-Storage 28 vorgesehen. Zu den in dem Image Hub 25 gespeicherten Bilddatensätzen B ist in dem Kontext Hub 24 jeweils ein Kontextdatensatz C in Form eines Tweets 27 hinterlegt. Dieser Tweet 27 enthält zumindest einen URI 29 (Uniform Ressource Identifier), der den Speicherort des Bilddatensatzes B im Blob-Storage 28 des Image Hubs 25 bezeichnet.
- 25 Für jeden der Hubs 22-25 wird im App Store 20 eine Applikation 31-34 zur Verfügung gestellt, die jeweils selektiv zur Darstellung und zur Bearbeitung der in dem jeweils zugeordne-
- 30
- 35

ten Hub 22-25 gespeicherten Datensätze dient. Jede der Applikationen 31-34 repräsentiert somit den zugeordneten Hub 22-25 auf der Ebene des Devices 8. Die Applikationen 31-34 sind daher - in Anlehnung an die Ressource-Hubs - auch als „Applications-Hubs“ bezeichnet.

Im Einzelnen umfassen die im App Store 20 zur Verfügung gestellten Applikationen somit

- 10 - eine Applikation 31 zur Verwaltung und Bearbeitung der in dem Patient-Hub 22 gespeicherten Patientendatensätze P,
- eine Applikation 32 zur Verwaltung und Bearbeitung von Aufgaben W,
- 15 - eine Applikation 33 zur Verwaltung und Bearbeitung von Kontextdatensätzen C, sowie
- eine Applikation 34 zur Anzeige und Bearbeitung der in dem Image-Hub 25 gespeicherten Bilddatensätze B.

20 Die Applikationen 31-34 können aus dem App Store 20 auf jedes der Devices 8 heruntergeladen und dort zur Ausführung gebracht werden. Sofern das System 1 - wie in FIG 1 dargestellt - unterschiedliche Arten von Devices (z.B. Personal-Computer 9, Tablet-Computer 11 und/oder Smartphones 12) umfasst, werden in dem App Store 20 jeweils verschiedene Versionen der

25 Applikationen 31-34 zur Verfügung gestellt, die zum Ablauf auf dem jeweiligen Device 8 angepasst sind.

Beispielsweise werden die Applikationen 31-34 zum Ablauf auf

30 dem Tablet-Computer 11 und dem Smartphone 12 in Form einer an den jeweiligen Gerätetyp angepassten App zur Verfügung gestellt, während die für den Personal-Computer 9 zur Verfügung gestellte Versionen der Applikationen 31-34 für den Ablauf in einem Internet-Browser konzipiert sind.

35

Im Vergleich zu einer herkömmlichen IT-Struktur für die Verwaltung von medizinischen Daten übernehmen der Patient Hub 22 und die zugeordnete Applikation 31 aus Sicht des Nutzers etwa

die Funktion eines herkömmlichen Krankenhaus-Informationssystem (HIS). Insbesondere sind die gesamten personenbezogenen (und datenschutzrelevanten) Patientendaten in dem Patient-Hub 22 aufgenommen, der somit für den einzelnen Patienten eine elektronische Patientenakte abbildet. Die übrigen Hubs 23-25 enthalten vorzugsweise keine Information, die aus sich heraus einem bestimmten Patienten zuordenbar wäre (Privacy Information). Der Worklist-Hub 23 und die zugeordnete Applikation 32 stellen sich dem Nutzer im Wesentlichen als Entsprechung zu einem herkömmlichen Radiologie-Informationssystem (RIS) dar, indem sie die innerhalb der Einrichtung 1 zur Durchführung anstehenden Aufgaben auflisten. Der Context-Hub 24 und die zugeordnete Applikation 33 übernehmen aus Nutzersicht etwa die Funktion eines herkömmlichen PACS, indem dort die zur Auffindung und Verarbeitung von Bilddatensätzen B erforderliche Information hinterlegt und recherchierbar ist. Der Image-Hub 25 und die zugeordnete Applikation 34 übernehmen schließlich etwa die Funktion eines herkömmlichen AV-Systems.

Obwohl die in den Hubs 22-25 und den zugeordneten Applikationen 31-34 enthalten Funktionen hochgradig miteinander verknüpft sind, ist jeder der Hubs 22-25 zusammen mit der jeweils zugeordneten Applikation 31-34 auch unabhängig von den jeweils anderen Hubs 22-25 und deren Applikationen 31-34 betreibbar. So kann beispielsweise auf einen in dem Image-Hub 25 enthaltenen Bilddatensatz B auch ohne den zugehörigen Kontextdatensatz C des Context-Hubs 24 zugegriffen werden, wenn der dem Image-Hub 25 zugeordneten Application 34 der URI 29 des anzuzeigenden Bilddatensatzes B auf andere Weise verfügbar gemacht wird, z.B. per E-Mail, SMS oder Twitter. Ferner können die Patientendaten P auch außerhalb der Public Cloud 13 vorgehalten werden, da der Patient-Hub 22 für die Funktion der übrigen Hubs 23-25 und der zugeordneten Applikationen 32-34 nicht erforderlich ist.

Ein Kernelement der dem Patienten-Hub 22 zugeordneten Applikation 31 ist eine sogenannte Browsing-Funktion, die den Tab-

le Storage 26 des Patient-Hubs 22 nach Tweets 27 durchsucht, die einem bestimmten, von einem Nutzer vorgebbaren Suchbegriff entsprechen. Von dem Patient-Hub 22 wird hierbei auf eine Suchanfrage Q (FIG 2) der Applikation 31 hin eine nachfolgend als Feed 35 bezeichnete Ergebnisliste, die die der Suchanfrage Q entsprechenden Tweets 27 enthält, an die Applikation 31 zurückgeliefert. Die Applikation 31 zeigt den erhaltenen Feed 35 auf dem Device 8 auf einer in FIG 3 exemplarisch dargestellten Seite 36 einer graphischen Benutzeroberfläche 37 an.

In der exemplarischen Ausführung gemäß FIG 3 weist die Seite 36 im Wesentlichen vier Anzeigebereiche unterschiedlicher Funktion auf, nämlich eine Kopfzeile 40, ein Suchfeld 41, ein Anzeigefeld 42 und eine Befehlsleiste 43. In der Kopfzeile 40 ist der Hub 22-24 namentlich benannt, dem die Applikation 31 zugeordnet ist. Im Beispiel der Applikation 31 ist die Kopfzeile 40 somit z.B. mit dem Wort „Patient Hub“ beschriftet. Des Weiteren ist in der Kopfzeile 40 vorzugsweise die Einrichtung 2 benannt, der das System 1 zugeordnet ist.

Das Suchfeld 41 enthält eine Eingabezeile 44, über die ein alphanumerischer Suchbegriff eingegeben werden kann, sowie eine Schaltfläche 45, durch deren Betätigung von einem Nutzer eine Suchanfrage Q auf Basis des zuvor in der Eingabezeile 44 eingegebenen Suchbegriffs gestartet werden kann.

Zusätzlich oder alternativ enthält das Suchfeld 41 ein Vorauswahlfeld 46, über das ein oder mehrere voreingestellte Suchbegriffe durch Antippen (im Falle eines berührungssensitiven Bildschirms) oder Anklicken mit einem gegebenenfalls vorhandenen Zeigergerät ausgewählt werden können.

Bei den über das Vorauswahlfeld 46 generierbaren Suchbegriffen handelt es sich insbesondere um häufig verwendete Suchschemata. Beispielsweise ermöglicht das Vorauswahlfeld 46 im Falle der mit dem Patient-Hub 22 korrespondierenden Applikation 31 eine Vorauswahl dahingehend, ob

- alle Tweets 27 („all“) oder
- nur solche Tweets 27 zu Patienten, für die Untersuchungen angesetzt sind („scheduled“) oder
- 5 - nur solche Tweets 27 zu Patienten, bei denen die zurückliegenden Leistung bereits abgerechnet wurden („billed“)

als Feed 35 zurückgeliefert werden sollen.

- 10 Optional ist vorgesehen, dass bei Generierung eines Suchbegriffs über das Vorauswahlfeld 26 eine alphanumerische Entsprechung dieses Suchbegriffs automatisch in der Eingabezeile 44 angezeigt wird.
- 15 Im Anzeigefeld 42 wird der aufgrund einer Suchabfrage Q zurückgelieferte Feed 35 angezeigt. Das Anzeigefeld 42 enthält somit in Form einer Liste diejenigen Tweets 27, die aufgrund der Suchanfrage Q der Applikation 31 in dem Patient-Hub 22 gefunden wurden. Zu jedem Tweet 27 werden beispielsweise ein
- 20 verkleinertes Bild 47 und/oder ein Schlagwort 48 angezeigt. So werden bei der dem Patient-Hub 22 zugeordneten Applikation 31 zu jedem angezeigten Tweet 27 vorzugsweise ein verkleinertes Bild des Patienten sowie der Name des Patienten und optional eine Patientenidentifikationsnummer angezeigt. Die Suchanfrage
- 25 Q, und der aufgrund dieser zurückgelieferte Feed 35 können von einem Nutzer interaktiv geändert werden.

- Die Suche wird basierend auf einer Syntax gemäß REST bzw. OData (Open Data Protocol Standard) definiert. Sie ist
- 30 grundsätzlich ähnlich zu einer WHERE-Klausel nach dem SQL-Standard, ist jedoch im Gegensatz zu der letzteren nicht von einer statischen Re-Indexierung abhängig. Insbesondere benötigen die Tabellen eines Hubs keine Index-Elemente für die Suche, weshalb auch keine Re-Indexierung erforderlich ist,
- 35 und entsprechend auch keine eigenen Daten-Server oder Datenbank-Server vorgehalten werden müssen.

Durch einfaches Antippen oder Anklicken eines im Anzeigefeld 42 angezeigten Tweets 27 kann dieser Tweet 27 von einem Nutzer ausgewählt werden.

5 Die Befehlsleiste 43 enthält eine Anzahl von Schaltflächen 49 und 50, durch deren Betätigung von einem Nutzer elementare Operationen zur Steuerung der Benutzeroberfläche 37 und zum Verwalten und Bearbeiten des angezeigten Feeds 35 vorgenommen werden können.

10

Die Applikation 31 ist in der in FIG 3 dargestellten Version für die Darstellung auf dem Smartphone 12 optimiert. Dabei wird auf dem Smartphone 12 zu jedem Zeitpunkt stets nur eine einzelne Seite 36 mit einem einzelnen Feed 35 angezeigt. Al-
15 lerdings können mehrere Instanzen der Seite 36 oder mehrere Instanzen der gesamten Applikation 31 parallel zueinander generiert werden, wobei diese Instanzen im Hintergrund der Anzeige nebenläufig auf dem Smartphone 12 ausgeführt werden. Somit können insbesondere mehrere Suchanfragen Q parallel zu-
20 einander abgearbeitet werden. Die Applikation 31 oder gegebenenfalls mehrere Instanzen der Applikation 31 laufen hierbei vorzugweise in einem generischen Rahmenmodul 51 (oder Container) ab. Dieses Rahmenmodul 51 kann beispielsweise als separate Applikation oder als Plug-in zu einem Web-Browser implementiert sein. Alternativ kann die Applikation 31 jedoch auch
25 zum unmittelbaren Ablauf in einem Web-Browser ausgebildet sein, z.B. als HTML5-Applikation.

Die Schaltflächen 50 der Befehlsleiste 43 ermöglichen hierbei
30 einen Wechsel zwischen den laufenden Instanzen der Applikation 31, indem bei Betätigung einer der Schaltflächen 50 die angezeigte Seite 36 der Benutzeroberfläche 37 ausgeblendet und die entsprechende Seite 36 einer anderen Instanz der Applikation 31 angezeigt wird.

35

Die weiteren Schaltflächen 49 der Befehlszeile 43 sind insbesondere folgenden Funktionen zugeordnet:

- „Create“: Auf Betätigung der entsprechenden Schaltfläche 49 hin werden eine neue Seite 36 der Benutzeroberfläche 37, ggf. zusammen mit einer neuen Instanz der Applikation 31 erzeugt.
- 5 - „Delete“: Auf Betätigung der entsprechenden Schaltfläche 49 hin werden die angezeigte Seite 36 der Benutzeroberfläche 37, ggf. zusammen mit der zugehörigen Instanz der Applikation 31 beendet,
- 10 - „Read“: Auf Betätigung der entsprechenden Schaltfläche 49 hin wird eine - in FIG 4 schematisch dargestellte - weitere Seite 52 der Benutzeroberfläche 37 geöffnet, in der der Dateninhalt des ausgewählten Tweets 27 zur bloßen Ansicht oder zur Veränderung angezeigt wird.

15 Die Seite 52 umfasst gemäß FIG 4 ebenfalls vier Bereiche, nämlich eine Kopfzeile 60, ein Überschriftenfeld 61, ein Anzeigefeld 62 sowie eine Befehlsleiste 63.

20 Die Kopfzeile 60 weist hierbei - analog zu der Kopfzeile 40 der Seite 36 - den zugeordneten Hub 22-24 und die Einrichtung 2 aus. Das Überschriftenfeld 61 kennzeichnet die Art des angezeigten Datensatzes und enthält somit im Falle der Applikation 31 beispielsweise das Wort „Patient“ (oder „Patient Entity“).

25 In dem Anzeigefeld 62 wird der Inhalt des ausgewählten Tweets 27 - im Falle der Applikation 31 somit der Inhalt des (Patienten-)Datensatzes - dargestellt. In diesem Fall werden in dem Anzeigefeld 62 insbesondere die Feldnamen 67 des Patienten-Datensatzes (zum Beispiel „Vorname“, „Nachname“, „Patienten-ID“, „Alter“, ...) zusammen mit dem jeweiligen Feldinhalt 68 (zum Beispiel also dem konkreten Vornamen, Nachnamen, der konkreten Patienten-ID bzw. dem konkreten Alter des Patienten dargestellt).

35 Die Befehlszeile 63 enthält Schaltflächen 64, denen insbesondere folgende Funktionen zugewiesen sind:

- „Create“: Auf die Betätigung der entsprechenden Schaltfläche 64 hin wird eine neue Instanz der Seite 52 zur Anlage eines neuen Tweets 27 und somit eines neuen (Patienten-)Datensatzes P erzeugt.
- 5 - „Update“: Auf die Betätigung der entsprechenden Schaltfläche 64 hin werden die - gegebenenfalls geänderten - Feldinhalte 68 gespeichert, indem die Applikation 31 den angezeigten Tweet 27 unmittelbar im Table Storage 26 des Patient-Hubs 22 mit den geänderten Angaben überschreibt.
- 10 - „Delete“: Auf die Betätigung der entsprechenden Schaltfläche 64 hin wird der angezeigte Tweet 27 durch die Applikation 31 unmittelbar im Table Storage 26 des Patient-Hubs 22 gelöscht; die angezeigte Seite 52 wird geschlossen.

15

Darüber hinaus enthält die Befehlszeile 63 eine Schaltfläche 65, auf deren Betätigung hin die Applikation 31 die angezeigte Seite 52 ohne Veränderung des Tweets 27 schließt und die zugrundeliegende Seite 36 erneut anzeigt.

20

Die Befehlszeilen 43 und 63 können weitere Schaltflächen für Befehle aufweisen. Ebenso können auch eine oder mehrere der anhand der Schaltflächen 49, 50, 64 und 65 beschriebenen Funktionen einem der gegebenenfalls vorhandenen elektromechanischen Tasten des Smartphones 12 zugeordnet sein. Die jeweilige Schaltfläche ist in diesem Fall vorzugsweise aus der Befehlszeile 43 entfernt.

25

Die Applikationen 32 und 33 haben jeweils eine der Applikation 31 entsprechende Browsing-Funktion. Insbesondere weist auch die jeweilige Benutzeroberfläche 37 der Applikationen 32 und 33 jeweils die anhand der FIG 3 und 4 beschriebenen Seiten 36 und 52 auf.

30

35 Allerdings greifen die Applikationen 32 und 33 auf den Worklist-Hub 23 bzw. auf den Context-Hub 24 zu, und sind entsprechend angepasst. So enthalten insbesondere

- die im Rahmen der Applikation 32 angezeigten Tweets 27 Angaben zu einer bestimmten Aufgabe W (beispielsweise Angaben zu dem Einstellungsdatum der Aufgabe, der mit der Aufgabe betreuten Abteilung, sowie Angaben zu der Aufgabe selbst und dem Taskflow, in den diese Aufgabe gegebenenfalls eingebunden ist), und
- die im Rahmen der Applikation 33 angezeigten Tweets 27 den Inhalt jeweils eines Kontext-Datensatzes C, insbesondere Angaben zum Dokumententyp, Speicherort, Datenformat sowie optional Aufnahmeparameter zu einem Bilddatensatz B.

Die Applikation 34 dient dagegen zur Anzeige („Viewing“) des nicht-alphanumerischen Inhalts eines im Image-Hub 25 gespeicherten Bilddatensatzes B. Hierfür umfasst die graphische Benutzeroberfläche 37 der Applikation 34 eine in FIG 5 dargestellte Seite 69. Ebenso wie die Seite 52 enthält auch die Seite 69 eine Kopfzeile 70, ein Überschriftenfeld 71, ein Anzeigefeld 72 und eine Befehlszeile 73. In dem Überschriftenfeld 71 ist hierbei beispielsweise die Art des angezeigten Bilddatensatzes B spezifiziert (z.B. als „2D-Viewing“, 3D-Viewing“, „Media-Viewing“, „Document-Viewing“, etc.). Im Anzeigefeld 72 wird der visuelle Inhalt des Bilddatensatzes B, also beispielsweise das Bild, Volumen, Video oder Textdokument angezeigt. Die Befehlszeile 73 enthält neben der Schaltfläche 65 weitere Schaltflächen 74 zur Beeinflussung der Anzeige, beispielsweise zum Vergrößern oder Verkleinern des im Anzeigefeld 72 angezeigten Dateninhalts oder um Verschieben eines angezeigten Bildausschnitts. Außerdem kann eine Schaltfläche 74 zum Löschen eines Bilddatensatzes B vorgesehen sein.

Anstelle der gemeinsamen Seite 69 zur Anzeige unterschiedlicher Datentypen kann die Applikation 33 auch verschiedene Viewingseiten umfassen, die jeweils zur Anzeige eines bestimmten Dokumententyps (Bild, Volumen, Video, Textdokument, etc.) eingerichtet sind.

Bezüglich der Anzeige von Volumendaten (3D-Viewing) ist die Applikation 34 in zwei Betriebsmodi betreibbar, nämlich in

- 5 - einem „Device-Only-Modus“, in dem die für die Anzeige der Volumendaten auf dem zweidimensionalen Bildschirm erforderlichen Rechenschritte, insbesondere die Anferti-
gung von gerenderten Ansichten V, auf dem Device 8 selbst vorgenommen werden, sowie
- 10 - einem „Cloud-Service-Modus“, in dem die Applikation 34 für diese Rechenoperationen durch eine entsprechende Anfrage R (FIG 2) Cloud-Compute-Dienste 21 anfordert.

Im Device-Only-Modus greift die Applikation 34 unmittelbar auf die im Blob-Storage 28 des Image-Hubs 25 gespeicherten
15 Bilddatensätze B zu. Im Cloud-Service-Modus werden die Bilddaten B durch Cloud-Compute-Dienste 21 bearbeitet, wobei lediglich die resultierenden zweidimensionalen Ansichten V von der Cloud 13 an die Applikation 34 geliefert werden. Aufgrund entsprechender Voreinstellung arbeitet die Applikation 34 von
20 Haus aus im Cloud-Service-Modus und schaltet nur dann in den Device-Only-Modus, wenn die Netzverbindung mit der Public Cloud 13 eine vorgegebene Mindestqualität unterschreitet.

Bevorzugt enthält die Applikation 34 zusätzlich Funktionen
25 zur Bearbeitung der Bilddaten, insbesondere zur Vermessung und/oder Segmentierung von Bildstrukturen und/oder zur Einbringung von Markierungen und/oder Annotationen. Zur Unterstützung der medizinischen Befundung der angezeigten Bilddaten enthält die Applikation 34 ferner eine Funktion zur Er-
30 stellung eines sogenannten Keyimage K (FIG 2). Wie verstehend erwähnt, handelt es sich hierbei um einen Schnappschuss, also eine Kopie der Bildinformation des angezeigten Bildausschnitts. Die Applikation 34 ist hierbei dazu eingerichtet, erstellte Keyimages K auf entsprechenden Nutzerbefehl direkt
35 als Bilddatensatz B in dem Blob-Speicher 28 des Image-Hubs 25 zu hinterlegen.

Die Applikationen 32-34 oder gegebenenfalls mehrere Instanzen dieser Applikationen 32-34 laufen ebenfalls bevorzugt in dem Rahmenmodul 51. Vorzugsweise ist hierbei – wie schematisch in FIG 2 angedeutet – ein gemeinsames Rahmenmodul 51 für alle Applikationen 31-34 vorgesehen. Alternativ kann aber auch jeder Applikation 31-34 ein eigenes Rahmenmodul zugeordnet sein.

Die Kommunikation zwischen jedem der Hubs 22-25 und der jeweils zugewiesenen Applikation 31-34 ist so gestaltet, dass sie mit dem im Internet-Datenverkehr gebräuchlichen REST(Representational State Transfer)-Prinzip kompatibel ist. Insbesondere genügen die zwischen den Hubs 22-25 und der jeweiligen Applikation 31-34 ausgetauschten Daten den Standards XML, JSON und ATOM. Bilddaten B sind im Image-Hub 25 insbesondere im JPEG-Format hinterlegt, Volumendaten vorzugsweise im DICOM-Format. Die gegebenenfalls von der Public-Cloud 13 an die Applikation 34 gelieferten Ansichten V (Scene-Graphs) von Volumendaten werden vorzugsweise im JPEG-Format erstellt.

Für den unmittelbaren Zugriff auf die in dem jeweiligen Hub 22-25 gespeicherten Daten nutzt jede Applikation 31-34 einen mehrfach instantiierbaren Cloud-Treiber, der in Form einer Anwender-Programmier-Schnittstelle (API) vorliegt.

Jede der Applikationen 31-33 ist dazu eingerichtet, Feeds auf Anweisung des Nutzers aus dem zugehörigen Hub 22-24 in einen lokalen Speicher des Device 8 zu externalisieren, so dass dieser Feed 35 und die darin enthaltenen Tweets 27 durch die Applikation 31-33 auch ohne Netzanbindung mit der Public Cloud 13 angezeigt und/oder bearbeitet werden können. Weiterhin ist jede Applikation 31-33 dazu eingerichtet, auf entsprechenden Befehl des Nutzers externalisierte Feeds 35 oder Tweets 27 an eine andere Instanz derselben oder einer anderen Applikation 31-34 zu übermitteln. Für die geräteinterne Übermittlung der Feeds 35 oder Tweets 27 ist im Speicher des Device 8 ein als „Exchangeboard“ bezeichneter lokaler Speicherbereich eingerichtet, auf den alle Instanzen der Applikatio-

nen 31-34 gemeinsam zugreifen. Zum Austausch über das Exchangeboard werden die auszutauschenden Feeds 35 oder Tweets 27 in XML/JSON-Files serialisiert, die im lokalen Speicher des Devices 8 abgelegt und anschließend wieder deserialisiert werden. Zum Austausch von Feeds 35 und Tweets 27 zwischen verschiedenen Devices 8 weisen die Applikationen 31-34 bevorzugt Schnittstellen zu Twitter oder einem E-Mail-Programm auf.

FIG 6 zeigt eine Variante der Applikationen 31-33, bei der die Benutzeroberfläche 37 für die Anzeige auf einem großformatigen Bildschirm, beispielsweise dem Bildschirm 10 des Personal-Computers 9 oder dem Bildschirm des Tablet-Computers 11 optimiert ist. Die Benutzeroberfläche 37 gemäß FIG 6 unterscheidet sich von der in den FIG 3 und 4 gezeigten Benutzeroberfläche 37 im Wesentlichen dadurch, dass mehrere Instanzen der Applikationen 31-33 und die entsprechenden Seiten 36 und/oder 52 der Benutzeroberfläche 37 nebeneinander (und somit gleichzeitig) sichtbar sind.

In der Darstellung gemäß FIG 6 weist die Benutzeroberfläche 37 weiterhin zusätzliche Navigations-Schaltflächen 50 auf, durch deren Betätigung wiederum zwischen den angezeigten Seiten 36 und/oder 52 geblättert werden kann.

Außerdem weist die Benutzeroberfläche 37 zusätzlich eine Menüleiste 80 auf, durch deren Betätigung zwischen den verschiedenen Applikationen 31-34 gewechselt werden kann. Die Menüleiste 80 ist dabei insbesondere Bestandteil eines Rahmenmoduls 81 (oder Containers), in dem die bestehenden Instanzen der Applikationen 31-34 ablaufen.

FIG 7 zeigt eine zur Anzeige auf einem großformatigen Bildschirm optimierte Version der Seite 69 der Applikation 34, bei der das Anzeigefeld 72 nahezu die gesamte Bildschirmfläche ausfüllt.

FIG 8 zeigt eine Variante des Systems 1, bei der die Applikation 34 zum Anzeigen von 2D-Bilddaten und Volumendaten ein (2D-Viewing-)Modul 92 und ein hiervon unabhängiges (3D-Viewing-)Modul 93 enthält. Die Applikation 34 ist hierbei in einer eingeschränkten Version auch mit lediglich einem der Module 92 bzw. 93 lauffähig. Das 2D-Viewing-Modul 92 greift hierbei zum Herunterladen des anzuzeigenden Bilddatensatzes B stets unmittelbar auf den Blob-Storage 28 des Image-Hubs 25 zu. Das 3D-Viewing-Modul 93 greift entweder ebenfalls unmittelbar auf die im Image-Hub 25 gespeicherten Volumendaten zu oder nimmt zur Erstellung der Views V Cloud-Compute-Dienste 21 in Anspruch.

Um einen bestimmten Bilddatensatz B mittels der Applikation 34 anzuzeigen, wird zum Beispiel von der dem Kontext-Hub 24 zugeordneten Applikation 33 der zugehörige Tweet 27 ermittelt und über das Exchangeboard an die Applikation 34 übergeben. Anhand des in dem übergebenen Tweet 27 enthaltenen URI 29 lädt die Applikation 34 den Bilddatensatz B aus dem Image-Hub 25.

FIG 9 zeigt eine weitere Variante des Systems 1, bei der dem Image-Hub 25 anstelle der einzigen Applikation 34 eine 2D-Viewing-Applikation 94 und eine hiervon unabhängige 3D-Viewing-Applikation 95 zugeordnet sind.

Gemäß FIG 10 ist jede Modalität 3 des Systems 1 mit einem in dem Rechner 7 implementierten Speichertreiber 100 versehen. Der Speichertreiber 100, der in Form einer Anwender-Programmier-Schnittstelle (API) für die Steuer- und Auswertesoftware der Modalität 3 vorliegt, ist hierbei dazu eingerichtet, einen von der Modalität 3 erzeugten und an den Speichertreiber 100 übergebenen Bilddatensatz B unmittelbar (ohne Inanspruchnahme von Cloud-Compute-Diensten 21) in den Blob-Storage 28 des Image-Hubs 25 zu schreiben. Weiterhin legt der Speichertreiber 100 einen zugehörigen Kontextdatensatz C als Tweet 27 in dem Table Storage 26 des Context-Hubs 24 an.

Der Speichertreiber 100 wandelt die Bilddaten dabei vorzugsweise (jedenfalls sofern es sich bei den Bilddaten um 2D-Bilddaten handelt) in das JPEG-Format um, sofern die Modalität 3 die Bilddaten ursprünglich in einem anderen Datenformat erstellt. Im Falle von Volumendaten werden diese dagegen bevorzugt im DICOM-Format im Blob-Speicher 28 des Image-Hubs 25 abgelegt.

Ferner nimmt der Speichertreiber 100 optional weitere Datenaufbereitungsprozesse (Preprocessing) vor, bevor er den Bildsatz in dem Image-Hub 25 ablegt.

Die Erfindung wird an den vorstehend beschriebenen Ausführungsbeispielen besonders deutlich, ist gleichwohl auf diese aber nicht beschränkt. Vielmehr können weitere Ausführungsformen der Erfindung aus der vorliegenden Beschreibung und den Ansprüchen abgeleitet werden. Insbesondere können die anhand der Ausführungsbeispiele beschriebenen Einzelmerkmale der Erfindung auch in anderer Weise kombiniert werden, ohne von der Erfindung abzuweichen.

Als Anlagen beigefügt sind:

- Anlage 1: Ein Beispiel-Code für die Benutzeroberfläche des Rahmens 51 (App-Hub Container-GUI)
- Anlage 2: Ein Beispiel-Code für die Logik des Rahmens 51 (App-Hub Container Code)
- Anlage 3: Ein Beispiel-Code für den Speichertreiber 100 (Direct Storage Adapter)

Anlage 1: Ein Beispiel-Code für die Benutzeroberfläche des Rahmens 51 (App-Hub Container-GUI)

```

<phone:PhoneApplicationPage
5   x:Class="AzureTableStore_CRUD_Sample.PivotMainPage"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/...
       presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:src="clr-namespace:AzureTableStore_CRUD_Sample;...
10       assembly=AzureTableStore_CRUD_Sample"
   xmlns:phone="clrnamespace:Microsoft.Phone.Controls;...
       assembly=Microsoft.Phone"
   xmlns:shell="clr-namespace:Microsoft.Phone.Shell;...
       assembly=Microsoft.Phone"
15   xmlns:controls="clr-namespace:Microsoft.Phone.Controls;...
       assembly=Microsoft.Phone.Controls"
   xmlns:d="http://schemas.microsoft.com/expression/...
       blend/2008"
   xmlns:mc="http://schemas.openxmlformats.org/...
20       markup-compatibility/2006"
   FontFamily="{StaticResource PhoneFontFamilyNormal}"
   FontSize="{StaticResource PhoneFontSizeNormal}"
   Foreground="{StaticResource PhoneForegroundBrush}"
   SupportedOrientations="Portrait" Orientation="Portrait"
25   mc:Ignorable="d" d:DesignHeight="696" d:DesignWidth="480"
   shell:SystemTray.IsVisible="True">

<phone:PhoneApplicationPage.Resources>
   <Storyboard x:Name="ResetPageTransitionList"...
30       Completed="OnResetPageTransitionList_Completed">
       <DoubleAnimationUsingKeyFrames Storyboard.TargetName=...
           "ContentPanel" Storyboard.TargetProperty=...
           "(UIElement.Projection).(PlaneProjection.RotationY)">
35       <EasingDoubleKeyFrame KeyTime="00:00:00"
           Value="0" />
       </DoubleAnimationUsingKeyFrames>
       <DoubleAnimationUsingKeyFrames Storyboard.TargetName=...

```

```

        "ContentPanel" Storyboard.TargetProperty=...
        "(UIElement.Projection).(PlaneProjection.CenterOfRotationX)">
    5      <EasingDoubleKeyFrame KeyTime="00:00:00"...
          Value="0" />
    </DoubleAnimationUsingKeyFrames>
  </Storyboard>
  <Storyboard x:Name="PageTransitionList" Completed=...
    "OnPageTransitionList_Completed">
10    <DoubleAnimationUsingKeyFrames Storyboard.TargetName=...
      "ContentPanel" Storyboard.TargetProperty=...
      "(UIElement.Projection).(PlaneProjection.RotationY)">
    <EasingDoubleKeyFrame KeyTime="00:00:00" Value="0" />
15    <EasingDoubleKeyFrame KeyTime="00:00:00.3"...
      Value="90">
      <EasingDoubleKeyFrame.EasingFunction>
        <CircleEase EasingMode="EaseIn" />
      </EasingDoubleKeyFrame.EasingFunction>
20    </EasingDoubleKeyFrame>
    </DoubleAnimationUsingKeyFrames>
    <DoubleAnimationUsingKeyFrames Storyboard.TargetName=...
      "ContentPanel" Storyboard.TargetProperty=...
      "(UIElement.Projection).(PlaneProjection.CenterOfRotationX)">
25    <EasingDoubleKeyFrame KeyTime="00:00:00"
      Value="0" />
    <EasingDoubleKeyFrame KeyTime="00:00:00.3"
      Value="0">
    <EasingDoubleKeyFrame.EasingFunction>
      <CircleEase EasingMode="EaseIn" />
    </EasingDoubleKeyFrame.EasingFunction>
30    </EasingDoubleKeyFrame>
    </DoubleAnimationUsingKeyFrames>
35  </Storyboard>
</phone:PhoneApplicationPage.Resources>

<Grid x:Name="LayoutRoot" Background="Transparent">

```

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin=...
    "12,0,12,0">
    <Grid.Projection>
        <PlaneProjection />
5    </Grid.Projection>

    <controls:Pivot x:Name="MainPivot" Title="AppTitle" >
        <controls:Pivot.Items>
            </controls:Pivot.Items>
10    </controls:Pivot>

    <StackPanel Grid.Row="1" Margin="12,0,12,0"...
        Orientation="Vertical" VerticalAlignment=...
        "Bottom">
15    <TextBlock Height="Auto" HorizontalAlignment="Left"...
        Margin="15,2,0,0" Text="{Binding Message,...
        Mode=TwoWay}" TextWrapping="Wrap" />
        <ProgressBar Height="20" IsIndeterminate="True"...
        Visibility="{Binding IsWorking, Converter=...
20    {StaticResource VisibilityConverter}}" />
    </StackPanel>

    </Grid>
</Grid>
25

<phone:PhoneApplicationPage.ApplicationBar>
    <shell:ApplicationBar IsVisible="True" >
        <!-- will be set at runtime in view model -->
        <shell:ApplicationBar.MenuItems>
30    <!-- will be set at runtime in view model -->
        </shell:ApplicationBar.MenuItems>
        </shell:ApplicationBar>
    </phone:PhoneApplicationPage.ApplicationBar>

35 </phone:PhoneApplicationPage>
```

Anlage 2: Ein Beispiel-Code für die Logik des Rahmens 51
(App-Hub Container Code)

```
#define _VIEWING
5
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
10 using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
15 using System.Windows.Media.Animation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;
using Microsoft.Samples.WindowsPhoneCloud.Storage...
    Client.Credentials;
20 using Microsoft.Samples.WindowsPhoneCloud.StorageClient;
using System.Globalization;
using System.Windows.Controls.Primitives;
using System.Windows.Navigation;
using Microsoft.Phone.Shell;
25 using Microsoft.Samples.Data.Services.Client;
using Microsoft.Samples.Data;
using Microsoft.Samples.Data.Services.Common;

30 namespace AzureTableStore_CRUD_Sample
{
    public partial class PivotMainPage : PhoneApplicationPage,
        IMainPage
    {
35        //-----
        // VIEW GUI Page variables
        //-----
```

```
private string CurrentViewModel_StateStore =...
    "CurrentViewModel_Normal_App";
private string AppFeed_All_StateStore = "AppFeed_Apps";
private bool isNewPageInstance = false;
5 private int externalNavigateToCounter = 0;
private string storedRequester = "";
private AppHubMainInternalApps internalApps;

// constructor
10 public PivotMainPage()
{
    InitializeComponent();
    // -- 19 --
    CreateAppFeed();
15 }

public void CreateAppFeed()
{
20     IsNewPageInstance = true;
    if (App.AppInstances.Count == 0)
    {
        MessageBox.Show("CreateAppFeed: there are no...
            AppInstances available - there might be...
25     network problems ...? Stop the program!");
        return;
    }
    App.AppInstances[0].O_CurrentViewModel = null;

30     for (int i = 0; i < App.AppInstances.Count; i++)
    {
        App.CurrAppIndex = i;
        switch (App.AppInstances[i].M_Model)
        {
35     case "Patient":
        App.AppInstances[i].O_AppFeed_Pivot =...
            new AppFeed<Patient, PivotMainPage>(this, i);
        this.Loaded += new RoutedEventHandler((App.App...
```

```
Instances[i].O_AppFeed_Pivot as AppFeed<...
Patient, PivotMainPage>).OnPage_Loaded);
break;

5 case "Worklist":
    App.AppInstances[i].O_AppFeed_Pivot =...
        new AppFeed<Worklist, PivotMainPage>(this,...
            i);
    this.Loaded += new RoutedEventHandler((App.App...
10 Instances[i].O_AppFeed_Pivot as AppFeed<...
        Worklist, PivotMainPage>).OnPage_Loaded);
    break;

case "Context":
15 App.AppInstances[i].O_AppFeed_Pivot =...
        new AppFeed<Context, PivotMainPage>(this, i);
    this.Loaded += new RoutedEventHandler((App.App...
        Instances[i].O_AppFeed_Pivot as AppFeed<...
        Context, PivotMainPage>).OnPage_Loaded);
20 break;

case "Score":
    App.AppInstances[i].O_AppFeed_Pivot =...
        new AppFeed<Score, PivotMainPage>(this, i);
25 this.Loaded += new RoutedEventHandler((App.App...
        Instances[i].O_AppFeed_Pivot as AppFeed<...
        Score, PivotMainPage>).OnPage_Loaded);
    break;

30 case "Kalle":
    App.AppInstances[i].O_AppFeed_Pivot =...
        new AppFeed<Kalle, PivotMainPage>(this, i);
    this.Loaded += new RoutedEventHandler((App.App...
        Instances[i].O_AppFeed_Pivot as AppFeed<...
35 Kalle, PivotMainPage>).OnPage_Loaded);
    break;

case "Sampletabel":
```

```
App.AppInstances[i].O_AppFeed_Pivot =...
    new AppFeed<Sampletable, PivotMainPage>...
    (this, i);
this.Loaded += new RoutedEventHandler((App.App...
5     Instances[i].O_AppFeed_Pivot as AppFeed<...
    Sampletable, PivotMainPage>).OnPage_Loaded);
break;

case "Blob":
10     App.AppInstances[i].O_AppFeed_Pivot =...
        new AppFeed<Blob, PivotMainPage>(this, i);
this.Loaded += new RoutedEventHandler((App.App...
    Instances[i].O_AppFeed_Pivot as AppFeed<...
    Blob, PivotMainPage>).OnPage_Loaded);
15     break;

case "QueueMessage":
    App.AppInstances[i].O_AppFeed_Pivot =...
        new AppFeed<QueueMessage, PivotMainPage>...
20     (this, i);
this.Loaded += new RoutedEventHandler((App.App...
    Instances[i].O_AppFeed_Pivot as AppFeed<...
    QueueMessage, PivotMainPage>).OnPage_Loaded);
break;
25

case "File":
    App.AppInstances[i].O_AppFeed_Pivot =...
        new AppFeed<File, PivotMainPage>(this, i);
this.Loaded += new RoutedEventHandler((App.App...
30     Instances[i].O_AppFeed_Pivot as AppFeed<...
    File, PivotMainPage>).OnPage_Loaded);
break;

case "Twitter":
35     App.AppInstances[i].O_AppFeed_Pivot =...
        new AppFeed<Twitter, PivotMainPage>(this, i);
this.Loaded += new RoutedEventHandler((App.App...
    Instances[i].O_AppFeed_Pivot as AppFeed<...
```

```
        Twitter, PivotMainPage>).OnPage_Loaded);
        break;

        case "ApplHub":
5           App.AppInstances[i].O_AppFeed_Pivot =...
                new AppFeed<ApplHub, PivotMainPage>(this, i);
            this.Loaded += new RoutedEventHandler((App.App...
                Instances[i].O_AppFeed_Pivot as AppFeed<...
                ApplHub, PivotMainPage>).OnPage_Loaded);
10           break;

        case "ApplFeed":
            App.AppInstances[i].O_AppFeed_Pivot =...
                new AppFeed<ApplFeed, PivotMainPage>...
15           (this, i);
            this.Loaded += new RoutedEventHandler((App.App...
                Instances[i].O_AppFeed_Pivot as AppFeed<...
                ApplFeed, PivotMainPage>).OnPage_Loaded);
            break;
20

        default:
            MessageBox.Show("CreateAppFeed: wrong M_Model=...
                " + App.AppInstances[i].M_Model + " used...
                in AppConfig.xml, stop working!");
25           return;
        }
    }
    // -- 20 --
}
30

public void LoadAppFeedItems(bool createFlag)
{
    if (App.AppInstances.Count == 0)
35     {
        MessageBox.Show("AppConfig.xml not loaded -->...
            stop processing!!");
        return;
    }
}
```

```
    }

    this.MainPivot.Items.Clear();

5    int cnt = App.AppInstances.Count;
    for (int i = 0; i < cnt; i++)
    {
        App.CurrAppIndex = i;
        switch (App.AppInstances[i].M_Model)
10    {
        case "Patient":
            (App.AppInstances[i].O_AppFeed_Pivot as AppFeed<...
                Patient, PivotMainPage>).LoadItem(...
                createFlag, i);
15        break;

        case "Worklist":
            (App.AppInstances[i].O_AppFeed_Pivot as AppFeed<...
                Worklist, PivotMainPage>).LoadItem(...
20        createFlag, i);
            break;

        case "Context":
            (App.AppInstances[i].O_AppFeed_Pivot as AppFeed<...
25        Context, PivotMainPage>).LoadItem(...
            createFlag, i);
            break;

        case "Score":
30        (App.AppInstances[i].O_AppFeed_Pivot as AppFeed<...
            Score, PivotMainPage>).LoadItem(...
            createFlag, i);
            break;

35        case "Kalle":
            (App.AppInstances[i].O_AppFeed_Pivot as AppFeed<...
                Kalle, PivotMainPage>).LoadItem(...
                createFlag, i);
```

```
        break;

        case "Sampletable":
            (App.AppInstances[i].O_AppFeed_Pivot as AppFeed<...
5              Sampletable, PivotMainPage>).LoadItem(...
              createFlag, i);
            break;

        case "Blob":
10          (App.AppInstances[i].O_AppFeed_Pivot as AppFeed<...
              Blob, PivotMainPage>).LoadItem(...
              createFlag, i);
            break;

        case "QueueMessage":
15          (App.AppInstances[i].O_AppFeed_Pivot as AppFeed<...
              QueueMessage, PivotMainPage>).LoadItem(...
              createFlag, i);
            break;

20          case "File":
              (App.AppInstances[i].O_AppFeed_Pivot as AppFeed<...
                  File, PivotMainPage>).LoadItem(...
                  createFlag, i);
25              break;

        case "Twitter":
            (App.AppInstances[i].O_AppFeed_Pivot as AppFeed<...
30              Twitter, PivotMainPage>).LoadItem(...
              createFlag, i);
            break;

        case "ApplHub":
            (App.AppInstances[i].O_AppFeed_Pivot as AppFeed<...
35              ApplHub, PivotMainPage>).LoadItem(...
              createFlag, i);
            break;
```

```
        case "ApplFeed":
            (App.AppInstances[i].O_AppFeed_Pivot as AppFeed<...
                ApplFeed, PivotMainPage>).LoadItem(...
                    createFlag, i);
5            break;

        default:
            MessageBox.Show("LoadAppFeedItems: wrong M_Model=...
                " + App.AppInstances[i].M_Model + " used...
10            in AppConfig.xml, stop working!");
            return;
        }
    }
}

15 public bool IsNewPageInstance
    {
        get { return isNewPageInstance; }
        set { isNewPageInstance = value; }
20    }

    public void DoSetPageTransitionList()
    {
        // -- 17 --
25    PageTransitionList.Begin();
    }

    public void DoResetPageTransitionList()
    {
30    // -- 22, 23 --
        ResetPageTransitionList.Begin();
    }

    private void OnNavigatedToUs(int SelectedIndex)
35    {
        // -- 21 --
        if (App.PageTransitions.ToUpper() == "ON" &&...
            App.PageTransitions.ToUpper() != null)
```

```
        {
            DoResetPageTransitionList();
        }
        else
5         {
            this.RestorePage(0);
        }
    }

10    protected override void OnNavigatedTo(Navigation...
        EventArgs e)
    {
        if ((e.IsNavigationInitiator == false) &&...
            (e.NavigationMode == System.Windows.Naviga...
15         tion.NavigationMode.Back))
        {
            if (storedRequester == "ChooserAndLauncher")
            {
                externalNavigateToCounter++;
20                 if (externalNavigateToCounter < 3)
                {
                    return;
                }
            }
            else
25             {
                externalNavigateToCounter = 0;
                storedRequester = "";
            }
        }
30    }

    // -- 21 --
    base.OnNavigatedTo(e); // base.
    if (App.PageTransitions.ToUpper() == "ON" &&...
35         App.PageTransitions.ToUpper() != null)
    {
        DoResetPageTransitionList();
    }
```

```
        else
        {
            this.RestorePage(0);
        }
5    }

protected override void OnNavigatedFrom(NavigationEvent...
                                   Args e)
{
10    if (storedRequester == "ChooserAndLauncher")
    {
        return;
    }
    base.OnNavigatedFrom(e);
15    if (e.NavigationMode != System.Windows.Navi...
                                   gation.NavigationMode.Back)
    {
        this.SaveGuiState();
20    }
}

//-----
// VIEW GUI Page GoBack Handling
25 //-----
protected override void OnBackKeyPress(System.Component...
                                   Model.CancelEventArgs e)
{
    string req = GetApplicationState<string>(...
30         "CurrentRequest");
    string rsp = GetApplicationState<string>(...
                                   "CurrentResponse");
    if (((req == "Add") || (req == "Update")) && ((...
                                   rsp == "Save") || (rsp == "Cancel") || (...
35         rsp == "Delete")))
        return;
    CallPreviousAppPage(0);
    e.Cancel = true;
```

```
    }

    public void RestorePage(int AppIndex)
    {
5      // -- 25 --
      if (this.IsNewPageInstance)
      {
          if ((AppIndex < 0) || (App.AppInstances[App...
                          Index].O_CurrentViewModel == null))
10      {
          if (this.State.Count > 0)
          {
              this.RestoreGuiState();
          }
15      else
          {
              this.LoadAppFeedItems(true);
          }
      }
20  }
      this.IsNewPageInstance = false;

      (App.AppInstances[0].O_CurrentViewModel as ViewModelBase...
          ).OnNavigatedTo();
25  (App.AppInstances[0].O_CurrentViewModel as ViewModelBase...
          ).ViewUpdateApplicationBarButtons(...
              this.ApplicationBar, new[] { "logout" }...
          ); // this.

30  }

    //-----
    // VIEW GUI Call Hierarchical Details Page Handling
    //-----
35  public void CallDetailPage()
    {
        if (App.NavigationUri == "" || App.NavigationUri == null)
```

```
{
    (App.AppInstances[0].O_CurrentViewModel as ViewModelBase...
        ).ApplicationBarIconButton_...
        UPDATE_ENTITY_Click();
5    }
    else
    {
        int selectedIndex=(App.AppInstances[0].O_CurrentView...
            Model as ViewModelBase).GeneralSelected...
10        Index;
        CallNextAppPage(selectedIndex);
    }
}

15 //-----
// VIEW GUI App Navigation Logic (Hierarchical Next App-...
// Hub Page Handling)
//-----

20 public void CallNextAppPage(int SelectedIndex)
{
    storedRequester = "";
    if (SelectedIndex < -1000)
    {
25        storedRequester = "ChooserAndLauncher";
        externalNavigateToCounter = 0;
        switch (SelectedIndex)
        {
            case -1001: // send email
30                if (internalApps == null) internalApps =...
                    new AppHubMainInternalApps();
                    internalApps.takeSelectedItemSelect...
                        EmailAdrAndSendViaEmail();
                    break;
35                case -1002: // send sms
                    if (internalApps == null) internalApps =...
                        new AppHubMainInternalApps();
```

```
        internalApps.takeSelectedListItemSelect...
            PhoneNrAndSendViaSms();
        break;
    }
5
    return;
}

// the regular App-Feed processing comes here ...
10
int maxAppIndexAvail = App.AppsInstances.Count - 1;
App.PreviousApp = App.CurrentApp;
switch (App.AppInstances[0].V_NextAppNavUri)
{
15     case "Application 0": // ApplHub
        App.CurrentApp = 0;
        break;

        case "Application 1": // Twitter
20         App.CurrentApp = 1;
        break;

        case "Application 2": // Patient
25         App.CurrentApp = 2;
        break;

        case "Application 3": // Worklist
30         App.CurrentApp = 3;
        break;

        case "Application 4": // Context
35         App.CurrentApp = 4;
        break;

        case "Application 5": // Iso-FileViewer
            App.CurrentApp = 5;
            break;
```

```
        case "Application 6": // Azure-Blob-Viewer
            App.CurrentApp = 6;
            break;

5        case "Application 7": // Azure-Queue-Viewer
            App.CurrentApp = 7;
            break;

        case "Application 8": // ???
10        App.CurrentApp = 0;
            break;

        case "Application 9":
            NavigateToViewingApp();
15        return;

        default:
            if ((App.AppInstances[0].V_NextAppNavUri == "*")...
                && (App.CurrentApp == 0) && (...
20                SelectedIndex == -1))
                App.CurrentApp = 2;
            else
                App.CurrentApp = SelectedIndex + 1;
            break;
25    }
    CreateAppFeed();
    OnNavigatedToUs(SelectedIndex);
}

30 public void CallPreviousAppPage(int SelectedIndex)
    {
        int maxAppIndexAvail = App.AppsInstances.Count - 1;
        App.PreviousApp = App.CurrentApp;
        switch (App.CurrentApp)
35    {
        case 0: // ApplHub
            App.CurrentApp = 0;
            break;
```

```
case 1: // Twitter
    App.CurrentApp = 0;
    break;
5
case 2: // Patient
    App.CurrentApp = 0;
    break;
10
case 3: // Worklist --> goto Patient
    App.CurrentApp = 2;!!
    break;
case 4: // Context --> goto Worklist
15
    App.CurrentApp = 3;
    break;
case 5: // Iso-FileViewer
    App.CurrentApp = 0;
20
    break;
case 6: // Azure-Blob-Viewer
    App.CurrentApp = 0;!!
    break;
25
case 7: // Azure-Queue-Viewer
    App.CurrentApp = 0;
    break;
30
case 8: // ????
    App.CurrentApp = 0;
    break;
case 9: // Viewing-App
35
    App.CurrentApp = 0;
    break;
default:
```

```
        App.CurrentApp = 0;
        break;
    }
    CreateAppFeed();
5    OnNavigatedToUs(SelectedIndex);
}

private void NavigateToViewingApp()
{
10    #if _VIEWING
    try
    {
        string currentItemType = (App.App...
                                Instances[0].O_CurrentViewModel as
15                                ViewModelBase).GetItemTypeFrom...
                                SelectedIndex();
        if (currentItemType == "File")
        {
            var c = (App.AppInstances[0].O_CurrentView...
20                    Model as ViewModelBase).GetExternalEntity...
                    FromSelectedIndex<File>();
            if ((c as Context) != default(Context))
            {
                NavigationService.Navigate(App.PrepareNavigate...
25                ToViewingApps("/ViewsViewing/Viewing...
                Page.xaml", (Context)c, "");
            }
        }
        else if (currentItemType == "Blob")
30        {
            var c = (App.AppInstances[0].O_CurrentView...
                    Model as ViewModelBase).GetExternalEntity...
                    FromSelectedIndex<Blob>();
            if ((c as string) != "")
35            {
                NavigationService.Navigate(App.PrepareNavigate...
                ToViewingApps("/ViewsViewing/Viewing...
                Page.xaml", null, c as string));
            }
        }
    }
    catch { }
}
```

```
    }  
  }  
  else if (currentItemType == "Message")  
  {  
5      var c = (App.AppInstances[0].O_CurrentView...  
              Model as ViewModelBase).GetExternalEntity...  
              FromSelectedIndex<QueueMessage>();  
      if ((c as string) != "")  
      {  
10         NavigationService.Navigate(App.PrepareNavigate...  
            ToViewingApps("/ViewsViewing/Viewing...  
            Page.xaml", null, c as string));  
      }  
    }  
  else if (currentItemType == "Twitter")  
  {  
      var c = (App.AppInstances[0].O_CurrentView...  
              Model as ViewModelBase).GetExternalEntity...  
              FromSelectedIndex<Twitter>();  
20     if ((c as string) != "")  
      {  
        NavigationService.Navigate(App.PrepareNavigate...  
          ToViewingApps("/ViewsViewing/Viewing...  
          Page.xaml", null, c as string));  
25     }  
    }  
  else  
      NavigationService.Navigate(App.PrepareNavigate...  
          ToViewingApps("/ViewsViewing/Viewing...  
30     Page.xaml", (App.AppInstances[0].O_Current...  
          ViewModel as ListViewModel<Context>...  
          ).SelectedItem, ""));  
    }  
  catch  
35  {  
    // do nothing here  
  }  
#endif
```

```
    }
    private void NavigateToApp(int SelectedIndex)
    {
        if (App.DesignerMode == null)
5         App.SetDefaultAppSettings();
        if (App.DesignerMode == null || ...
            App.DesignerMode.ToUpper() == "NORMAL" || ...
            App.DesignerMode.ToUpper() == "")
        {
10         NavigationService.Navigate(new Uri("/Views/...
            NormalMainPage.xaml?selectedItem=" + ...
            SelectedIndex, UriKind.Relative));

            return;
        }
15     if (App.DesignerMode.ToUpper() == "PIVOT")
        {
            NavigationService.Navigate(new Uri("/Views/...
                PivotMainPage.xaml?selectedItem=" + ...
                SelectedIndex, UriKind.Relative));
20         return;
        }
        if (App.DesignerMode.ToUpper() == "PANORAMA")
        {
            NavigationService.Navigate(new Uri("/Views/...
25             PanoramaMainPage.xaml?selectedItem=" + ...
            SelectedIndex, UriKind.Relative));

            return;
        }
    }
30

    //-----
    // VIEW GUI save/restore state
    //-----

35     public void SaveGuiState()
    {
        SetApplicationState(AppFeed_All_StateStore, ...
            App.AppInstances);
```

```
    }

    // neu: from AppFeed.cs
    public void RestoreGuiState()
5      {
        App.AppInstances = GetApplicationState<List<Appl...
                                FeedData>>(AppFeed_All_StateStore);
      }

10     //-----
    // VIEW Application State Handling
    //-----

    public void SetApplicationState(string key, object...
15                                value)
    {
        if (PhoneApplicationService.Current.State.Contains...
                                Key(key))
        {
20            PhoneApplicationService.Current.State.Remove(key);
        }
        PhoneApplicationService.Current.State.Add(key, ...
                                value);
    }

25     public F GetApplicationState<F>(string key)
    {
        if (!PhoneApplicationService.Current.State.Contains...
                                Key(key))
30        {
            return default(F);
        }

        return (F)PhoneApplicationService.Current.State[key];
35    }

    public void RemoveApplicationState(string key)
    {
```

```
        If (PhoneApplicationService.Current.State.Contains...
            Key(key))
        {
            PhoneApplicationService.Current.State.Remove(key);
5         }
    }

    //-----
    // VIEW GUI Page Storyboard Transition Handling
10  //-----

    public void OnPageTransitionList_Completed(object...
        sender, EventArgs e)
    {
15    // -- 18 --
        this.CallDetailPage();
    }

    public void OnResetPageTransitionList_Completed(...
20        object sender, EventArgs e)
    {
        // -- 24 --
        this.RestorePage(0);
    }
25 }
}
```

Anlage 3: Ein Beispiel-Code für den Speichertreiber 100 (Direct Storage Adapter)

```
#define UPLOAD
5  #define _KeyImageWatermarkUsed
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Net;
10  using System.Windows;
    using System.Windows.Controls;
    using System.Windows.Documents;
    using System.Windows.Input;
    using System.Windows.Media;
15  using System.Windows.Media.Animation;
    using System.Windows.Shapes;
    using System.Windows.Navigation;
    using Microsoft.Phone.Controls;
    using Microsoft.Phone.Shell;
20  using System.Text.RegularExpressions;
    using System.Windows.Interop;
    using System.Windows.Media.Imaging;
    using System.Windows.Resources;
    using System.Diagnostics;
25  using System.Threading;
    using System.Xml;
    using System.Xml.Linq;
    using System.ComponentModel;
    using System.Globalization;
30
    #if UPLOAD
    using SLBlobUploader.Code;
    using SLBlobUploader.Code.Abstract;
    using SLBlobUploader.Code.Infrastructure;
35  using SLBlobUploader.Code.StorageClient;
    using System.IO;
    using System.IO.IsolatedStorage;
    using System.Collections;
```

```
using System.Windows.Threading;
#endif

namespace AzureTableStore_CRUD_Sample
5  {
    public class UploadKeyImage : INotifyPropertyChanged
    {
        // state indication to GUI progressbar (ViewModel)
        private string message;
10    private bool isWorking = false;

        // Uploader parameters
        private const long MaxFileSizeKb = Constants.MaxFileSizeKB;
        private const string UploadButtonText = "Upload";
15    private const string CancelButtonText = "Cancel";
        private const long BytesPerKb = 1024;
        private List<UserFile> files = null;
        private string sasUrl = string.Empty;
        private string timeOutSeconds = string.Empty;
20    private bool sasExpired = false;
        private DateTime operationStartTime;
        private UserFile userFile = null;

        // contextfolder object in use
25    Context selectedContextObject;

        // sync to main thread
        private Dispatcher Dispatcher;

30    // State
        private String State = "Idle";
        private string KeyImagefileNameWithoutExt = "";
        private string _TimeOutSeconds;
        private bool _SasExpired;
35    private string _SasUrl;
        IsolatedStorageFileStream myThumbnailFileStream;
        IsolatedStorageFileStream myKeyImageFileStream;
        private string IsoKeyImageFileName = "";
```

```
private string IsoKeyImageThumbnailFileName = "";
private string CloudPathName = "";

// owner
5 ViewingPage owner;

//-----
// Constructor
//-----
10
public UploadKeyImage()
{
}

15 public UploadKeyImage(ViewingPage parent)
{
owner = parent;
TextBox tb = new TextBox();
private void ClearNotificationMesssageArea()
20 {
this.Message = "";
this.IsWorking = false;
}

25 //-----
// 0) API-Call - SAVE screen captured "KeyImages" to...
cloud storage
//-----
public string DoKeyImaUpload(Context SelContextFolder...
30 Object, WriteableBitmap KeyImaBitmap,...
string mTimeOutSeconds,...
bool mSasExpired, string mSasUrl)
{
if (State == "Idle")
35 {
State = "Blocked part 1";
UploadDoNotification();
this.selectedContextObject = SelContextFolderObject;
```

```
        this._TimeOutSeconds = mTimeOutSeconds;
        this._SasExpired = mSasExpired;
        this._SasUrl = mSasUrl;
        this.IsoKeyImageFileName = "tempKeyIma.jpg";
5    this.IsoKeyImageThumbnailFileName =...
            "tempKeyImaThumbnail.jpg";
        saveKeyImagesInIsoStore(KeyImaBitmap);
        var myStore = IsolatedStorageFile.GetUserStore...
            ForApplication();
10    this.myThumbnailFileStream = myStore.OpenFile(IsoKeyImage...
            ThumbnailFileName, FileMode.Open,...
            FileAccess.ReadWrite);

        this.KeyImagefileNameWithoutExt = storeKeyImageThumbnailIn-
15    ContextFolderBlobStore(myThumbnailFileStream,
        mTimeOutSeconds, mSasExpired, mSasUrl);
        return ("Idle");
    }
    return State;
20 }

private void UploaderDoneNotification(string msg)
25 {
    if (State == "Blocked part 1")
    {
        this.myThumbnailFileStream.Close();
        var myStore = IsolatedStorageFile.GetUser...
30         StoreForApplication();

        this.myKeyImageFileStream = myStore.OpenFile(this.IsoKey...
            ImageFileName, FileMode.Open,...
            FileAccess.ReadWrite);
35    storeKeyImageInContextFolderBlobStore(this.KeyImagefile...
            NameWithoutExt, myKeyImageFileStream,...
            this._TimeOutSeconds, this._Sas...
            Expired, this._SasUrl);
```

```
State = "Blocked part 2";

#if early
    UploadDoneNotification(msg);    storeKeyImageInContextFolderT-
5  ableStore(this.selected..
                                ContextObject, this.CloudPathName,...
                                this.KeyImagefileNameWithoutExt);
                                // filenames of destination in cloud!
    this.myKeyImageFileStream.Close();
10  State = "Idle";

    /* &&&&&&&&& UploaderDoneNotification only called once!!!
    try to evaluate why and then reenable this path
    again!! &&&&&&*/
15  #else
    }

    else if (State == "Blocked part 2")
    {
20  doTableEntryAndUploadDoneNotification(msg);
    }
    else
    {
    MessageBox.Show("Uploaded KeyImage ended abnormal!");
25  }
    #endif
    }

private void TableEntryAndUploadDoneNotification(string msg)
30  {
    storeKeyImageInContextFolderTableStore(this.selected..
                                ContextObject, this.CloudPathName,...
                                this.KeyImagefileNameWithoutExt);
    this.myKeyImageFileStream.Close();
35  UploadDoneNotification(msg);
    State = "Idle";
    }

private delegate void doUploadDoneNotificationDelegate(...
```

```
        string msg);

private void doUploadDoneNotification(string msg)
{
5   bool isDispatcherThread = this.Dispatcher.CheckAccess();
   if (isDispatcherThread == true)
   {
       UploadDoneNotification(msg);
   }
10  else
   {
       this.Dispatcher.BeginInvoke(new doUploadDoneNotification...
                                   Delegate(UploadDoneNotification), ...
                                   msg);
15
   }
}

private delegate void doTableEntryAndUploadDone...
20      NotificationDelegate(string msg);

private void doTableEntryAndUploadDoneNotification(...
                                   string msg)
{
25  bool isDispatcherThread = this.Dispatcher.CheckAccess();
   if (isDispatcherThread == true)
   {
       TableEntryAndUploadDoneNotification(msg);
   }
30  else
   {
       this.Dispatcher.BeginInvoke(new doTableEntryAndUploadDone...
                                   NotificationDelegate(TableEntry...
                                   AndUploadDoneNotification), msg);
35  }
}

//-----
```

```
// 1) Save stored iso Thumbnail-image as jpeg in "uploads"...
        container of azure blob store
//-----
private string storeKeyImageThumbnailInContext...
5         FolderBlobStore(Stream Video_Stream,...
        string mTimeoutSeconds, bool mSas...
        Expired, string mSasUrl)

{
10  try
    {
        string pathName = "";
        DateTime CurrentTime = System.DateTime.Now;
        string fileNameWithoutExt = "keyIma_" +
15  String.Format("{0:s}", CurrentTime) + "_" + Guid.NewGuid();
        string fileNameAndExt = fileNameWithoutExt +...
            "_thumbnail.jpg";
        using (Stream stream = new MemoryStream())
        {
20  stream.Position = 0;
        UploadStream(pathName, fileNameAndExt, Video_Stream,...
            mTimeoutSeconds, mSasExpired, mSasUrl);
        }
        return fileNameWithoutExt;
25  }
    catch
    {
        string lblMessage = "KeyImage-Thumbnail cannot be saved,...
            lease timeout?";
30  UploaderDoneNotification(lblMessage);
        return (null);
    }
}

35  //-----
// 2) Save stored iso keyImage-image as jpeg in "uploads"...
        container of azure blob store
```

```
//-----

private void storeKeyImageInContextFolderBlobStore(...
    string fileNameWithoutExt, Stream key...
5      Ima_Stream, string mTimeOutSeconds,...
    bool mSasExpired, string mSasUrl)

{
    try
10  {
        string pathName = "";
        string fileName = fileNameWithoutExt + ".jpg
        using (Stream stream = new MemoryStream())
        {
15      stream.Position = 0;
        UploadStream(pathName, fileName, keyIma_Stream, mTimeOut...
            Seconds, mSasExpired, mSasUrl);
        }
    }
20 catch
    {
        string lblMessage = "KeyImage cannot be saved, lease...
            timeout?";
        UploaderDoneNotification(lblMessage);
25  }
    }

//-----
// 3) Save stored keyImage as tweet in "contectfolder"...
30      table of azure table store
//-----

private void storeKeyImageInContextFolderTableStore(...
    Context selectedContextObject,...
    string path, string filename)
35  {
    Context cn = (Context)selectedContextObject.DeepCopy();
    cn.PartitionKey = selectedContextObject.PartitionKey;
    cn.RowKey = System.Guid.NewGuid().ToString();
```

```

cn.ServerBaseUri = "http://singapore4.blob.core.windows.net";
if (path == "")
cn.DZInstDataRelPath = "/" + "uploads" + "/" + filename;
else
5 cn.DZInstDataRelPath = "/" + "uploads" + "/" + path + "/" +
filename;
cn.Thumb = cn.ServerBaseUri + cn.DZInstDataRelPath +
"_thumbnail.jpg";
cn.DZInstDataRelPath += ".jpg"; // trick 5
10 cn.DZMetaDataRelPath = "jpeg";
cn.DZMHDDDataRelPath = "None";
cn.ARegion = "KeyImage";
cn.StorageFormat = "JPEG";
cn.StorageHostVolumeSeries2dAnz = "1";
15 if (((App.AppInstances[0].O_CurrentViewModel as ListView...
Model<Context>).Driver.Item...
Type == "Twitter") || ((App.App...
Instances[0].O_CurrentViewModel as ...
ListViewModel<Context>).Driver.Item...
20 Type == "Unknown"))

{
return;
}
25 (App.AppInstances[0].O_CurrentViewModel as ListView-
Model<Context>).AddEntity(cn);
}

//-----
30 private void saveKeyImagesInIsoStore(WritableBitmap ...
KeyIma_bitmap)

{
try
35 {
#if _KeyImageWatermarkUsed
StreamResourceInfo streamInfo = Application.GetResource...
Stream(new Uri("/AzureTable...

```

```
Store_CRUD_Sample;component/images/...
KeyImageAsRedWhite.png", UriKind...
.Relative));
Stream EmbeddedResourceStream = streamInfo.Stream;
5  if (EmbeddedResourceStream != null)
    {
        BitmapImage bitMapImage = new BitmapImage();
        bitMapImage.SetSource(EmbeddedResourceStream);
        WriteableBitmap bitmap = new WriteableBitmap(bitMapImage);
10  KeyIma_bitmap = ApplyKeyImageWithWatermark(KeyIma_bitmap,...
        bitmap);

        bitmap.Clear();
        bitmap = null;
15  EmbeddedResourceStream.Close();
        EmbeddedResourceStream.Dispose();
        EmbeddedResourceStream = null;
        bitMapImage = null;
    }
20  #endif
    using (IsolatedStorageFile isolatedStore = IsolatedStorage...
        File.GetUserStoreForApplication())

25  {
    if (isolatedStore.FileExists(this.IsoKeyImageFileName))
    {
        isolatedStore.DeleteFile(this.IsoKeyImageFileName);
    }
30  if (isolatedStore.FileExists(this.IsoKeyImageThumbnail...
        FileName))
    {
        isolatedStore.DeleteFile(this.IsoKeyImageThumbnailFileName);
    }
35  using (MemoryStream streamKeyIma = new MemoryStream())
    {
        JPGUtil.EncodeJpg(KeyIma_bitmap, streamKeyIma);
        streamKeyIma.Position = 0;
```

```
using (IsolatedStorageFileStream stream = new Isolated...
        StorageFileStream(this.IsoKeyImageFile...
        Name, FileMode.Create, isolatedStore))

5  {
    streamKeyIma.Position = 0;
    streamKeyIma.CopyTo(stream);
    stream.Close();
  }
10 }
    using (MemoryStream streamThumbnail = new MemoryStream())
    {
        KeyIma_bitmap.SaveJpeg(streamThumbnail, 120, 120, 0, 80);
        streamThumbnail.Position = 0;
15  using (IsolatedStorageFileStream stream = new
        IsolatedStorageFileStream(this.IsoKeyImageThumbnailFile...
        Name, FileMode.Create, isolatedStore))
        {
            streamThumbnail.Position = 0;
20  streamThumbnail.CopyTo(stream);
            stream.Close();
        }
    }
    isolatedStore.Dispose();
25 }
}
catch (IsolatedStorageException e)
{
    string lblMessage = "KeyImage cannot be saved in ISO store...
30         temporarily: " + e.Message ;
    UploaderDoneNotification(lblMessage);
}
KeyIma_bitmap.Clear();
KeyIma_bitmap = null;
35 }

#if _KeyImageWatermarkUsed
public WriteableBitmap ApplyKeyImageWith...
```

```

        Watermark(WriteableBitmap input,...
        WriteableBitmap Watermark)
    {
        var w = Watermark.PixelWidth;
5    var h = Watermark.PixelHeight;
        var result = input.Clone();
        var position = new Rect( input.PixelWidth - w - 40,...
                                input.PixelHeight - h - 40, w, h);
        result.Blit(position, Watermark, new Rect(0, 0, w, h));
10    return result;
    }
#endif

private void old_saveStreamAsIsoFile(Stream str,...
15    string isoFileName)
    {
        try
        {
            using (IsolatedStorageFile isStore = IsolatedStorage...
20    File.GetUserStoreForApplication())
            {
                if (isStore.FileExists(isoFileName))
                {
                    isStore.DeleteFile(isoFileName);
25    }
            using (IsolatedStorageFileStream targetStream =...
                    isStore.OpenFile(isoFileName, File...
                    Mode.CreateNew, FileAccess.Write))
            {
30    byte[] readBuffer = new byte[4096];
                int bytesRead = -1;
                while ((bytesRead = str.Read(readBuffer, 0, read-
                    Buffer.Length)) > 0)
                {
35    targetStream.Write(readBuffer, 0, bytesRead);
                }
            }
        }
    }
```

```
    }
    catch
    {
        string lblMessage = "KeyImage, error saving " + isoFile...
5         Name + " in ISO store temporarily!";
        UploaderDoneNotification(lblMessage);
    }
}

10 private void saveStreamAsIsoFile(Stream incoming...
        Stream, string isoFileName)
    {
        try
        {
15         using (IsolatedStorageFile isolatedStore = IsolatedStorage...
                File.GetUserStoreForApplication())
            {
                // Delete file if it exists
                if (isolatedStore.FileExists(isoFileName))
20                 {
                    isolatedStore.DeleteFile(isoFileName);
                }
                using (var stream = new IsolatedStorageFileStream(iso...
                        FileName, FileMode.CreateNew,...
25                        isolatedStore))
                {
                    incommingStream.Position = 0;
                    incommingStream.CopyTo(stream);
                }
30             }
        }
        catch (IsolatedStorageException ise)
        {
            string lblMessage = "KeyImage, error saving " + isoFile...
35             Name + " in ISO store temporarily!";
            UploaderDoneNotification(lblMessage);
        }
    }
```

```
private void UploadStream(string pnam, string fnam,...
                        Stream stream, string mTimeOutSeconds,...
                        bool mSasExpired, string mSasUrl)
5  {
    string lblMessage = "undefined";
    bool initOk = initBlobUploader(mTimeOutSeconds, mSas...
                                Expired, mSasUrl);

    if (initOk)
10  {
    UploadStreamToCloudFile(pnam, fnam, stream);
    }
    else
    {
15  lblMessage = "File not saved, init error, lease timeout?";
    UploaderDoneNotification(lblMessage);
    }
    }

20  public bool initBlobUploader(string mTimeOutSeconds,...
                                bool mSasExpired, string mSasUrl)
    {
    timeOutSeconds = mTimeOutSeconds;
    sasExpired = mSasExpired;
25  sasUrl = mSasUrl;
    if (sasExpired)
    {
    if (this.userFile != null)
    {
30  this.userFile.CancelUpload();
    }
    return false;
    }
    return true;
35  }
```

```
private bool UploadStreamToCloudFile(string pnam,...
                                   string fnam, Stream stream)
{
    string lblMessage = "undefined";
5   if (this.files == null)
    {
        this.files = new List<UserFile>();
    }
    this.userFile = new UserFile();
10  this.userFile.PathName = pnam;
    this.userFile.FileName = fnam;
    this.userFile.FileStream = stream;
    this.userFile.StreamBytes = stream.Length;
    this.userFile.UIDispatcher = this.Dispatcher;
15  this.userFile.UploadContainerUrl = new Uri(this.sasUrl);
    this.userFile.UploadCompletedEvent += new EventHandler<UploadCompletedEventArgs>(this.OnUpload...
                                                Completed);
    if ((this.userFile.FileStream.Length / BytesPerKb)...
20      <= MaxFileSizeKb && this.user...
        File.FileStream.Length > 0)
    {
        this.files.Add(this.userFile);
    }
25  else
    {
        lblMessage = this.userFile.FileStream.Length > 0 ?
        string.Format("IllegalMaxFileSize: {0} in KB", MaxFile...
                      SizeKb / BytesPerKb) : string.Format(...
30      "IllegalMinFileSize: {0} in KB",...
        MaxFileSizeKb / 1024);
        MessageBox.Show("UploadStreamToCloudFile: " + lblMessage);
        return false;
    }
35  this.operationStartTime = DateTime.Now;
    if (this.files != null)
    {
        foreach (var file in this.files)
```

```
{
file.Upload(string.Empty);
}
}
5 else
{
lblMessage = "no files to transfer!";
MessageBox.Show("UploadStreamToCloudFile: " + lblMessage);
return false;
10 }
this.files = null;
lblMessage = "done";
return true;
}
15 private bool CancelUpload()
{
string lblMessage = "done";
this.userFile.CancelUpload();
this.files = null;
20 MessageBox.Show("CancelUpload: " + lblMessage);
return true;
}

private void OnUploadCompleted(object sender, Upload...
25 CompletedEventArgs e)
{
string lblMessage = "undefined";
{
if (this.userFile != null)
30 {
switch (e.Reason)
{
case Constants.UploadCompleteReason.UploadCommitted:
var duration = DateTime.Now - this.operationStartTime;
35 var fileSizeInKB = (float)this.userFile.StreamBytes / Bytes...
PerKb;
var fileSizeMessage = fileSizeInKB > BytesPerKb ? string...
.Concat(((float)fileSizeInKB / Bytes...
```

```
                PerKb).ToString(), " MB") :
string.Concat(fileSizeInKB.ToString(), " KB");
lblMessage = string.Format("upload ok in {0} seconds",...
                duration.TotalSeconds);
5  this.userFile = null;
   UploaderDoneNotification(lblMessage);
   break;
   case Constants.UploadCompleteReason.ErrorOccurred:
   lblMessage = string.Format("upload error: {0}", e.Error...
10         Message);
   this.userFile = null;
   UploaderDoneNotification("Upload Finished: " + lblMessage);
   break;
   case Constants.UploadCompleteReason.UserCancelled:
15  if (!sasExpired)
   {
   lblMessage = string.Format("upload cancelled");
   }
   this.userFile = null;
20  UploaderDoneNotification("Upload Finished: " + lblMessage);
   break;
   default:
   lblMessage = string.Format("upload: unknown error occurred");
   this.userFile = null;
25  UploaderDoneNotification("Upload Finished: " + lblMessage);
   break;
   }
   }
   }
30  }
   #endif
   }
   }
```

Bezugszeichenliste

	1	System
	2, 2'	(medizinische) Einrichtung
5	3	Modalität
	4	Computertomograph
	5	C-Bogengerät
	6	Magnetresonanztomograph
	7	(Steuer- und Auswerte-) Rechner
10	8	Device
	9	Personal-Computer
	10	Bildschirm
	11	Tablet-Computer
	12	Smartphone
15	13	Public Cloud
	14	(Datenübertragungs-) Netz
	15	Intranet
	16	Internet
	17	Firewall
20	18, 18'	Subscription
	19	Cloud Storage
	20	App Store
	21	Cloud-Compute-Service
	22	(Patient-) Hub
25	23	(Worklist-) Hub
	24	(Context-) Hub
	25	(Image-) Hub
	26	Table-Storage
	27	Tweet
30	28	Blob-Storage
	29	URI
	31	Applikation
	32	Applikation
	33	Applikation
35	34	Applikation
	35	Feed
	36	Seite
	37	(graphische) Benutzeroberfläche

	40	Kopfzeile
	41	Suchfeld
	42	Anzeigefeld
	43	Befehlsleiste
5	44	Eingabezeile
	45	Schaltfläche
	46	Vorauswahlfeld
	47	Bild
	48	Schlagwort
10	49	Schaltfläche
	50	Schaltfläche
	51	Rahmen
	52	Seite
	60	Kopfzeile
15	61	Überschrift
	62	Anzeigefeld
	63	Befehlszeile
	64	Schaltfläche
	65	Schaltfläche
20	67	Feldnamen
	68	Feldinhalt
	69	Seite
	70	Kopfzeile
	71	Überschrift
25	72	Anzeigefeld
	73	Befehlszeile
	74	Schaltfläche
	80	Menüzeile
	81	Rahmen
30	92	(2D-Viewing-)Modul
	93	(3D-Viewing-)Modul
	94	(2D-Viewing-)Applikation
	95	(3D-Viewing-)Applikation
	100	Speichertreiber
35		

	P	Patientendatensatz
	W	Aufgabe
	C	Kontextdatensatz
	B	Bilddatensatz
5	V	Ansicht
	Q	Suchanfrage
	R	Anfrage
	K	Keyimage

Patentansprüche

1. Verfahren zur Verwaltung und Bearbeitung von Daten einer medizinischen Einrichtung (2, 2'),

- 5 - bei welchem für mindestens zwei unterschiedliche Arten von medizinischen Datensätzen (P,W,C,B) jeweils separate Speicherbereiche (22-25) in einem Cloud Storage (19) einer Public Cloud (13) zur Verfügung gestellt werden, wobei die in dem jeweiligen Speicherbereich (22-25) gespeicherten Datensätze (P,W,C,B) derart vorgehalten werden, dass ein unmittelbarer Zugriff auf diese Datensätze (P,W,C,B) aus dem Internet (16) bei Vorliegen einer dem Speicherbereich (22-25) zugeordneten Subscription (18, 18') möglich ist,
- 10 - bei welchem für jeden Speicherbereich (22-25) mindestens eine spezifisch zugeordnete Applikation (31-34,94,95) zum Ablauf auf einem Nutzer-Gerät (8,9,11,12) zur Verfügung gestellt wird, die zum Abruf und/oder zur Darstellung der in dem zugeordneten Speicherbereich (22-25) enthaltenen Datensätze (P,W,C,B) eingerichtet ist.

20

2. Verfahren nach Anspruch 1, bei welchem mindestens zwei der nachfolgend aufgezählten Speicherbereiche zur Verfügung gestellt werden:

- 25 - ein Patienten-Speicherbereich (22), in dem jeder Datensatz (P) persönliche und medizinische Daten über einen bestimmten Patienten enthält,
- ein Arbeitslisten-Speicherbereich (23), in dem jeder Datensatz eine von einem medizinischen Nutzer zu erledigende Aufgabe (W) oder Aufgabefolge enthält,
- 30 - ein Bilddaten-Speicherbereich (25), in dem jeder Datensatz (B) Bilddaten enthält und/oder
- ein Kontext-Speicherbereich (24), in dem jeder Datensatz (C) Kontextinformation zu einem Bilddatensatz (B) insbesondere einen Verweis (29) auf den Speicherort des Bilddatensatzes (B) enthält.
- 35

3. Verfahren nach Anspruch 1 oder 2,
bei welchem mittels mindestens einer der Applikationen (31-
34, 94, 95) auf einen in dem zugeordneten Speicherbereich (22-
25) enthaltenen Datensatz (P,W,C,B) unmittelbar zugegriffen
5 wird.

4. Verfahren nach einem der Ansprüche 1 bis 3,
bei welchem mittels mindestens einer der Applikationen (31-
34) ein in dem zugeordneten Speicherbereich (22-25) enthalte-
10 ner Datensatz (P,W,C,B) unmittelbar erzeugt, editiert und ge-
löscht wird.

5. Verfahren nach einem der Ansprüche 1 bis 4,
bei welchem mindestens eine der Applikationen (34) sowohl ei-
15 nem Device-Only-Modus als auch in einem Cloud-Service-Modus
betrieben werden kann, wobei

- in dem Device-Only-Modus der Zugriff und die Darstellung
der in dem zugeordneten Speicherbereich (25) enthaltenen
Datensätze (B) ausschließlich von der Applikation (34) vor-
20 genommen wird,
- in dem Cloud-Service-Modus zum Zugriff und/oder zu Darstel-
lung der in dem zugeordneten Speicherbereich (25) enthalte-
nen Datensätze (B) Rechenleistung (21) der Public Cloud
(13) herangezogen wird.

25

6. Verfahren nach einem der Ansprüche 1 bis 5,
- bei welchem innerhalb mindestens eines Speicherbereichs
(22-24) ein Tabellenspeicher (26) zur Verfügung gestellt
wird, wobei innerhalb des Tabellenspeichers (26) jedem in
30 dem Speicherbereich (22-24) gespeicherten Datensatz (P,W,C)
ein Tabelleneintrag (27) zugeordnet ist, der den zugeordne-
ten Datensatz (P,W,C) enthält, und
- bei welchem der Inhalt des Tabellenspeichers (26) von der
dem Speicherbereich (22-24) zugeordneten Applikation (31-
35 33) zur Extraktion von ausgewählten Tabelleneinträgen (27)
in einer Ergebnisliste (35) unmittelbar durchsuchbar ist.

7. Verfahren nach Anspruch 6,
bei welchem mittels mindestens einer Applikation (31-33) die
Ergebnisliste (35) derart aus dem zugeordneten Speicherbe-
reich (22-24) der Public Cloud (13) in einen lokalen Speicher
5 des Nutzer-Geräts (8,9,11,12) externalisiert wird, dass diese
Ergebnisliste (35) durch die Applikation (31-33) auch ohne
Netzverbindung mit der Public Cloud (13) angezeigt, bearbei-
tet und/oder an eine andere Instanz derselben Applikation
(31-33) oder einer anderen Applikation (31-34,94,95) übermit-
10 telt werden kann.

8. Verfahren nach einem der Ansprüche 1 bis 7,
bei welchem durch eine medizinische Modalität (3-6) ein pati-
entenspezifische Bilddaten enthaltender Bilddatensatz (B) er-
15 zeugt und unmittelbar in dem zugeordneten Speicherbereich
(25) des Cloud Storage (19) ablegt wird.

9. System (1) zur Verwaltung und Bearbeitung von Daten einer
medizinischen Einrichtung (2,2'),
20 - mit mindestens zwei in einem Cloud Storage (19) einer Pub-
lic Cloud (13) angelegten Speicherbereichen (22-25) zur
Speicherung mindestens zweier unterschiedlicher Datenarten
von medizinischen Datensätzen (P,W,C,B), wobei jeder Spei-
cherbereich (22-25) einer bestimmten Datenart (P,W,C,B) zu-
25 gewiesen ist, und wobei die in dem jeweiligen Speicherbe-
reich (22-25) gespeicherten Datensätze (P,W,C,B) derart
vorgehalten sind, dass ein Zugriff auf diese Datensätze
(P,W,C,B) aus dem Internet (16) bei Vorliegen einer dem
Speicherbereich (22-25) zugeordneten Subscription (18,18')
30 möglich ist, und
- mit mindestens einer Applikationen (31-34,94,95) für jeden
Speicherbereich (22-25) zum Ablauf auf einem Nutzer-Gerät
(8,9,11,12), die zum Abruf und zur Darstellung der in dem
zugeordneten Speicherbereich (22-25) enthaltenen Datensätze
35 (P,W,C,B) eingerichtet ist.

10. System (1) nach Anspruch 9,
bei dem die mindestens zwei Speicherbereiche umfassen:

- einen Patienten-Speicherbereich (22), in dem jeder Datensatz (P) persönliche und medizinische Daten über einen bestimmten Patienten enthält,
- einen Arbeitslisten-Speicherbereich (23), in dem jeder Datensatz eine von einem medizinischen Nutzer zu erledigende Aufgabe (W) oder Aufgabefolge enthält,
- einen Bilddaten-Speicherbereich (25), in dem jeder Datensatz (B) Bilddaten enthält, und/oder
- einen Kontext-Speicherbereich (24), in dem jeder Datensatz (C) Kontextinformation zu einem Bilddatensatz (b), insbesondere einen Verweis (29) auf den Speicherort des Bilddatensatzes (B) enthält.

11. System (1) nach Anspruch 9 oder 10,
bei welchem mindestens eine der Applikationen (31-34,94) dazu eingerichtet ist, auf die in dem zugeordneten Speicherbereich (22-25) enthaltenen Datensätze (P,W,C,B) unmittelbar zuzugreifen.

12. System (1) nach einem der Ansprüche 9 bis 11,
bei welchem mindestens eine der Applikationen (31-34) dazu eingerichtet ist, die in dem zugeordneten Speicherbereich (22-25) enthaltenen Datensätze (P,W,C,B) unmittelbar zu erzeugen, zu editieren und zu löschen.

13. System (1) nach einem der Ansprüche 9 bis 12,
bei welchem mindestens eine der Applikationen (34) sowohl einem Device-Only-Modus als auch in einem Cloud-Service-Modus betreibbar ist, wobei

- in dem Device-Only-Modus der Zugriff und die Darstellung der in dem zugeordneten Speicherbereich (25) enthaltenen Datensätze (B) ausschließlich von der Applikation (34) vorgenommen wird,
- in dem Cloud-Service-Modus zum Zugriff und/oder zur Darstellung der in dem zugeordneten Speicherbereich (25) enthaltenen Datensätze (B) Rechenleistung (21) der Public Cloud (13) herangezogen wird.

14. System (1) nach einem der Ansprüche 9 bis 13,

- bei welchem mindestens ein Speicherbereich (22-24) einen Tabellenspeicher (26) umfasst, wobei innerhalb des Tabellenspeichers (26) jedem in dem Speicherbereich (22-24) gespeicherten Datensatz (P,W,C) ein Tabelleneintrag (27) zugeordnet ist, der den zugeordneten Datensatz (P,W) enthält, und

5 - bei welchem der Inhalt des Tabellenspeichers (26) von der dem Speicherbereich (22-24) zugeordneten Applikation (31-33) zur Extraktion von ausgewählten Tabelleneinträgen (27) in einer Ergebnisliste (35) unmittelbar durchsuchbar ist.

15. System (1) nach Anspruch 14,

bei welchem mindestens eine der Applikationen (31-33) dazu eingerichtet ist, die Ergebnisliste (35) derart aus dem zugeordneten Speicherbereich (22-24) des Cloud Storage (19) in einen lokalen Speicher des Nutzer-Geräts (8,9,11,12) zu externalisieren, dass diese Ergebnisliste (35) durch die Applikation (31-33) auch ohne Netzverbindung mit der Public Cloud
15 (13) angezeigt, bearbeitet und/oder an eine andere Instanz derselben Applikation (31-33) oder einer anderen Applikation (31-34,94,95) übermittelt werden kann.

16. System (1) nach einem der Ansprüche 9 bis 15,

25 mit einem in einer medizinischen Modalität (3-6) implementierten oder implementierbaren Speichertreiber (100), der dazu eingerichtet, einen von der Modalität (3-6) erzeugten, patientenspezifische Bilddaten enthaltenden Bilddatensatz (B) unmittelbar in dem zugeordneten Speicherbereich (25) der Public Cloud (13) abzulegen.
30

FIG 1

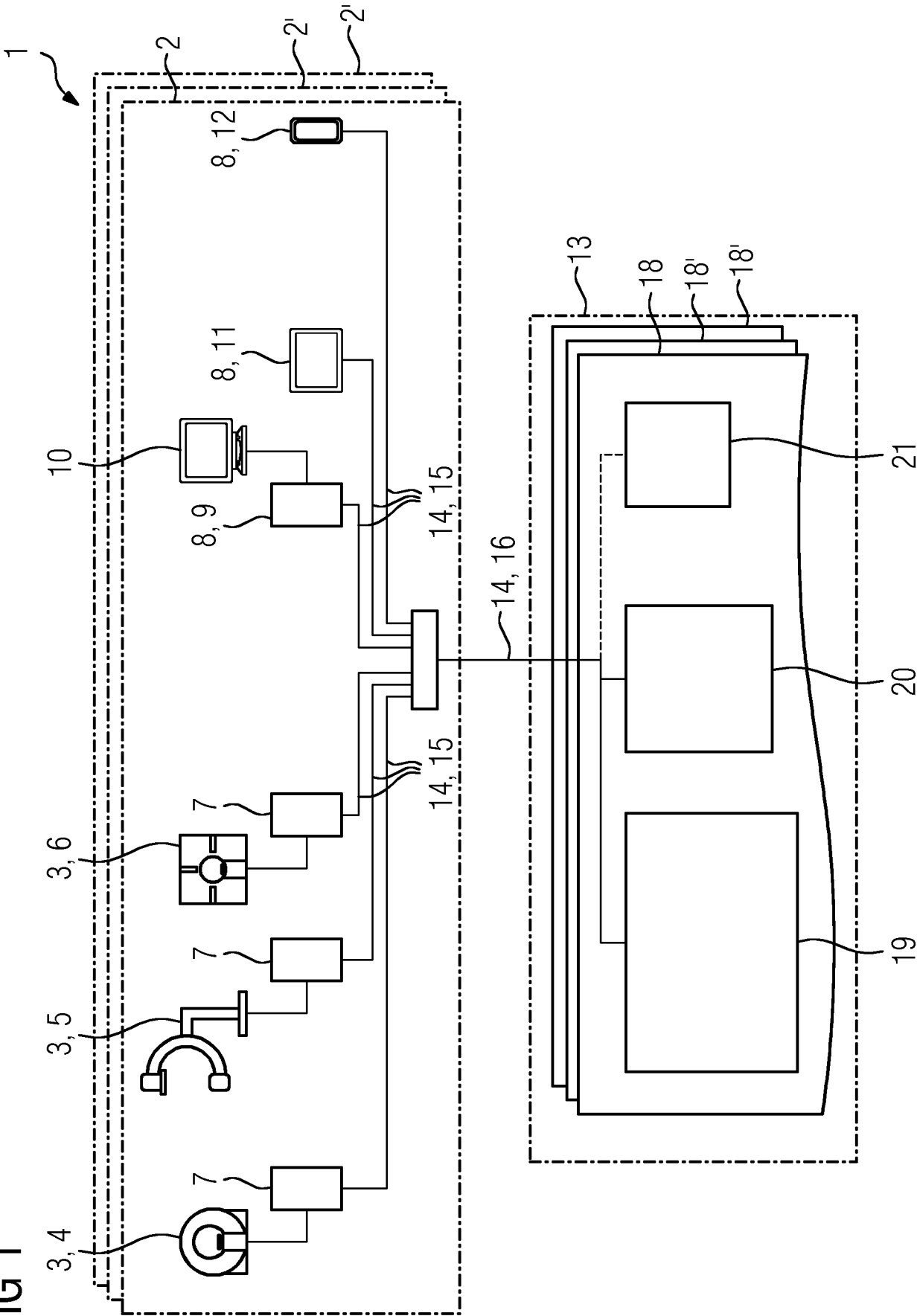


FIG 2

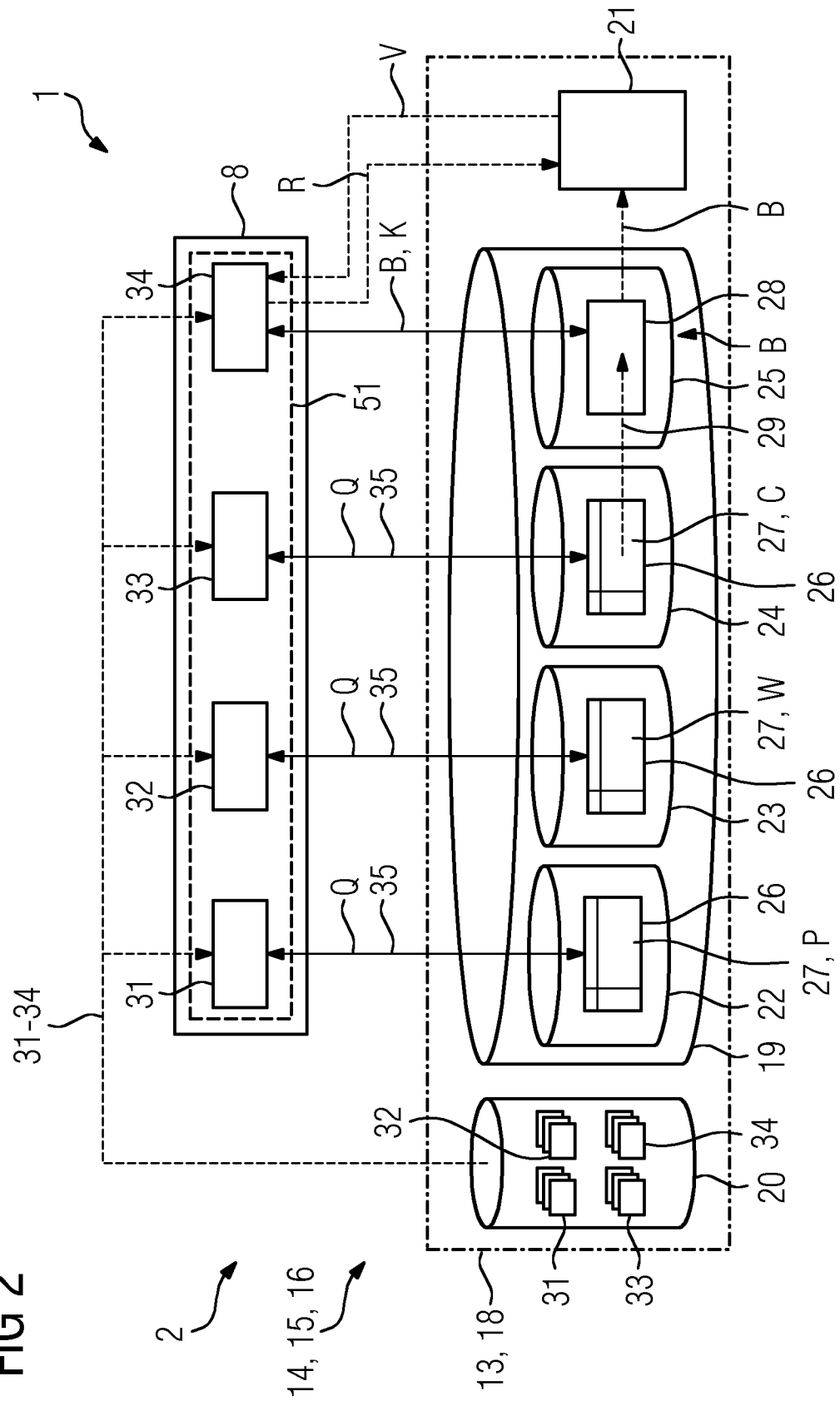


FIG 3

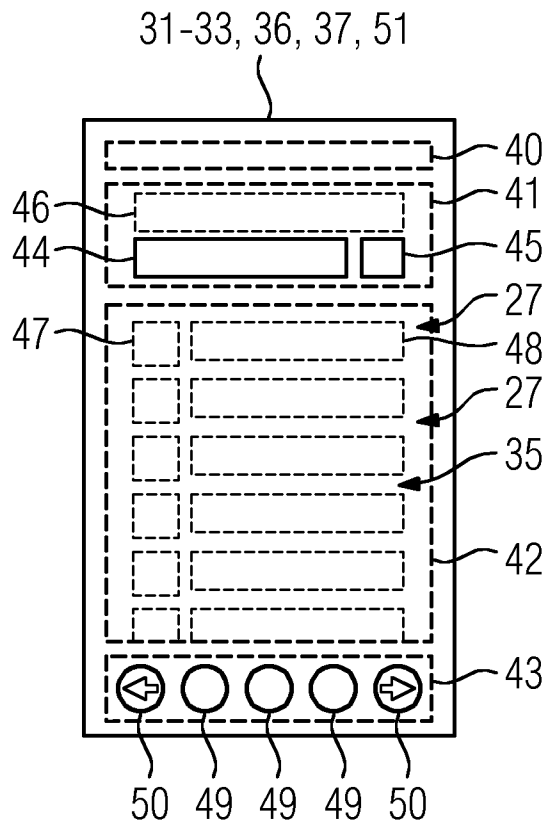


FIG 4

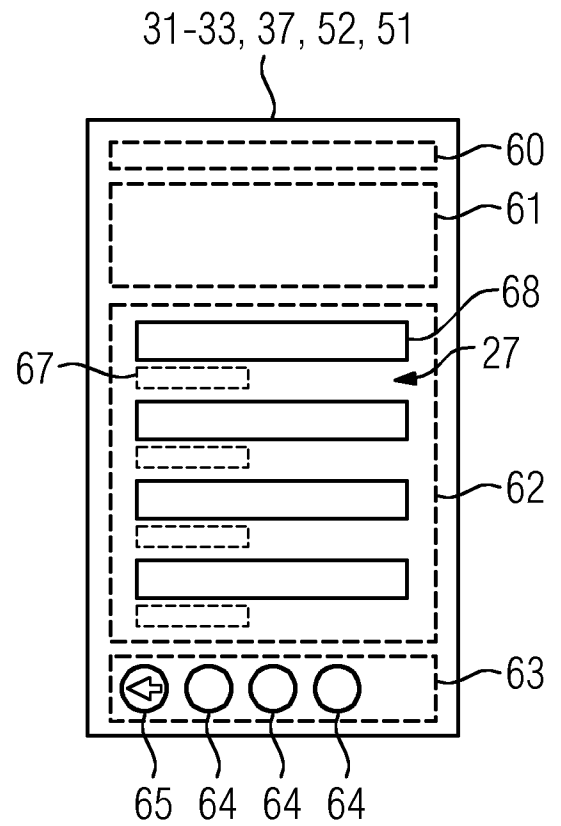


FIG 5

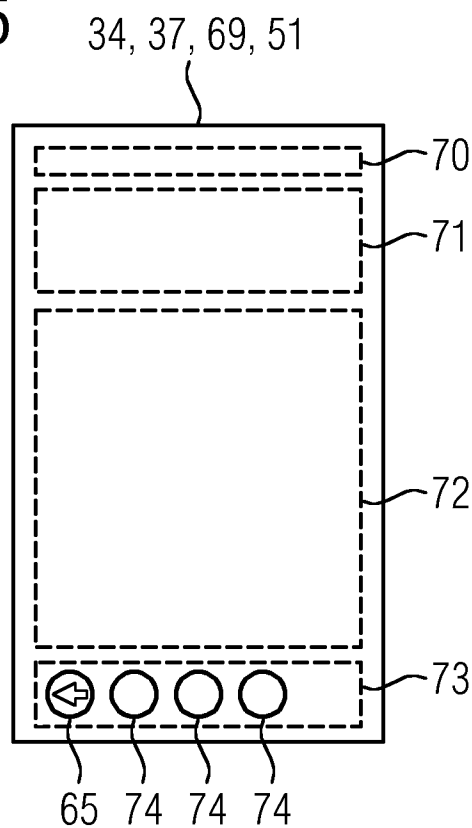


FIG 6

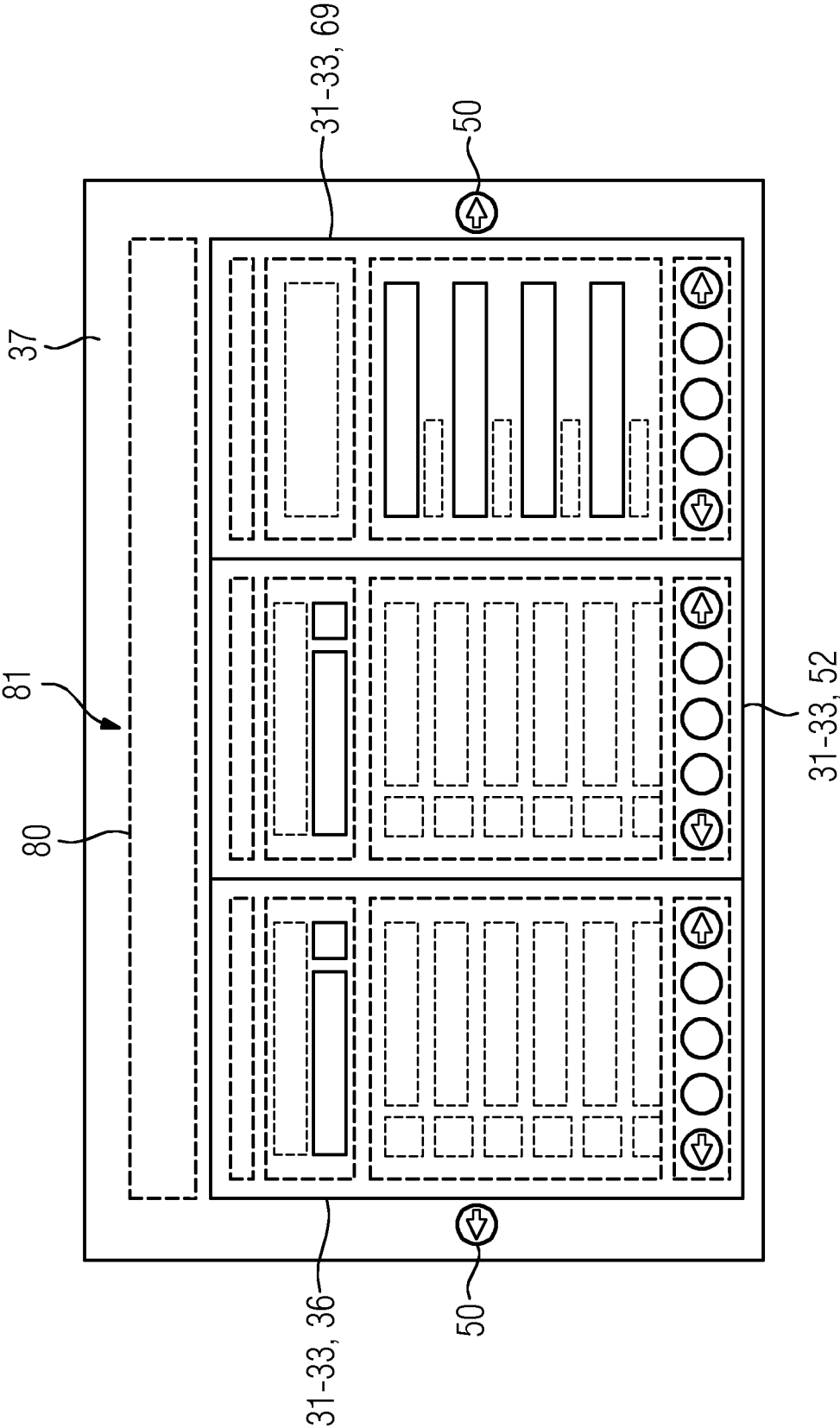
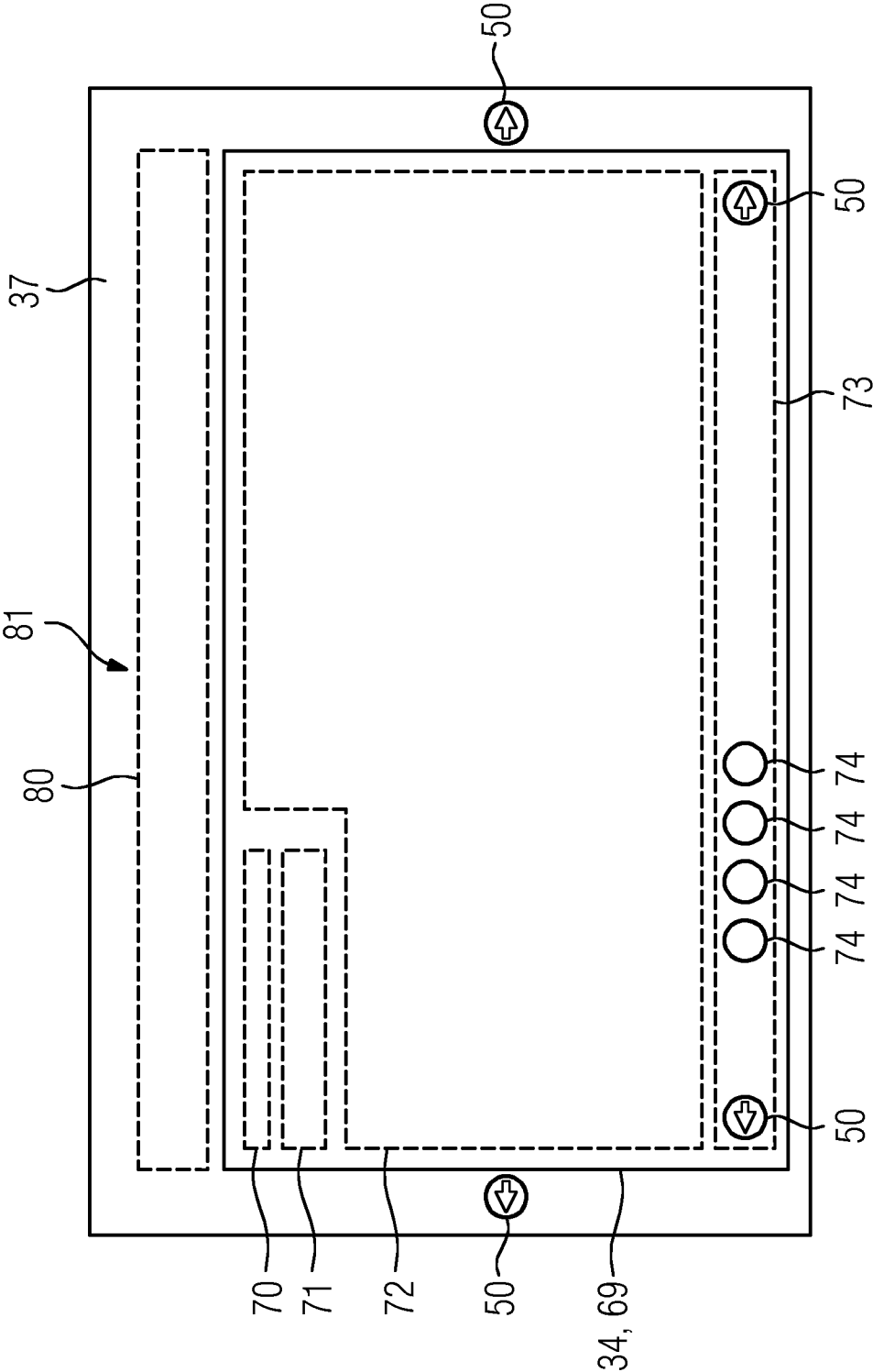


FIG 7



6/7

FIG 8

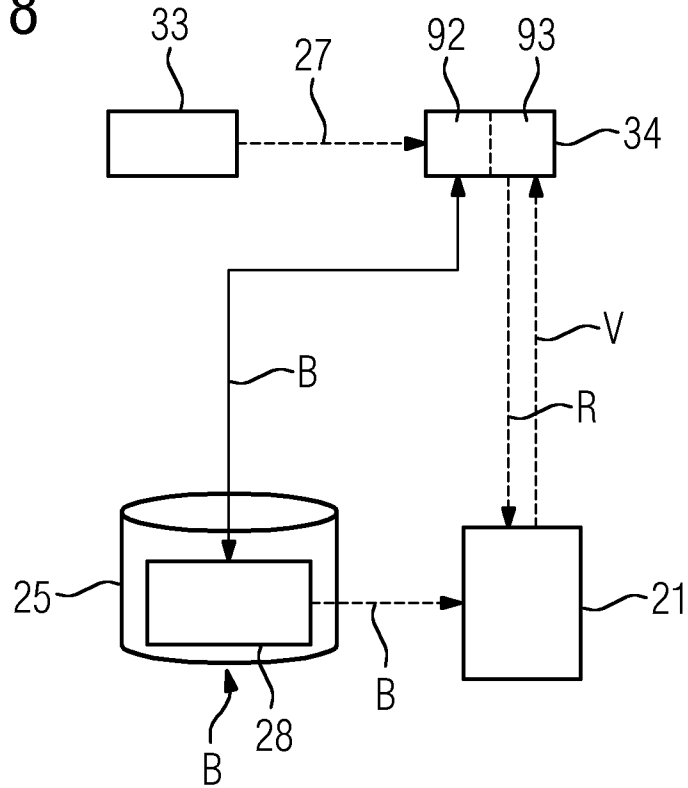


FIG 9

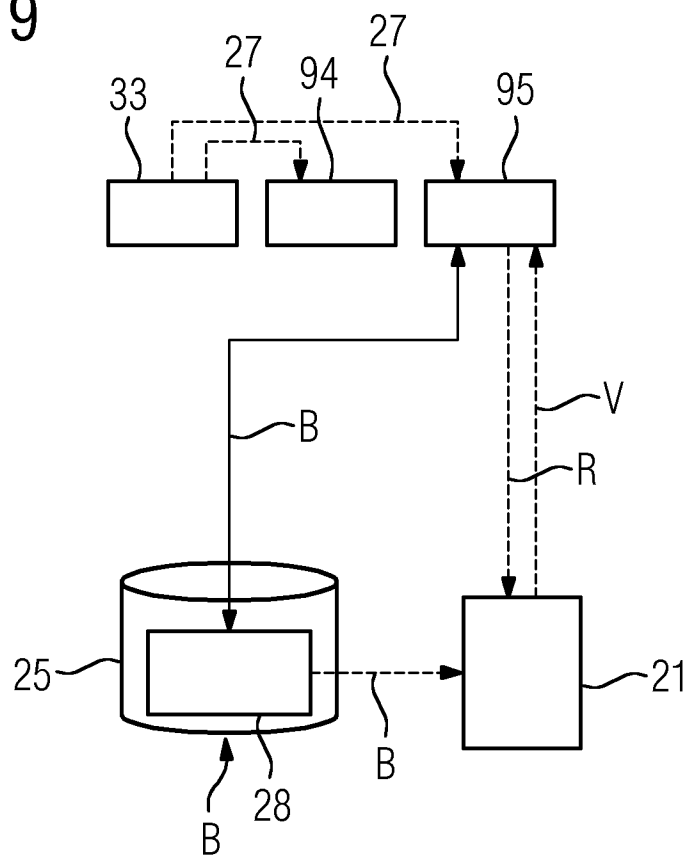
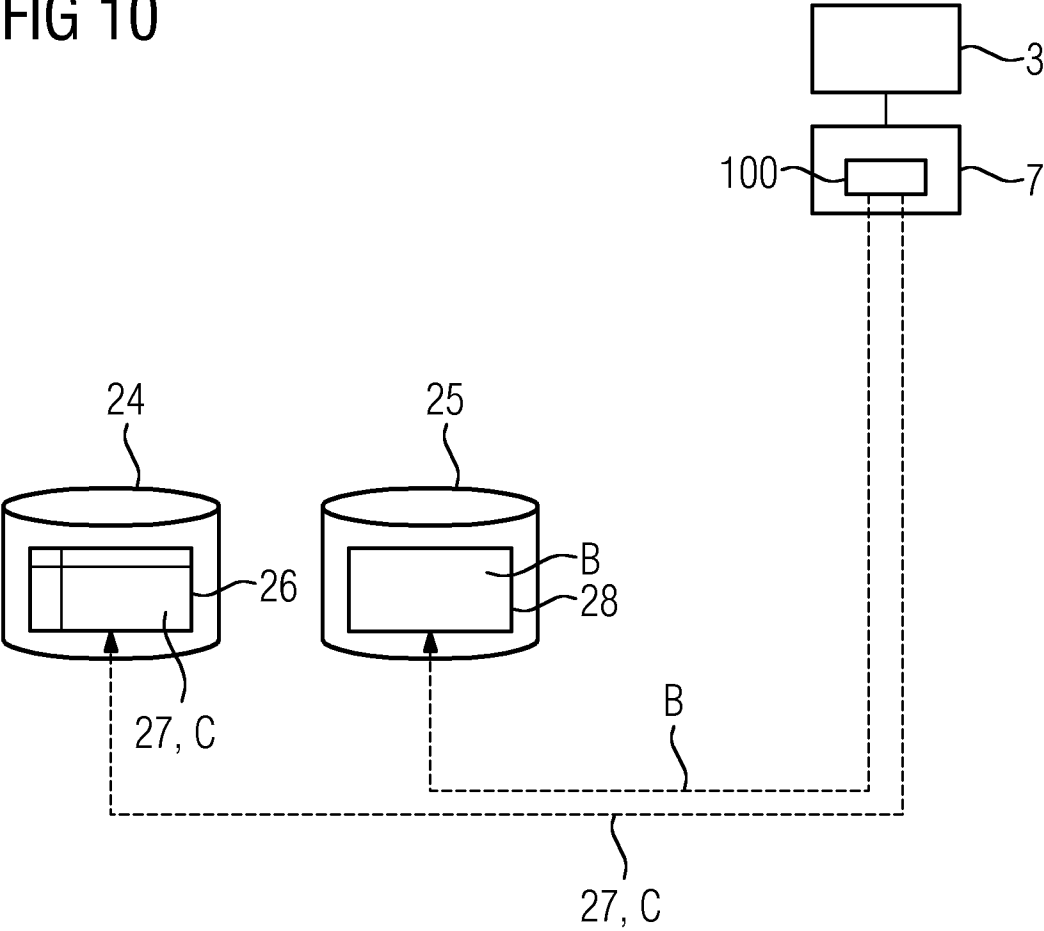


FIG 10



INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2013/070097

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F19/00
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2013/208966 A1 (ZHAO TIECHENG [US] ET AL) 15 August 2013 (2013-08-15) paragraph [0013] - paragraph [0017] paragraph [0031] - paragraph [0033] paragraph [0040] - paragraph [0041] paragraph [0044] paragraph [0051] - paragraph [0055] -----	1-16
X	US 2010/036879 A1 (FRIESE THOMAS [DE] ET AL) 11 February 2010 (2010-02-11) paragraph [0105] - paragraph [0112] paragraph [0086] - paragraph [0093] ----- -/-	1-16



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

16 June 2014

Date of mailing of the international search report

25/06/2014

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Authorized officer

Eichenauer, Lars

INTERNATIONAL SEARCH REPORT

International application No

PCT/EP2013/070097

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	BENJAMIN M ET AL: "From shared data to sharing workflow: Merging PACS and teleradiology", EUROPEAN JOURNAL OF RADIOLOGY, ELSEVIER SCIENCE, NL, vol. 73, no. 1, 1 January 2010 (2010-01-01), pages 3-9, XP026832170, ISSN: 0720-048X [retrieved on 2009-11-14] Kapitel 3 und 4; figures 1, 2 -----	1-16
X	BAHGA ARSHDEEP ET AL: "A Cloud-based Approach for Interoperable Electronic Health Records (EHRs)", IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, IEEE, PISCATAWAY, NJ, USA, vol. 17, no. 5, 1 September 2013 (2013-09-01), pages 894-906, XP011525485, ISSN: 2168-2194, DOI: 10.1109/JBHI.2013.2257818 [retrieved on 2013-08-30] Elemente "EHR Data" und "File Servers"; figure 8 Sektionen "Database Tier" und "Storage Tier"; figure 14 -----	1-16
A	US 2010/049740 A1 (IWASE AKIO [US] ET AL) 25 February 2010 (2010-02-25) paragraph [0037]; figure 2 paragraph [0049] - paragraph [0054] -----	1-16
A	US 2012/233668 A1 (LEAFE EDWARD [US] ET AL) 13 September 2012 (2012-09-13) paragraph [0202] paragraph [0211] paragraph [0224] - paragraph [0245] -----	1-16
A	US 2010/332401 A1 (PRAHLAD ANAND [IN] ET AL) 30 December 2010 (2010-12-30) paragraph [0069] - paragraph [0071] paragraph [0401] -----	1-16
A	PATEL G: "DICOM Medical Image Management the challenges and solutions: Cloud as a Service (CaaS)", COMPUTING COMMUNICATION&NETWORKING TECHNOLOGIES (ICCCNT), 2012 THIRD INTERNATIONAL CONFERENCE ON, IEEE, 26 July 2012 (2012-07-26), pages 1-5, XP032443043, DOI: 10.1109/ICCCNT.2012.6396083 the whole document -----	1-16

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/EP2013/070097

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2013208966 A1	15-08-2013	US 2013208966 A1	15-08-2013
		WO 2013123085 A1	22-08-2013
US 2010036879 A1	11-02-2010	DE 102008037094 A1	26-11-2009
		US 2010036879 A1	11-02-2010
US 2010049740 A1	25-02-2010	EP 2327042 A1	01-06-2011
		JP 2012501015 A	12-01-2012
		US 2010049740 A1	25-02-2010
		US 2013124459 A1	16-05-2013
		WO 2010022375 A1	25-02-2010
US 2012233668 A1	13-09-2012	NONE	
US 2010332401 A1	30-12-2010	AU 2010266433 A1	19-01-2012
		CA 2765624 A1	06-01-2011
		CA 2838107 A1	06-01-2011
		EP 2449477 A2	09-05-2012
		US 2010332401 A1	30-12-2010
		US 2010332454 A1	30-12-2010
		US 2010332456 A1	30-12-2010
		US 2010332479 A1	30-12-2010
		US 2010332818 A1	30-12-2010
		US 2010333116 A1	30-12-2010
		US 2013024424 A1	24-01-2013
		US 2013238572 A1	12-09-2013
		WO 2011002777 A2	06-01-2011

A. KLASSIFIZIERUNG DES ANMELDUNGSGEGENSTANDES

INV. G06F19/00

ADD.

Nach der Internationalen Patentklassifikation (IPC) oder nach der nationalen Klassifikation und der IPC

B. RECHERCHIERTE GEBIETE

Recherchierter Mindestprüfstoff (Klassifikationssystem und Klassifikationssymbole)

G06F

Recherchierte, aber nicht zum Mindestprüfstoff gehörende Veröffentlichungen, soweit diese unter die recherchierten Gebiete fallen

Während der internationalen Recherche konsultierte elektronische Datenbank (Name der Datenbank und evtl. verwendete Suchbegriffe)

EPO-Internal, WPI Data

C. ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	US 2013/208966 A1 (ZHAO TIECHENG [US] ET AL) 15. August 2013 (2013-08-15) Absatz [0013] - Absatz [0017] Absatz [0031] - Absatz [0033] Absatz [0040] - Absatz [0041] Absatz [0044] Absatz [0051] - Absatz [0055] -----	1-16
X	US 2010/036879 A1 (FRIESE THOMAS [DE] ET AL) 11. Februar 2010 (2010-02-11) Absatz [0105] - Absatz [0112] Absatz [0086] - Absatz [0093] ----- -/-	1-16



Weitere Veröffentlichungen sind der Fortsetzung von Feld C zu entnehmen



Siehe Anhang Patentfamilie

* Besondere Kategorien von angegebenen Veröffentlichungen :

"A" Veröffentlichung, die den allgemeinen Stand der Technik definiert, aber nicht als besonders bedeutsam anzusehen ist

"E" frühere Anmeldung oder Patent, die bzw. das jedoch erst am oder nach dem internationalen Anmeldedatum veröffentlicht worden ist

"L" Veröffentlichung, die geeignet ist, einen Prioritätsanspruch zweifelhaft erscheinen zu lassen, oder durch die das Veröffentlichungsdatum einer anderen im Recherchenbericht genannten Veröffentlichung belegt werden soll oder die aus einem anderen besonderen Grund angegeben ist (wie ausgeführt)

"O" Veröffentlichung, die sich auf eine mündliche Offenbarung, eine Benutzung, eine Ausstellung oder andere Maßnahmen bezieht

"P" Veröffentlichung, die vor dem internationalen Anmeldedatum, aber nach dem beanspruchten Prioritätsdatum veröffentlicht worden ist

"T" Spätere Veröffentlichung, die nach dem internationalen Anmeldedatum oder dem Prioritätsdatum veröffentlicht worden ist und mit der Anmeldung nicht kollidiert, sondern nur zum Verständnis des der Erfindung zugrundeliegenden Prinzips oder der ihr zugrundeliegenden Theorie angegeben ist

"X" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann allein aufgrund dieser Veröffentlichung nicht als neu oder auf erfinderischer Tätigkeit beruhend betrachtet werden

"Y" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese Verbindung für einen Fachmann naheliegend ist

"&" Veröffentlichung, die Mitglied derselben Patentfamilie ist

Datum des Abschlusses der internationalen Recherche

16. Juni 2014

Absendedatum des internationalen Recherchenberichts

25/06/2014

Name und Postanschrift der Internationalen Recherchenbehörde

Europäisches Patentamt, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Bevollmächtigter Bediensteter

Eichenauer, Lars

C. (Fortsetzung) ALS WESENTLICH ANGESEHENE UNTERLAGEN		
Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	BENJAMIN M ET AL: "From shared data to sharing workflow: Merging PACS and teleradiology", EUROPEAN JOURNAL OF RADIOLOGY, ELSEVIER SCIENCE, NL, Bd. 73, Nr. 1, 1. Januar 2010 (2010-01-01), Seiten 3-9, XP026832170, ISSN: 0720-048X [gefunden am 2009-11-14] Kapitel 3 und 4; Abbildungen 1, 2 -----	1-16
X	BAHGA ARSHDEEP ET AL: "A Cloud-based Approach for Interoperable Electronic Health Records (EHRs)", IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, IEEE, PISCATAWAY, NJ, USA, Bd. 17, Nr. 5, 1. September 2013 (2013-09-01), Seiten 894-906, XP011525485, ISSN: 2168-2194, DOI: 10.1109/JBHI.2013.2257818 [gefunden am 2013-08-30] Elemente "EHR Data" und "File Servers"; Abbildung 8 Sektionen "Database Tier" und "Storage Tier"; Abbildung 14 -----	1-16
A	US 2010/049740 A1 (IWASE AKIO [US] ET AL) 25. Februar 2010 (2010-02-25) Absatz [0037]; Abbildung 2 Absatz [0049] - Absatz [0054] -----	1-16
A	US 2012/233668 A1 (LEAFE EDWARD [US] ET AL) 13. September 2012 (2012-09-13) Absatz [0202] Absatz [0211] Absatz [0224] - Absatz [0245] -----	1-16
A	US 2010/332401 A1 (PRAHLAD ANAND [IN] ET AL) 30. Dezember 2010 (2010-12-30) Absatz [0069] - Absatz [0071] Absatz [0401] -----	1-16
A	PATEL G: "DICOM Medical Image Management the challenges and solutions: Cloud as a Service (CaaS)", COMPUTING COMMUNICATION&NETWORKING TECHNOLOGIES (ICCCNT), 2012 THIRD INTERNATIONAL CONFERENCE ON, IEEE, 26. Juli 2012 (2012-07-26), Seiten 1-5, XP032443043, DOI: 10.1109/ICCCNT.2012.6396083 das ganze Dokument -----	1-16

INTERNATIONALER RECHERCHENBERICHT

Angaben zu Veröffentlichungen, die zur selben Patentfamilie gehören

Internationales Aktenzeichen

PCT/EP2013/070097

Im Recherchenbericht angeführtes Patentdokument		Datum der Veröffentlichung		Mitglied(er) der Patentfamilie		Datum der Veröffentlichung
US 2013208966	A1	15-08-2013	US	2013208966 A1		15-08-2013
			WO	2013123085 A1		22-08-2013

US 2010036879	A1	11-02-2010	DE	102008037094 A1		26-11-2009
			US	2010036879 A1		11-02-2010

US 2010049740	A1	25-02-2010	EP	2327042 A1		01-06-2011
			JP	2012501015 A		12-01-2012
			US	2010049740 A1		25-02-2010
			US	2013124459 A1		16-05-2013
			WO	2010022375 A1		25-02-2010

US 2012233668	A1	13-09-2012	KEINE			

US 2010332401	A1	30-12-2010	AU	2010266433 A1		19-01-2012
			CA	2765624 A1		06-01-2011
			CA	2838107 A1		06-01-2011
			EP	2449477 A2		09-05-2012
			US	2010332401 A1		30-12-2010
			US	2010332454 A1		30-12-2010
			US	2010332456 A1		30-12-2010
			US	2010332479 A1		30-12-2010
			US	2010332818 A1		30-12-2010
			US	2010333116 A1		30-12-2010
			US	2013024424 A1		24-01-2013
			US	2013238572 A1		12-09-2013
			WO	2011002777 A2		06-01-2011
