University of Applied Sciences and Arts Northwestern Switzerland
School of Business

Betriebliche Informationssysteme

L5 – Systemarchitektur

L5 – Systemarchitektur

„Was ist eine Systemarchitektur?“

- **Wie sind System aufgebaut? Aus welchen Hauptkomponenten bestehen sie?**
- **Die historische Entwicklung von Systemarchitekturen verstehen**
- **Basis Systemarchitektur Entscheidungen einordnen und anwenden können**
- **Erste Kenntnisse zur Thematik Sicherheit in der Cloud erlangen**

- ***Weiterführende Literatur***
  - *Schubert, Winkelmann (2023). Betriebswirtschaftliche Anwendungssysteme: https://link.springer.com/chapter/10.1007/978-3-658-40945-6_1*
  - *Sicherheit in der Cloud - Wirtschaftsinformatik reloaded https://www.fhnw.ch/plattformen/iwi/2021/11/25/sicherheit-in-der-cloud/*

Historische Entwicklung von Systemarchitekturen

Systemarchitektur?

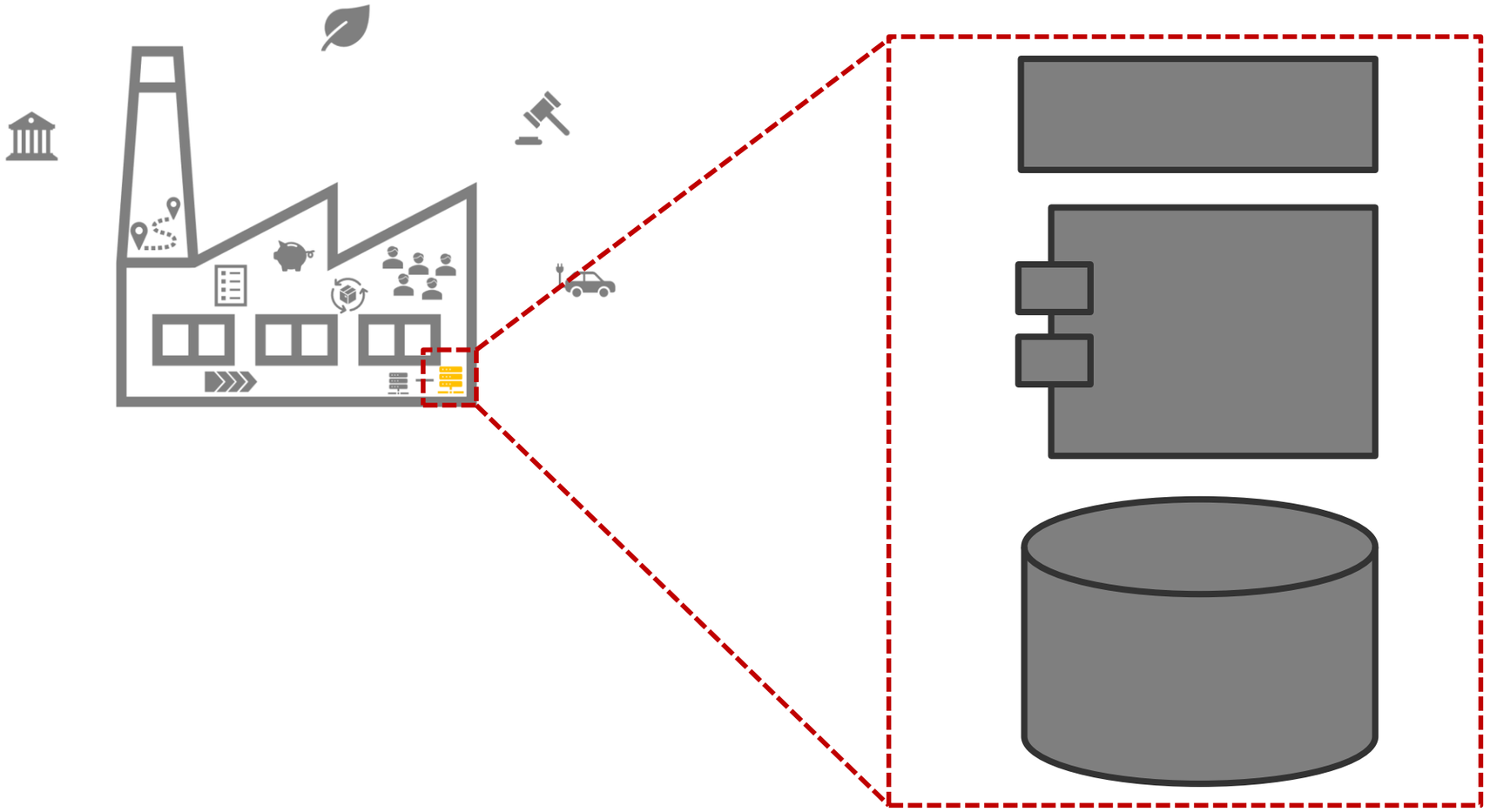
Systemarchitektur bezieht auf die **Struktur** eines Systems, die deren **Komponenten**, ihre **Interaktion und** ihre **Beziehungen** zueinander beschreibt. Es handelt sich um ein konzeptionelles Framework, das dazu dient, die **Gesamtheit eines Systems** (Hardware, Software, Netzwerk, etc.) **zu verstehen**.

Aus der Systemarchitektur kann eine mögliche **Skalierbarkeit, Sicherheit, Zuverlässigkeit & Wartbarkeit** interpretiert werden. Sie definiert die Grenzen des Systems.

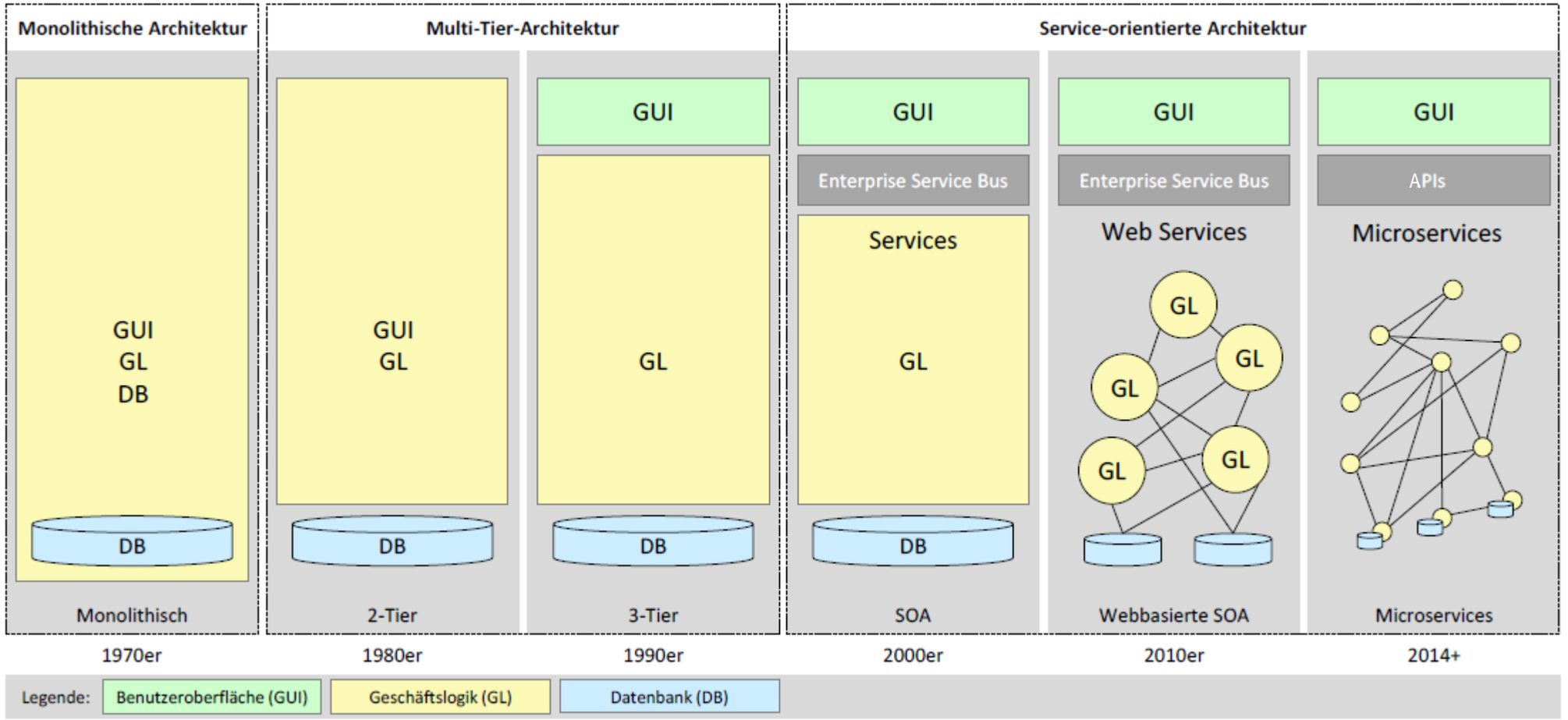
Die Systemarchitektur wird vor einer Einführung erstellt und muss **kontinuierlich** den aktuellen Gegebenheiten **angepasst** werden.

Ein jeweils relevanter Entscheid ist, ob ein System in der Cloud oder onPrem vorhanden sein soll, weil…

Big Picture



Architekturformen – historische Entwicklung



Quelle: Schubert, Winkelmann (2023). Betriebswirtschaftliche Anwendungssysteme

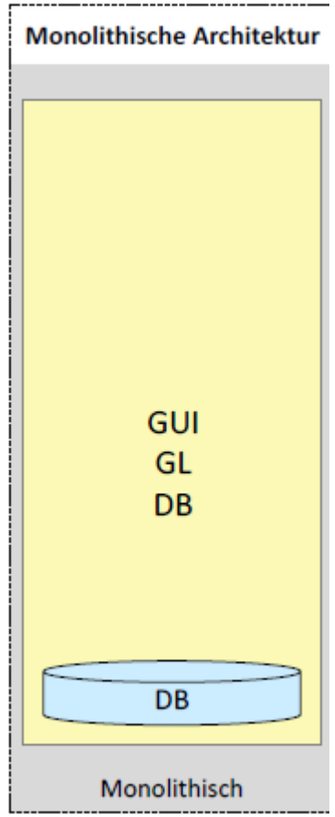
Übung: GUI, GL, DB

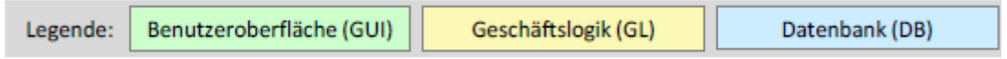
Überlegt euch für ein Hotelbuchungssystem, welche Aufgaben die jeweilige Schicht (GUI, Geschäftslogik, Datenbank) übernimmt.

2-3er Gruppen / 5 min

Monolith

- **ERP-Systeme der ersten Stunde**
- **Datenbank, Applikation und Benutzerschnittstelle untrennbar miteinander verbunden (Mainframe)**
- ☺ **Hohe Performance (aus damaliger Sicht) durch Optimierung auf spezifische Hardware.**
- ☺ **Einheitliche, integrierte Lösung ohne Abhängigkeiten zwischen Komponenten.**
- ☹ **Kaum Skalierbarkeit: Änderungen erfordern oft die komplette Neuentwicklung des Systems.**
- ☹ **Geringe Flexibilität und schwierige Wartung.**





Quelle: Schubert, Winkelmann (2023). Betriebswirtschaftliche Anwendungssysteme

Mehr-Schichten-Architektur

- Einführung von mehreren Schichten (z. B. 2-Tier, 3-Tier, n-Tier).
- Trennung von Präsentation, Geschäftslogik und Datenhaltung.

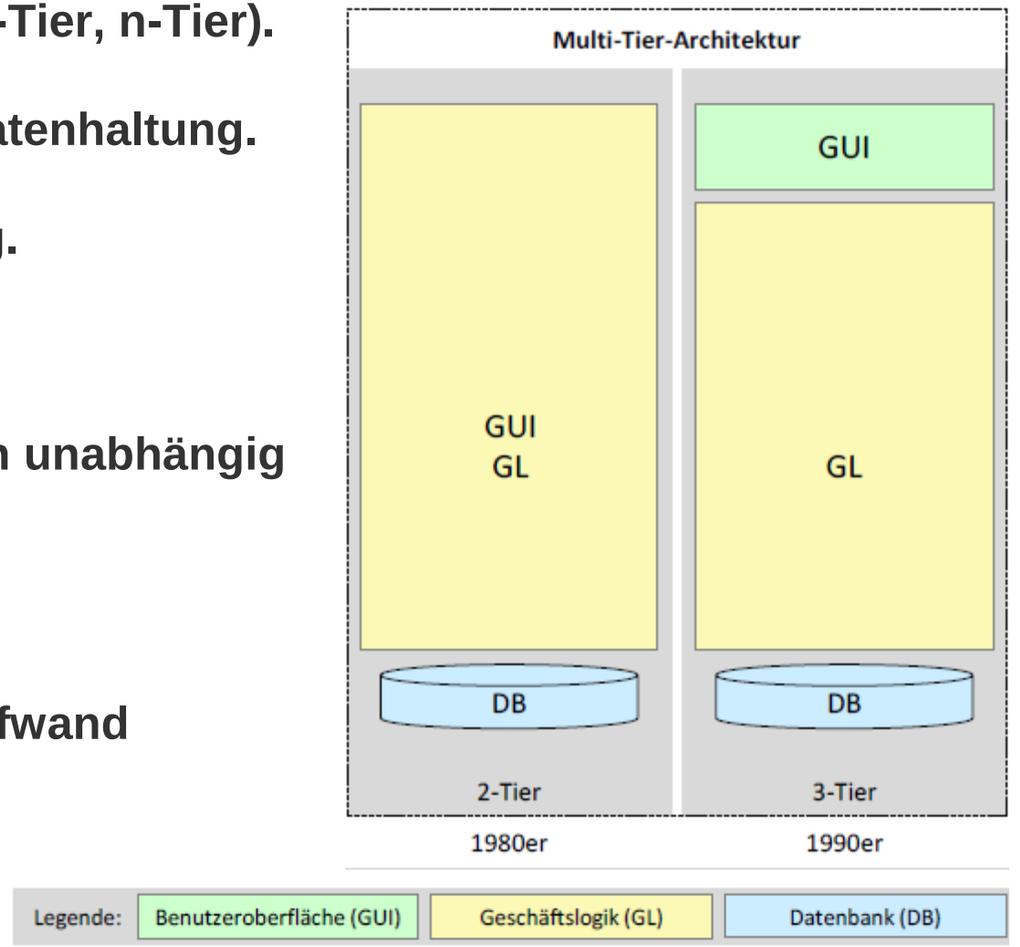
☺ Bessere Skalierbarkeit durch verteilte Verarbeitung.

☺ Flexiblere Entwicklung durch getrennte Schichten.

☺ Verbesserte Wartbarkeit, da einzelne Komponenten unabhängig weiterentwickelt werden können.

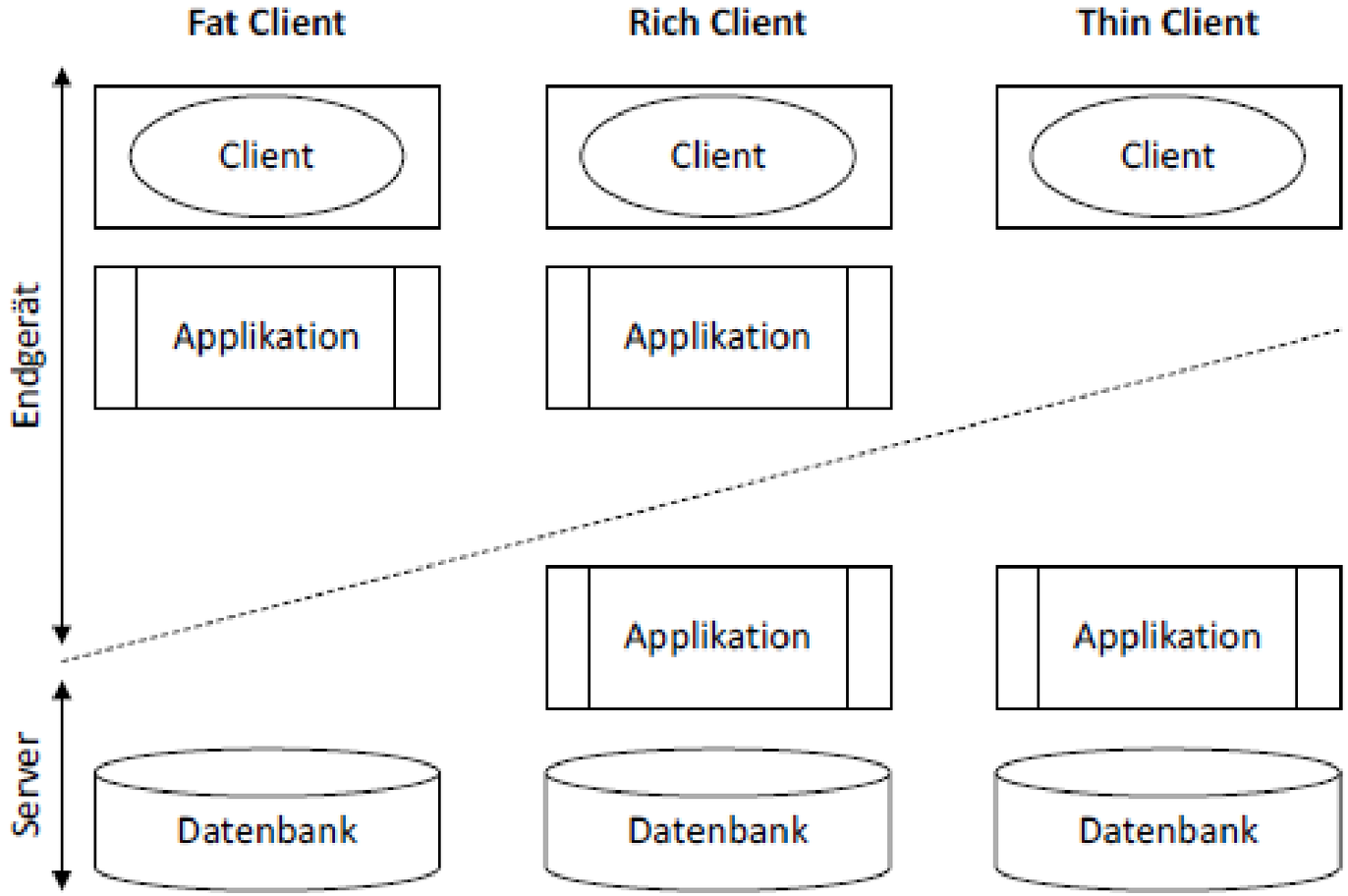
☹ Höhere Komplexität durch verteilte Architekturen.

☹ Performance-Einbussen durch Kommunikationsaufwand zwischen den Schichten



Quelle: Schubert, Winkelmann (2023). Betriebswirtschaftliche Anwendungssysteme

Client-Server-Architektur: Thin, Rich und Fat Client



Quelle: Schubert, Winkelmann (2023). Betriebswirtschaftliche Anwendungssysteme

Übung: Thin, Rich und Fat Client

Überlegt euch, welche Thin, Rich und Fat Client ihr verwendet.

2-3er Gruppen / 5 min

Service-orientierten Architekturen (SOA)

- **SOA ist keine eigentliche Technologie, sondern ein Paradigma, d.h. es vermittelt eine Idee für die Aufteilung und das Zusammenwirken von verteilten Komponenten eines Anwendungssystems.**
- **Die Aufteilung von Komponenten im SOA Paradigma zog den Bedarf einer Kommunikation zwischen Diensten mit sich. Im Beispiel eine zentrale Middleware-Lösung (Software) - hier Enterprise Service Bus (ESB).**
- **Diese führten allerdings nicht im erwarteten Umfang zur Komplexitätsreduzierung, sondern erhöhten diese teilweise sogar noch.**

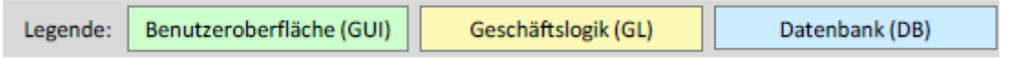
☺ **Höhere Flexibilität durch Wiederverwendbarkeit einzelner Dienste.**

☺ **Bessere Interoperabilität zwischen verschiedenen Systemen.**

☹ **Höherer Entwicklungs- und Betriebsaufwand.**

☹ **Komplexitätsreduzierung konnte nicht erreichen werden, sie erhöhte sich teilweise sogar.**

☹ **Performance-Einbussen**



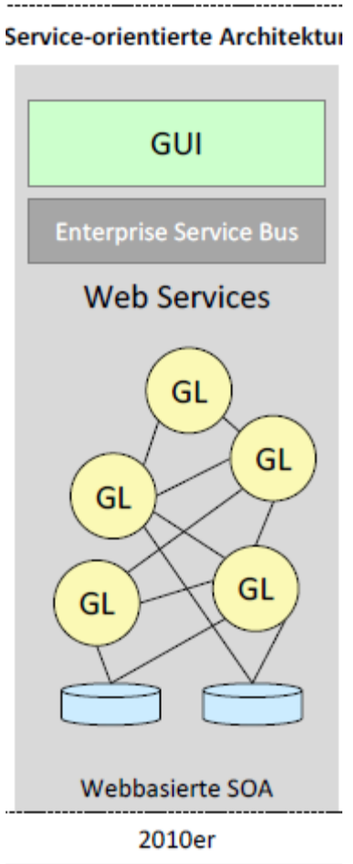
Quelle: Schubert, Winkelmann (2023). Betriebswirtschaftliche Anwendungssysteme

Web Services

- **Webservices bieten im Wesentlichen die Möglichkeit, entfernte Funktionsaufrufe (engl. remote procedure calls, RPC) über eine Webinfrastruktur zu tätigen.**
- **Ein Beispiel ist der Aufruf eines Webservice für die Abfrage des Versandfortschritts eines Pakets, das durch einen Lieferdienst transportiert wird.**
- **Webservices können sowohl von Endbenutzergeräten (z. B. im Browser), von Applikationen oder von anderen Webservices über das Internet aufgerufen werden.**

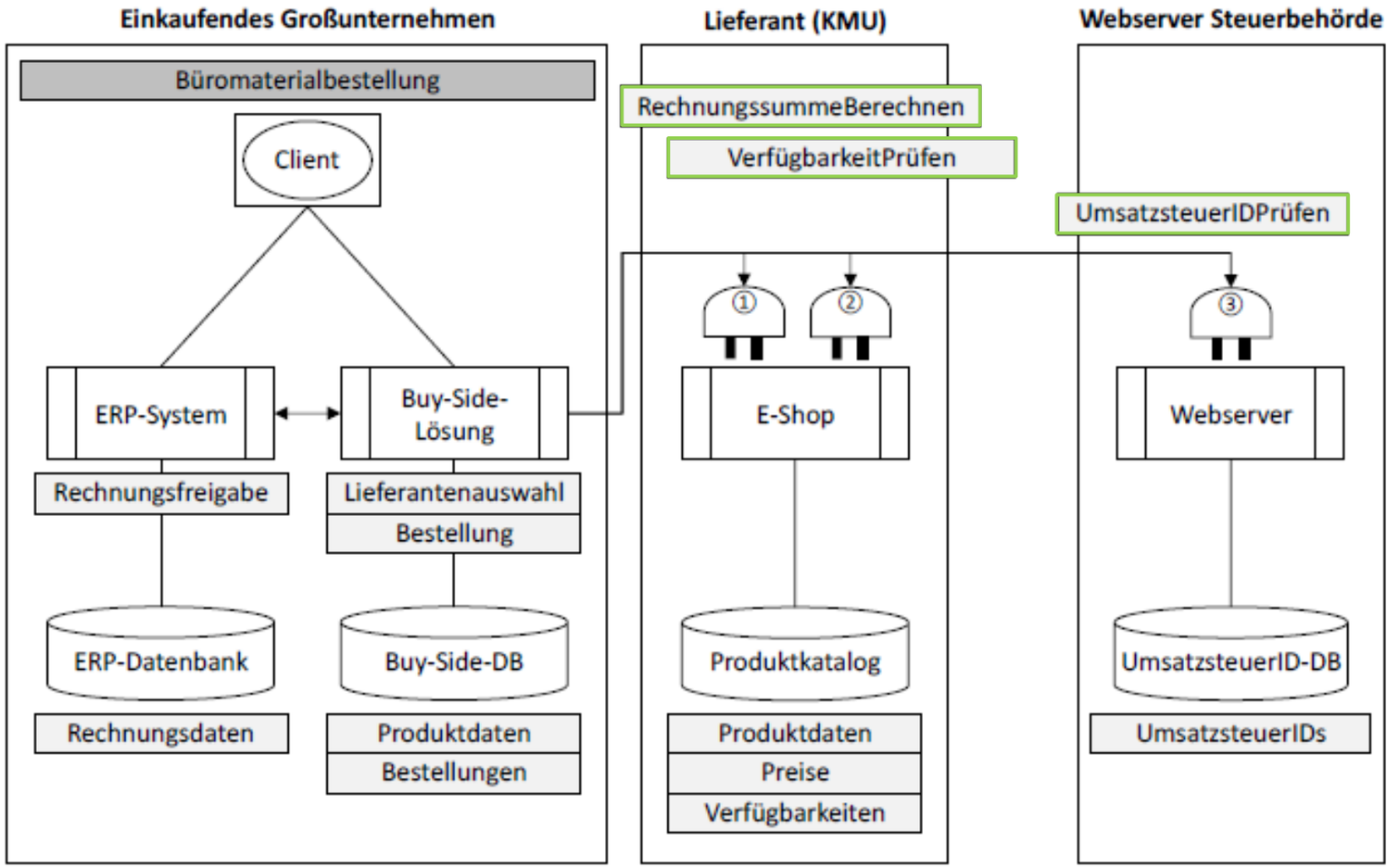
☺ **Plattform- und sprachunabhängig durch standardisierte Web-Schnittstellen.**

☹ **Erfordert hohe Sicherheitsstandards für Web-APIs.**



Quelle: Schubert, Winkelmann (2023). Betriebswirtschaftliche Anwendungssysteme

Webservices als „Steckdosen“



Quelle: Schubert, Winkelmann (2023). Betriebswirtschaftliche Anwendungssysteme

Microservices

- **Weiterentwicklung von SOA, wobei Anwendungen in kleine, eigenständige Dienste zerlegt werden.**
- **Jeder Microservice läuft unabhängig und kommuniziert über APIs (Schnittstellen).**

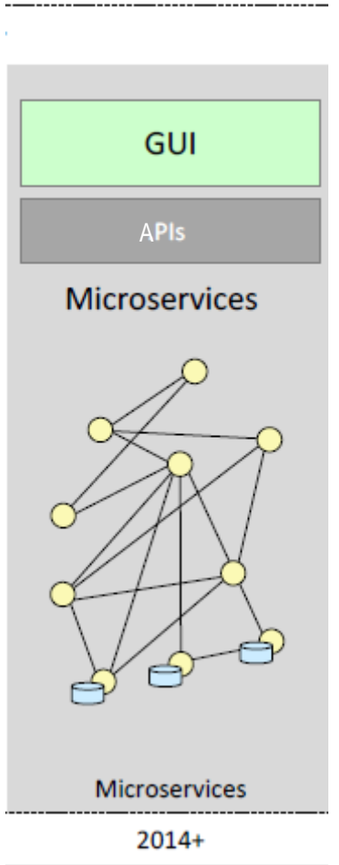
☺ **Maximale Skalierbarkeit durch verteilte Dienste.**

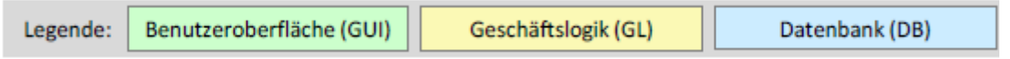
☺ **Schnelle Bereitstellung und Wartung einzelner Services.**

☺ **Verbesserte Fehlertoleranz (Ausfall eines Dienstes betrifft nicht das gesamte System).**

☹ **Hohe Komplexität bei der Verwaltung vieler kleiner Dienste.**

☹ **Notwendigkeit eines leistungsfähigen Orchestrierungs- und Monitoring-Systems (z. B. Kubernetes)**





Quelle: Schubert, Winkelmann (2023). Betriebswirtschaftliche Anwendungssysteme

Übung: Microservices und Docker

- **Lasst euch von ChatGPT etc. oder durch googeln Docker erklären.**
- **Überlegt euch, wie Microservices und Docker zusammenhängen.**

2-3er Gruppen / 10 min

Quelle Docker-Icon: https://www.techwithkunal.com/blog/getting-started-with-docker

Webservice vs Microservice?

Serverless computing

- **Serverless ist eine Möglichkeit, Microservices zu implementieren**
- **„Serverlos" bedeutet, dass die Server für Benutzende (i.d.R. Entwickler*in) unsichtbar sind und diese nicht verwalten muss.**
- **Es werden nur genutzte Kapazitäten bezahlt.**
- **Der Cloud-Service-Anbieter (CSP) fährt die erforderlichen Rechenressourcen je nach Bedarf hoch und runter.**

Kriterium	**Microservices**	**Serverless Computing**
Lang laufende Prozesse	✅ Gut geeignet	❌ Nicht ideal (Funktionen sind kurzlebig)
Hohe Kontrolle über Infrastruktur	✅ Ja	❌ Gering (Cloud-Anbieter verwaltet Infrastruktur)
Ereignisgesteuerte Workloads	⚠️ Möglich, aber komplex	✅ Ideal (wird nur bei Bedarf ausgeführt)
Skalierung	⚠️ Manuell oder Container-basiert	✅ Automatisch durch Cloud
Kosten bei wenig Nutzung	❌ Fixe Serverkosten	✅ Nur bezahlen, wenn Code läuft

Quelle: ChatGPT (2025) Prompt «Vergleiche Microservices und Serverless Computing gemäss https://www.ibm.com/de-de/topics/serverless». Zugriff 31.01.2025

Weitere Basis Systemarchitektur Entscheidungen

Begriffe: OnPrem vs Cloud

OnPrem

- Das Unternehmen betreibt die Hardware und Software in den eigenen Räumlichkeiten.
- Wartung, Sicherheit und Updates liegen vollständig in der Verantwortung der IT-Abteilung des Unternehmens.

Cloud

- IT-Ressourcen (z. B. Rechenleistung, Speicher, Anwendungen) werden über das Internet bereitgestellt und von einem Drittanbieter verwaltet.
- Unternehmen können IT-Infrastruktur mieten, anstatt sie selbst zu betreiben.

Quelle: OpenAI. (2024). *ChatGPT* [Large language model]. /g/g-2fkFE8rbu-dall-e

Begriffe: Standard vs Individualsoftware

Standardsoftware ist für einen breiten Kundenkreis konzipiert und bietet vorkonfigurierte Funktionen, die durch Customizing und Parametrisierung (vgl. Folien Einführung ERP) angepasst werden können. Vorteile sind:

- Kosteneffizienz: Geringere Entwicklungs- und Einführungskosten.
- Release-Fähigkeit: Standardsoftware ist einfacher zu aktualisieren und unterstützt langfristig den Betrieb

Eine **Individualisierung** (durch Eigenentwicklung oder Modifikation des Quellcodes) bietet:

- Massgeschneiderte Lösungen: Spezifische Anforderungen können präzise abgedeckt werden.
- Höhere Komplexität: Anpassungen können die Wartung erschweren und Updates problematisch machen

Quelle: Schubert, Winkelmann (2023). Betriebswirtschaftliche Anwendungssysteme / Alpar et.al. (2019) Anwendungsorientierte Wirtschaftsinformatik

Eigenschaften: Standard vs Individual

	Standardsoftware	Individualsoftware
Funktionsumfang	Deckt ein eingeschränktes Anforderungsprofil ab.	Deckt die individuellen Anforderungen zu 100 % ab.
Einsatzzweck	Werden in Unternehmen oft dort eingesetzt, wo allgemeine Prozesse abgebildet werden, etwa in der Lohnbuchhaltung, dem Kunden- oder Warenmanagement. Beispiele sind SAP oder DATEV.	Sind an die jeweiligen Bedürfnisse der Branche oder des Unternehmens angepasst und kommen oft dort zum Einsatz, wo individuelle Anforderungen erfüllt werden müssen um Geschäftsziele zu erreichen oder Prozesse zu optimieren. Beispiele für Individualsoftwarelösungen sind: - Eine Verwaltungssoftware für eine Zeitarbeitsfirma - Ein ERP-System für den Messebau - Eine Visual-Merchandising-Lösung für den Einzelhandel - Eine Software zur Risikoeinschätzung, Bewertung und Freigabe von Großprojekten - Ein Dokumentenmanagementsystem auf Basis von Mircrosoft SharePoint
Anpassung	Das Anforderungsprofil kann nur begrenzt erweitert werden und Anpassungen sind meist mit einem hohen Zeitaufwand verbunden.	Anpassungen sind jederzeit möglich, wodurch auf (Markt-)veränderungen flexibel reagiert werden kann.
Verfügbarkeit	sofort	Entwicklung kann Wochen oder Monate dauern.
Integration	Integration Ist zeit- und kostenintensiv, da die jeweiligen Prozesse an die Standardsoftware angepasst werden müssen. Ist schnell und unkompliziert, da die Software speziell für die jeweilige Systemarchitektur entwickelt wurde. Eignet sich daher besonders gut, wenn viele Drittsysteme vorhanden sind.	Ist schnell und unkompliziert, da die Software speziell für die jeweilige Systemarchitektur entwickelt wurde. Eignet sich daher besonders gut, wenn viele Drittsysteme vorhanden sind.
Kosten	Meist günstiger in der Anschaffung als eine Individualsoftware. Hinzu kommen jedoch die monatlichen Lizenzgebühren sowie weitere Kosten für Anpassungen etc.	Meist mit höheren Initialkosten verbunden, führt jedoch im Laufe der Zeit zu geringeren Folgekosten und mehr Zeitersparnis.
Lizenzen	Lizenzen werden vollständig erworben oder auf Zeit gemietet. Die vollen Lizenzkosten sind auch dann fällig, wenn Unternehmen nur einen Teil der Software tatsächlich nutzen.	Lizenzen liegen zu 100 % beim Auftraggeber.

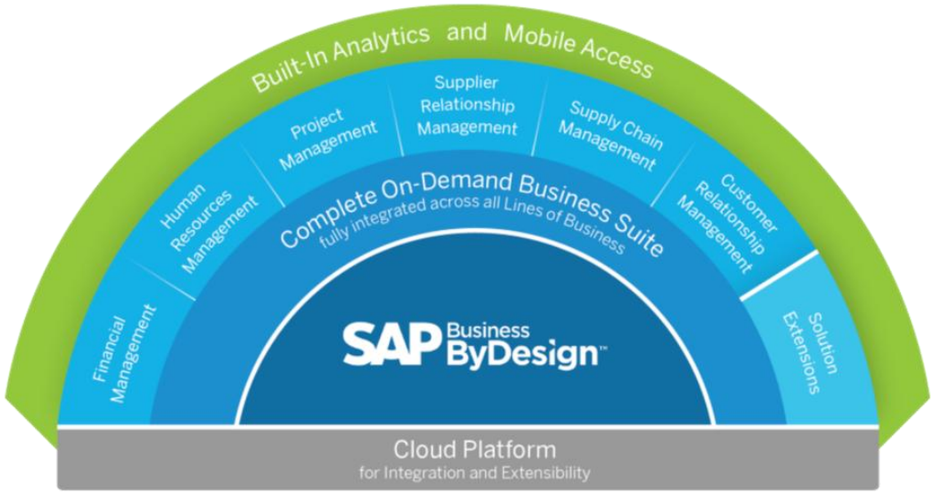
Quelle: https://www.digital-experts.com/blog/standardsoftware-vs-individualsoftware-das-sind-die-unterschiede

Begriffe: Best of Breed vs Best of Suite

Der Begriff „**Best of Breed**“ bedeutet per Definition, dass ein Unternehmen, anstatt auf eine Komplettlösung von einem Anbieter zu setzen (**Best of Suite**), die Strategie verfolgt, für jeden Unternehmensbereich die optimale Software-Lösung zu integrieren.

„Best of Breed“ kann sinngemäss zu „das Beste seiner Art“ ins Deutsche übersetzt werden

Beispiele?



Quelle: https://neteris.com/en/business-applications/sap-business-by-design/

Eigenschaften: Best of breed vs Best of Suite

Best of Breed	Best of Suite
Der Ansatz kombiniert unterschiedliche spezialisierte Software-Lösungen.	Best of Suite bietet eine ganzheitliche Lösung aus einem Produkt-Bundle.
Bei Best of Breed werden die besten Systeme und Tools benutzt. Diese müssen miteinander integriert und funktional verzahnt werden.	Der Idealfall ermöglicht Best of Suite ein reibungsloses Zusammenspiel der Komponenten und eine leichte Bedienbarkeit.
Die individuellen Bedürfnisse verschiedener Unternehmensbereiche werden mit dem Best-of-Breed-Ansatz leichter abgedeckt, da nach spezialisierten Lösungen gesucht wird.	Personalisierte Anpassungen sind im Leistungsumfang oft nicht vorgesehen. Um individuellen Anforderungen gerecht zu werden, braucht es oft spezielles Customizing.
Die verschiedenen Tools kommen von verschiedenen Anbietern. Ihr Unternehmen ist also nicht von einem einzigen Anbieter und seinen Lösungen abhängig, sondern kann selbstbestimmt agieren. Die Kommunikation wird dadurch aber aufwändiger.	Da die Software nur von einem Anbieter stammt, gibt es auch nur einen Ansprechpartner. Dies erleichtert die Kommunikation und Vertragsverhandlungen, bedeutet aber auch weniger Flexibilität.
Bei Best of Breed entstehen in der Regel weniger Kosten bei der Anpassung der Tools an die internen Strukturen. Die neusten Programme sind darauf ausgelegt, sich schnell in bestehende Systeme integrieren zu können.	Der Best-of-Suite-Ansatz gilt als vergleichsweise teuer und bedeutet für viele Unternehmen im Mittelstand eine größere Investition. Es kann dazu kommen, dass die Kosten-Nutzen-Rechnung nicht zu Gunsten des Unternehmens ausfällt.
Der flexible Umgang mit Tools und Softwares kann für Ihr Unternehmen einen Wettbewerbsvorteil bedeuten, weil Sie stets mit den besten und modernsten Technologien arbeiten.	Manche Tools und Anwendungen, die Unternehmen unbedingt benötigen, sind mit Komplettlösungen nicht kompatibel oder nur umständlich integrierbar. Die Folgen können weniger Funktionalität und eine verzögerte Reaktion auf Marktveränderungen sein.

Quelle: https://www.freshworks.com/freshservice/de/best-of-breed-blog/

Übung: Was ist tendenziell die Wahl?

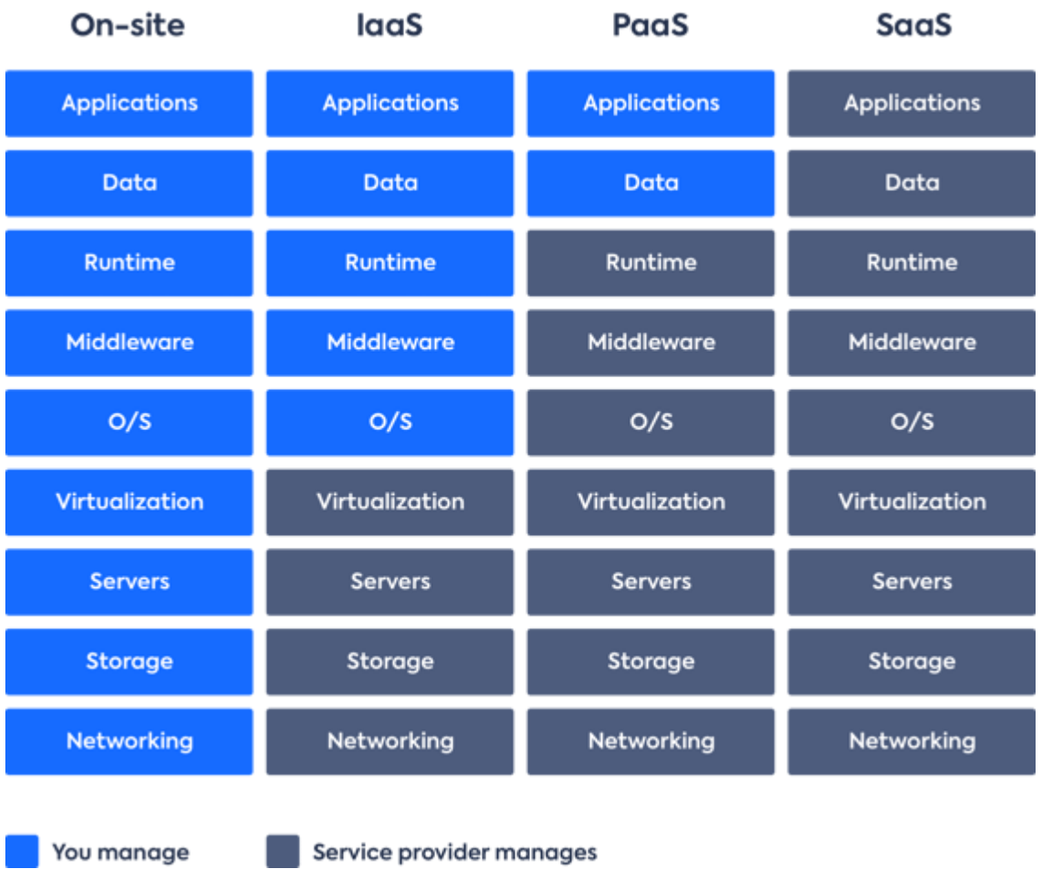
	Standard oder Individual	Best of Breed oder Best of Suite
Ein ERP für eine mittelgrosse Anwaltskanzlei		
Ein Regelsystem für die Steuerung der Energieversorgung einer Gemeinde		
Eine Adressverwaltung für einen Verein		
Eine Prüfungs-Software für eine Hochschule		
Die Software für einen Roboter einer Joghurtabfüllanlage		
…		
Ergänzt noch 3 eigene Vorschläge		

2-3er Gruppen / 10 min

SaaS, PaaS, IaaS

Quelle: OpenAI. (2024). *ChatGPT* [Large language model]. /g/g-2fkFE8rbu-dall-e

Unterschiede SaaS, PaaS, IaaS



Quelle: https://www.leanix.net/de/wiki/saas/iaas-vs-paas-vs-saas

SaaS

SaaS (Software-as-a-Service) wird auf Remote-Servern betrieben und vom Anbieter verwaltet, aktualisiert und gewartet. Dadurch kommt dem Endbenutzer zwar weniger Verantwortung zu – gleichzeitig hat er aber auch weniger Kontrolle über die Services.

SaaS ist ein voll funktionsfähiger Service, auf den über jeden beliebigen Webbrowser zugegriffen werden kann.

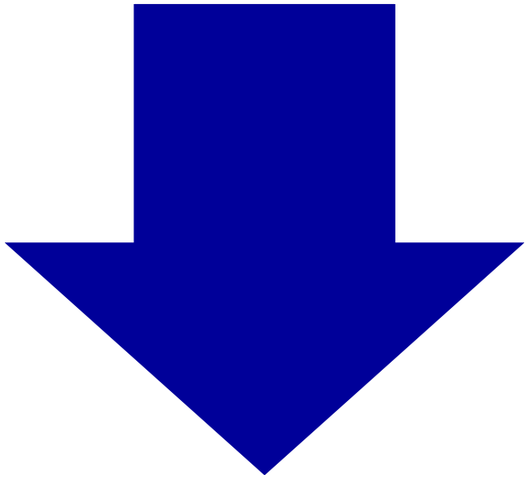
Beispiele von Business Information Systems?

Quelle: https://www.leanix.net/de/wiki/saas/iaas-vs-paas-vs-saas

SaaS: Oft genannte Vor- und Nachteile

Vorteile

- Geringere Kosten
- Skalierbarkeit
- Integration
- Upgrades
- Benutzerfreundlichkeit



Nachteile

- Datensicherheit nicht komplett unter eigener Kontrolle
- Begrenzte Individualisierung
- Interoperabilität
- Geringere Kontrolle
- Redundante Ressourcen
- Schatten-IT

Wann ist SaaS wirklich SaaS?

Wenn die Vorteile auch wirklich genutzt werden können! Eher nicht wenn

- **lokale Installation oder Remote Desktops bspw. Für eine Admin-Komponente benötigt wird**
- **das Lizenzmodell tlw. traditionell ist (Einmalgebühren)**
- **eine Skalierbarkeit nicht oder nur mit viel Vorlauf unterstützt wird**
- **Updates nicht durch den Anbieter durchgeführt werden**
- **…**

PaaS

PaaS steht für Platform-as-a-Service. Die Plattform kann über das Internet genutzt werden und bietet Entwicklern sowohl ein Framework als auch Tools, um massgeschneiderte Applikationen und Software zu entwickeln.

PaaS kann als „Light-Version“ von IaaS betrachtet werden. Genau wie bei IaaS haben die Kunden auch hier Zugriff auf Server und Rechenzentren, die von einem Drittanbieter gewartet und verwaltet werden. Der Hauptnutzen von PaaS liegt jedoch in der Entwicklung passgenauer SaaS-Applikationen.

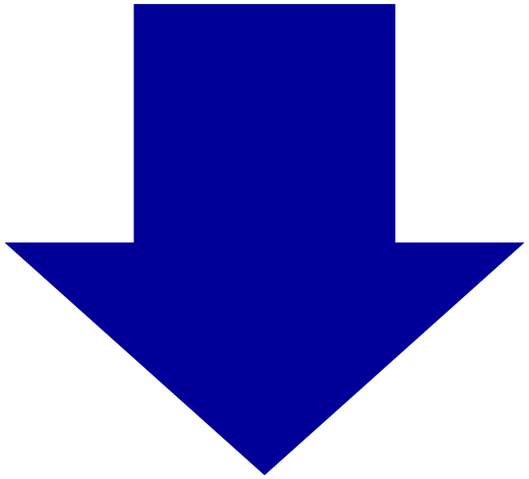
Beispiele von Business Information Systems?

Quelle: https://www.leanix.net/de/wiki/saas/iaas-vs-paas-vs-saas

PaaS: Oft genannte Vor- und Nachteile

Vorteile

- Geschwindigkeit
- Skalierbarkeit
- Verminderte Konfiguration
- Plattform-Themen sind geregelt



Nachteile

- Datensicherheit nicht komplett unter eigener Kontrolle
- Integration
- Know-How für das vorhandene Framework

IaaS

IaaS steht für Infrastructure-as-a-Service. Es ermöglicht Unternehmen, virtualisierte Computerressourcen zu kaufen (z.B. Networking oder Speicherplatz auf Abruf), anstatt Geld in kostspielige Hardware zu investieren. IaaS ist dynamisch skalierbar und bietet Unternehmen mehr Flexibilität als On-Premises-Lösungen.

IaaS kann als Fundament von Cloud Computing angesehen werden. Die virtualisierten Komponenten werden durch das Internet zur Verfügung gestellt und funktionieren genauso wie die Server und Hardware, die Unternehmen sonst in ihren Gebäuden haben.

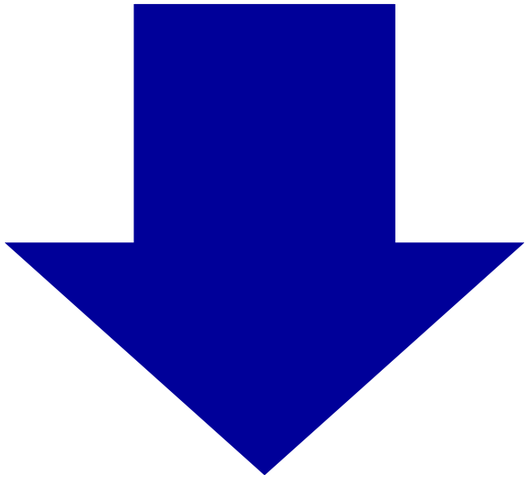
Beispiele von Business Information Systems?

Quelle: https://www.leanix.net/de/wiki/saas/iaas-vs-paas-vs-saas

IaaS: Oft genannte Vor- und Nachteile

Vorteile

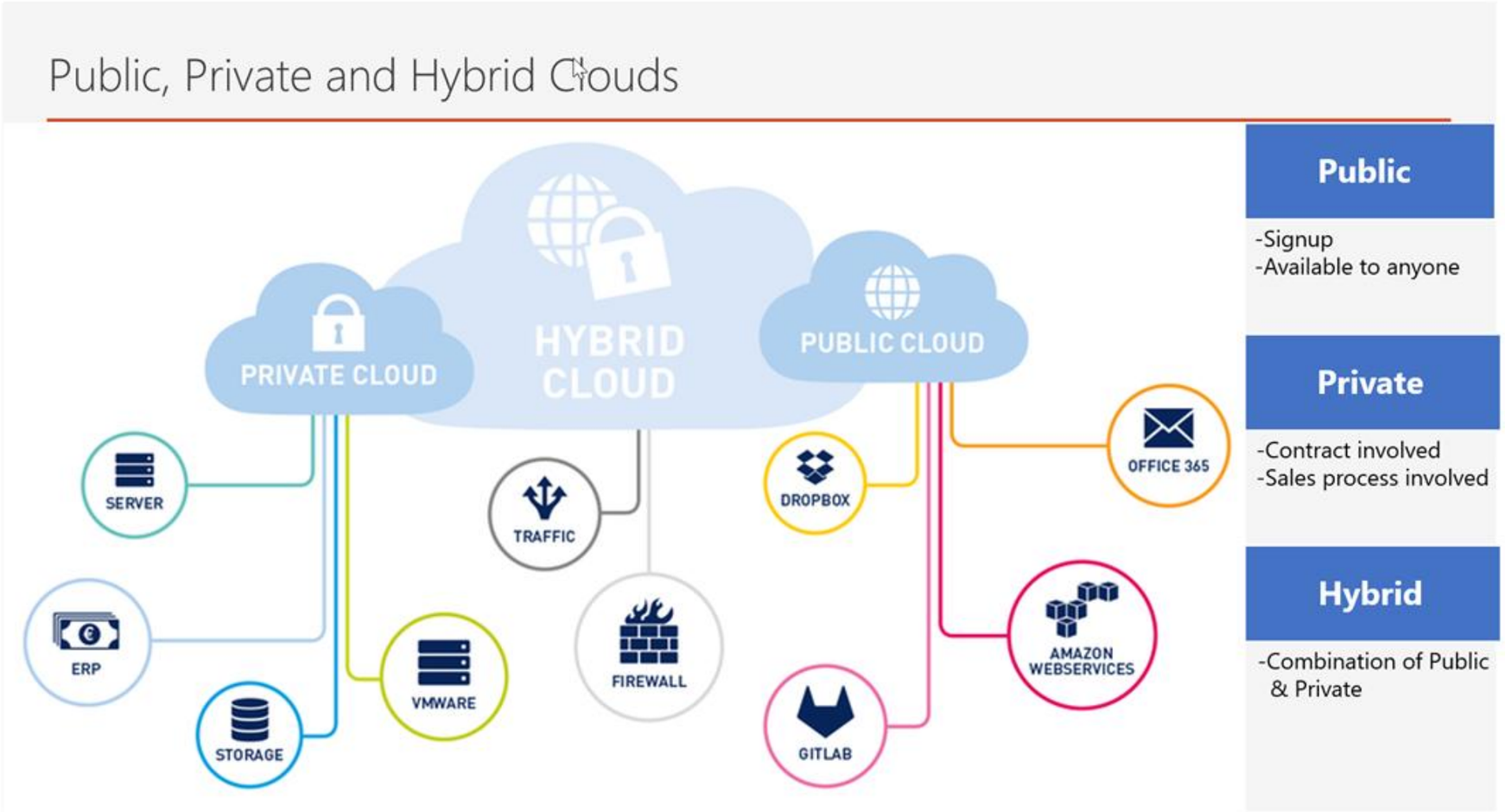
- Flexibilität
- Automatisierung
- Geringere Kosten
- Kontrolle über Infrastruktur
- Skalierbarkeit



Nachteile

- Altsysteme
- Sicherheit nicht komplett unter eigener Kontrolle

Cloud-Infrastrukturen



Quelle: https://www.tech-quantum.com/public-private-and-hybrid-cloud/

Sicherheit in der Cloud

Quelle: Dall-E. Prompt «Grafik zu Sicherheit in der Cloud». Zugriff 31.01.2025

Übung: Sicherheit in der Cloud

Lest den Artikel: [Sicherheit in der Cloud - Wirtschaftsinformatik reloaded (fhnw.ch)]

Der Artikel bezieht sich auf

1. **Prozesse**
2. **Business Continuity**
3. **Technik**

Anforderung	Anforderungen aus der Security- und IT-Abteilung an Systeme	Prozesse	Business Continuity	Technik
Betriebsmodell	On Prem / Private Cloud / Public Cloud: Private Cloud bevorzugt			
Lizenzmodell	Kauf inkl. Wartung & Support / Subscription basiert: Subscription basiert bevorzugt			
Verfügbarkeit (SLA)	Mindestens 99.5% garantiert pro Jahr			
Application Backup/Restore, Lösung	Integrierte Backup-Lösung, Backup mindestens einmal täglich			
Application Backup/Restore, Aufbewahrungszeit	Mindestens 6 Monate im Standard-Umfang vorhanden			
Application Backup/Restore, Speicherung	Backup Daten sind verschlüsselt gespeichert			
Application Monitoring	Möglichkeit Performance, Fehlermeldungen und Logs der Applikation jederzeit zu betrachten			
Deployment	Mindestens 2-Umgebungs-Architektur (DEV/TEST und PROD)			
Software Architektur				
Mobiles Front End	Responsive Design für 100% aller Funktionen / native App oder progressive App unter Android und iOS			
Offline-Verfügbarkeit	Notwendig für User unterwegs			
Protokolle	Traffic zwischen DB, App und Web Frontend mit https verschlüsselt und Verwendung Port 443			
Dokumentation	Software-Architektur ist beschrieben und steht online zur Verfügung			
Authentication & Authorization				
Authentifizierung (Login) & Authorization (Berechtigungen)	Single Sign-On via Azure AD Integration			
Multi Factor Authentication	Via Microsoft Authenticator App			
Datenintegration				
Integrationsarchitektur	APIs RESTful (keine File-Upload basierten Schnittstellen (FTP, SFTP, CSV, XML))			
Format für Datenaustausch	JSON			
Betrieb				
IT-Trainings & Knowledge Base	Werden vom Anbieter als Teil des Services zur Verfügung gestellt			
Release-Zyklen (Updates/Patches)	2x jährlich als Teil des Services			

Gewisse Punkte findet ihr auch in der Liste der Security & (IT-)Architektur Anforderungen. Welche? Markiert diese mit der entsprechenden Farbe

1-3er Gruppen / Zeit: 15 Minuten

Feedback Gruppenarbeit

Auftrag 4

Gruppenarbeit

Auftrag 5