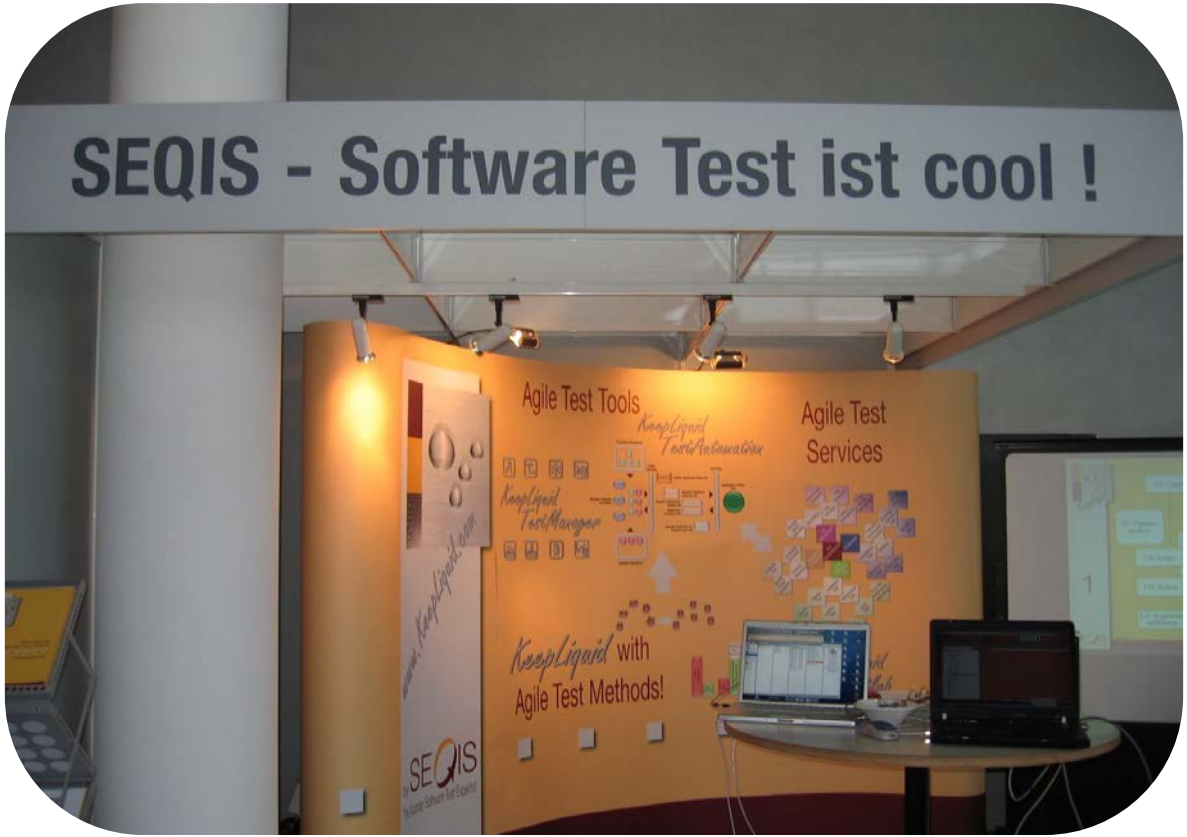
SEQIS

Ten more things

Nachhaltigkeit in der Software-Entwicklung: Wurscht oder Wahnsinn?

Klemens Loschy

Ein Blick zurück ins Jahr 2002 …



Features > Quality & functional > non-functional

functional Requirements

- Einfach zu definieren
  - “Was” soll passieren
- Starke Stakeholder
  - Fachbereich
  - Endbenutzer
- Einfach zu testen
  - manuell
  - standard Testing Know How
- Sofortiger Mehrwert

non-functional Requirements

- schwer zu definieren
  - “Wie” soll etwas sein
  - viele Details wichtig
    - Hardware Ressourcen
    - Datenmenge & -vielfalt
    - App-Benutzung
- Technische Stakeholder
  - Betrieb
- Aufwendig zu testen
  - Experten-Tools
  - Programmierkenntnisse
- Langfristige Investition
  - … nur bei Problemen sichtbar

functional > non-functional

“Bank-Austria: Ausfälle bei Online-Banking erzürnen Kunden”
(Der Standard, 31.10.2012)

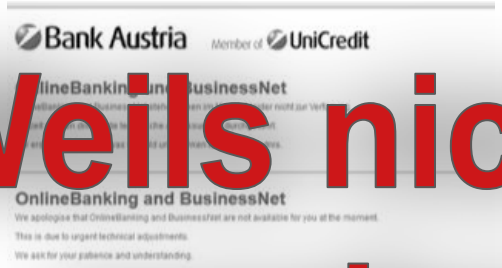
“Don’t Smoke-Volksbegehren: Erneut IT-Probleme bei Unterstützungserklärungen”
(Die Presse, 20.02.2018)

“Sommerzeit abschaffen? Umfrage-Server kollabiert”
(Die Krone, 06.07.2018)

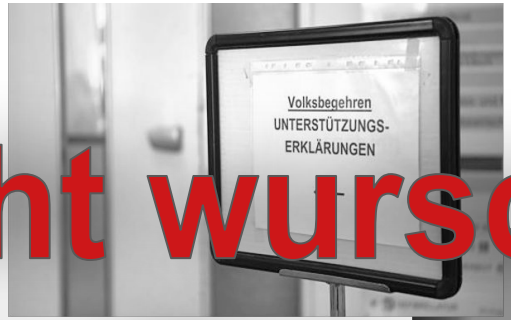
Corona Impfung auf Notruf NOE (10.02.2021)

 Quelle: https://www.derstandard.at, https://www.diepresse.com, https://notrufnue.com

functional > non-functional



“Bank-Austria: [illegible] erzürnen Kunden”
(Der Standard, 31.10.2012)



“Don’t [illegible]-Volksbegehren: erneut [illegible] Unterstützungserklärungen”
(Die Presse, 20.02.2018)

“Sommerzeit abschaffen?
Umfrage-Server kollabiert”
(Die Krone, 06.07.2018)

Corona Impfung auf Notruf NOE (10.02.2021)

Weils nicht wurscht ist, sondern Wahnsinn!

SEQIS

1. Achte auf die Qualität, besonders der non-functional Requirements

- Fix im Prozess verankern
- Einen NFR Verantwortlichen definieren

“Nachhaltigkeit” ist bisher noch kaum Thema

Nachhaltigkeit

SEQIS

Was verstehen wir bei razzfazz.io unter …

Nachhaltigkeit

Was verstehen wir bei razzfazz.io unter …

Ganzheitlich und langfristig Denken und Entscheidungen treffen

Qualität >> Quantität

Nachhaltigkeit

Sinnvoller Einsatz **aller** Ressourcen, besonders der **Mitarbeiter**

… ist nicht zuletzt auch

- Zukunftssicherheit
- Effizienzsteigerung
- Kostenersparnis

mehr als “nur” CO2 Reduktion

2. Konkretisieren Sie Nachhaltigkeit!

- Jeder versteht darunter etwas anderes
- Ist unerlässlich um das Ziel konkret zu verfolgen
- Nachhaltigkeit damit auch messbar machen

Wo müssen wir nachhaltig handeln?

Produktidee & Design

Umsetzungsprozess

Umsetzung & Betrieb

Nachhaltigkeit: Produktidee & Design

Produktidee & Design

Umsetzungsprozess

Umsetzung & Betrieb

Wer ist meine Zielgruppe und Einsatzgebiet?

Wer ist meine Zielgruppe und Einsatzgebiet?

oder

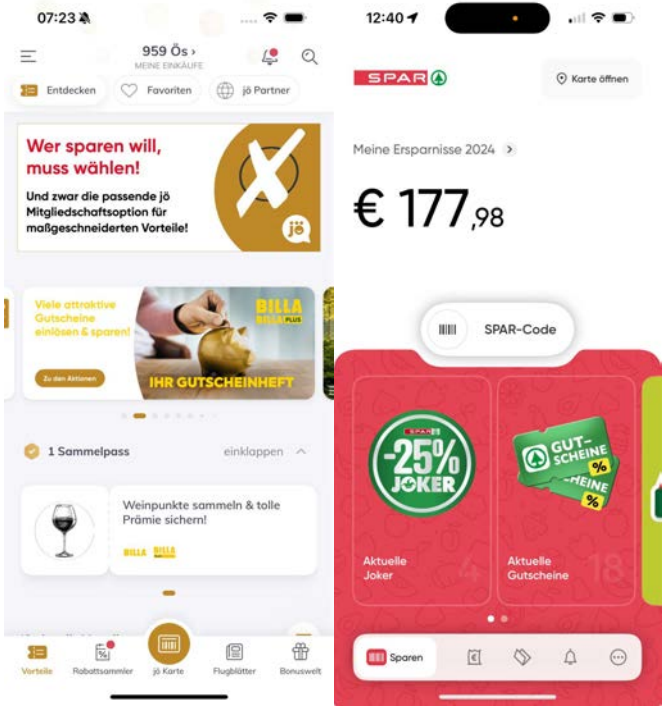
&

&

Keine oder mangelhafte Zielgruppendefinition

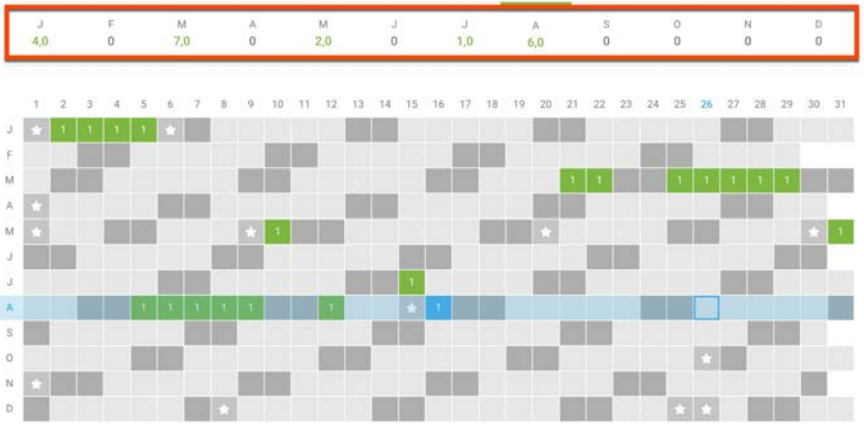
Quelle: ORFON App

Keine oder mangelhafte Zielgruppendefinition



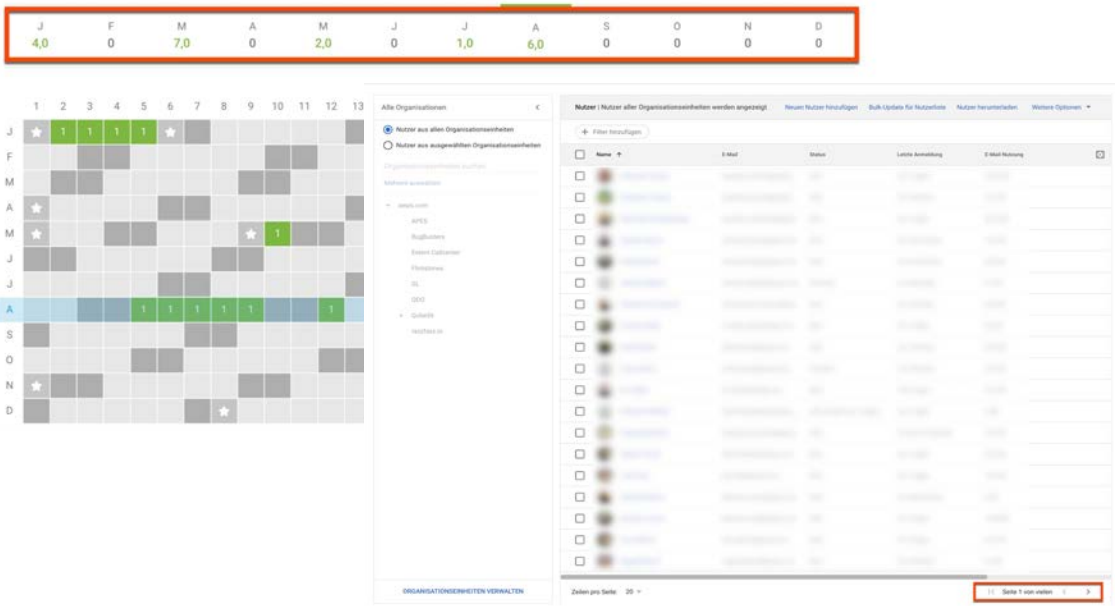
Quelle: Jö Bonus Club App, SPAR App

Also: Reduktion, aber gezielt und sinnvoll



Quelle: https://mocapp.com

Also: Reduktion, aber gezielt und sinnvoll



Quelle: https://admin.google.com

SEQIS

Dynamisch individualisiert mit AI

Man könnte …

- das Benutzerverhalten analysieren (anonymisiert) und dementsprechend den “Welcome Screen” anpassen
- erweiterte Daten wie Position und Uhrzeit ebenfalls berücksichtigen
- damit Predictions ableiten für neue User oder neue Situationen

3. Die Software muss die User unterstützen!

- Personas definieren und deren Bedürfnisse fokussieren
- Sinnvolle (reduzierte) Standardeinstellung und Use Cases setzen

  und nicht “noch mehr” ist automatisch “noch besser”

Nachhaltigkeit: Umsetzungsprozess

Produktidee & Design

Umsetzungsprozess

Umsetzung & Betrieb

Unklare/unpassende Prozesse

Unklare/unpassende Prozesse

Ein guter Prozess …

- ist aktuell und dokumentiert
- ist angepasst an die Gegebenheiten
- ist, wo möglich, automatisiert
- unterstützt den Ablauf

Meeting Hell

Meeting Hell

Daily

Sprint Planning 1

Sprint Planning 2

Review Retro

Backlog Grooming

Meeting Hell

Daily

Sprint Planning 1

Sprint Planning 2

Review Retro

Backlog Grooming

Scrum of Scrums

Weekly

sonst. spontane Abstimmungen und Regelmeetings

Meeting Hell

Wieso machen wir das, was bringt das?

Wieso bin ich da überhaupt dabei? Was ist meine Rolle/Aufgabe?

… und wieso ist *der* überhaupt dabei :)?

Daily

Sprint Planning 1

Sprint Planning 2

Review Retro

Backlog Grooming

Scrum of Scrums

Weekly

sonst. spontane Abstimmungen und Regelmeetings

Meeting Hell

Ein gutes Meeting …

- hat eine kommunizierte Agenda und Zielsetzung
- hat genug Vorlaufzeit
- hat einen gezielten Teilnehmerkreis
- wird dokumentiert
- hat ToDos mit Verantwortlichen und Due Date

Prozesse neu gedacht …

Prozesse neu gedacht …

Prozesse anpassen …

- an die Gegebenheiten und Menschen
- “Stop Liste” einführen
- Quick-Wins zuerst umsetzen
- große Änderungen Schritt für Schritt umsetzen
- Mut zum Misserfolg
- den “Blick von Außen” nutzen

4. Prozesse müssen deren Akteure unterstützen …

- … oder werden Sie bremsen!
- Passen Sie Ihre Prozesse an Ihre Bedürfnisse an
- Führen Sie Meetings zielgerichtet durch

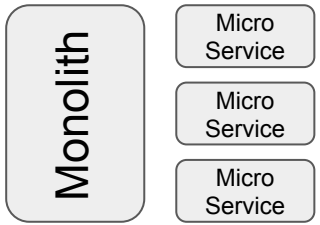
Nachhaltigkeit: Umsetzungsprozess

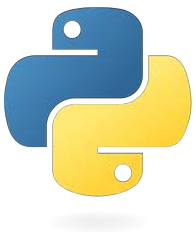
Produktidee & Design

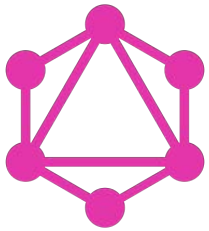
Umsetzungsprozess

Umsetzung & Betrieb

Allgemeine, “große” Architekturentscheidungen







Quelle: https://de.wikipedia.org

SEQIS

Konkrete, “kleine” Architekturentscheidungen

node (http)	express	koa
-	215 kB	96,6 kB
-	20k LoC	15k LoC
-	31 dependencies	23 dependencies

Quelle: https://nodejs.org, https://github.com/expressjs/express, https://github.com/koajs/koa

SEQIS

Konkrete, “kleine” Architekturentscheidungen

node (http)	express	koa
-	215 kB	96,6 kB
-	**20k LoC**	**15k LoC**
-	**31 dependencies**	**23 dependencies**

SEQIS

Konkrete, “kleine” Architekturentscheidungen

dependencies brauchen …

- “Platz”
  - Storage, Memory, Datentransfer
- langfristig Zuwendung
  - aktuell halten
    - Security Issues
    - Änderungen im Lizenzmodell
  - → jedes Update muss getestet werden

… und das gilt für den gesamten dependency-tree!

Quelle: https://npmgraph.js.org/?q=express

Konkrete, “kleine” Architekturentscheidungen

dependencies brauchen

- “Platz”
  - Storage, Memory, Da
- langfristig Zuwendun
  - aktuell halten
    - Security Issues
    - Änderungen im
  - → jedes Update mus

… und das gilt für den g
dependency-tree!

SEQIS

5. Treffen Sie Architektur Entscheidungen bewusst …

- … und nicht “weil wirs immer so machen”!
- Für jede Aufgabenstellung gibt es die passenden Tools
- Falsche Entscheidungen können (fast) nicht mehr behoben werden

Clean Code 4 the win

sprechend
modular
wartbar
strukturiert
verstaendlich
erweiterbar
konsistent
selbterklaerend
clean
lesbar
testbar

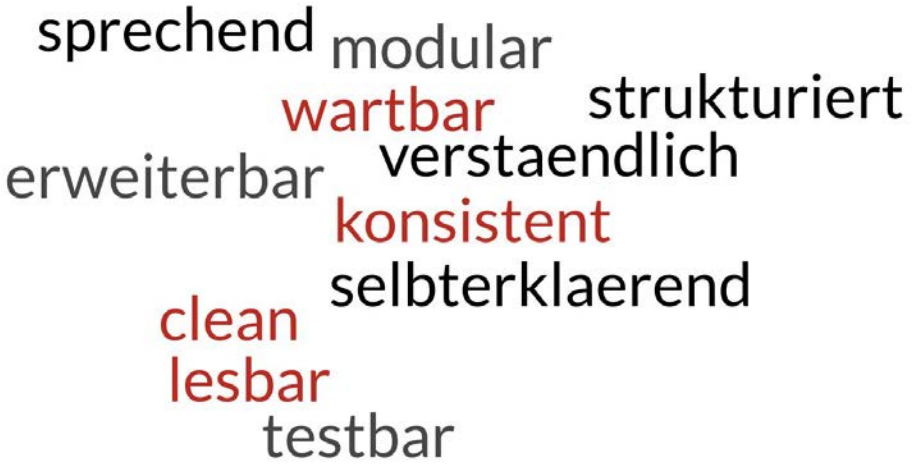
Clean Code 4 the win

jo eh … aber wieso ist das so wichtig?

- rasche Weiterentwicklung
- weniger fehleranfällig
- schnelleres Onboarding

… und wie schaff ich das?

- Code Reviews
- Statische Code Analyse
- Unit Testing
- → TmT @ Youtube: https://t.ly/S9-3D



Unit Testing

- **Wieso**
  - **Einzige sinnvolle Art und Weise langfristig Erweiterungen und Refactoring zu ermöglichen!**
- **Wann**
  - TDD: bei komplexer Business Logic schwierig
  - Test as you Code: Tests treiben oder folgen der Business Logic
  - im Anschluss: am einfachsten, potentiell höchster Refactoring Aufwand
- **Wer**
  - man selbst: sehr effizient
  - ein Entwickler-Kollege: 4-Augen Prinzip, Know how sharing
- **Was noch**
  - Unit Tests testen nicht nur, sie führen zu deutlich besserem Code und Architektur
  - Primär von Entwickler für Entwickler
  - Fail once!
  - Kombinieren mit Code Coverage
  - Müssen durch weitere Test-Arten ergänzt werden

Quelle: https://www.openknowledge.de/blog/die-testpyramide

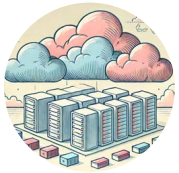
SEQIS

Coding und Unit Testing mit AI

Was (mit Codeium) schon (ganz gut) geht:

- StackOverflow on steroids
- Refactoring der BL
  - z.m. Alternativen
- Generation von Unit Tests
  - für z.B. Legacy Code
- → Q-News @ SEQIS: https://t.ly/LVNAi

SEQIS



6. Ihre Codebasis ist wichtig, behandeln Sie sie entsprechend!

- Clean Code ist nicht nur ein Schlagwort
- Unit Tests sind die wichtigste Grundlage für eine langfristige, zukunftssichere und nachhaltige Entwicklung

Daten …

Data Transfer
(Text, Audio, Video, …)

Daten …

Beispiel Google Suche

- 100.000 Suchanfragen / Sekunde
- Kleines Logo: 4,3 kB
  - 430 MB/s bei falschem Cache Header
- Großes Logo: 171 kb
  - 17,1 GB/s bei Client Resize + falschem Cache Header

Data Transfer
(Text, Audio, Video, …)

SEQIS

Daten …

Reduktion durch:

- Kommunikation verringern
  - Caching richtig einsetzen
    - Browser Cache und Header
    - In-App Cache
  - → je näher am Client desto effizienter
- Compression & Minification
  - JavaScript Minifier verwenden
  - On the fly mit z.B. *gz* komprimieren
- Image Resizing
  - Bilder bereits am Server in den richtigen Größen ablegen
- Content Delivery Network (CDN)

Data Transfer
(Text, Audio, Video, …)

Daten, Daten …

Data Storage
(EMails, Logs, Files)

Daten, Daten …

Reduktion durch:

- Log Rotation und Retention verwenden
- Logging allgemein reduzieren
- richtigen Log-Level setzen
- alte Daten regelmäßig kontrollieren und entfernen …
  - … oder z.m. komprimieren

Data Storage (EMails, Logs, Files)

Daten, Daten und noch mehr Daten …

Dark Data

Daten, Daten und noch mehr Daten …

Auch unbekannte Daten kosten echtes Geld:

- Ressourcenverschwendung
  - Herstellung von Datenträgern
  - Energie für den Betrieb
- DSGVO
  - Benutzerbezogene Daten auf Anfrage löschen
- Aufwendig
  - Migration bei Systemwechsel
- Security
  - Der Verlust von bekannten Daten ist nicht gut, von unbekannten umso mehr!

Dark Data

7. Daten reduzieren, weils eben nicht wurscht ist!

- Spart langfristig Zeit, Geld und Ressourcen
- Unbekannte birgen unbekannte Risiken
- Bereits verfügbare Mechanismen richtig nutzen

Testen, aber richtig!

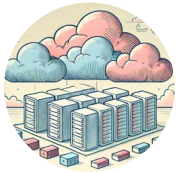
“Wir sind nur noch am Warten der automatisierten Testfälle”

“Wir testen eh schon so viel und trotzdem haben wir Fehler in der Produktion”

“Bei jeder Änderungen müssen wir immer alle Testfälle durchführen, weil …”

“Die Testautomation läuft ewig bis wir verwertbare Ergebnisse bekommen”

Testen, aber richtig!



Business Value & Kritikalität
- Testintensität festlegen
- 4 Augen Prinzip

Akzeptanzkriterien
- Testinhalte ableiten

Test-Ebene
- auf welcher Ebene wird getestet

Developer Insights
- welche Programmteile sind besonders fehleranfällig

Timeboxing & Session Based Testing
- maximale Zeit begrenzen
- Testfokus setzen

Stabilität & Kritikalität
- Entscheidung Testautomation

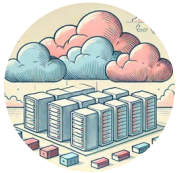
non-functional Requirements
- definieren und testen
- “Nachhaltigkeit” als zusätzliches Kriterium

SEQIS

Testfallerstellung mit AI

AI als “Test-Experten”:

- möglichst genaue Beschreibung der Anforderungen
  - “Hausverstand” hat die AI (noch) keine
  - alles, was nicht definiert ist, führt potentiell zu falschen Ergebnissen
- die Ergebnisse …
  - müssen jedenfalls kontrolliert werden
  - sind ein guter Startpunkt
  - ersparen Schreibarbeit
  - liefern Ideen für zusätzliche Tests

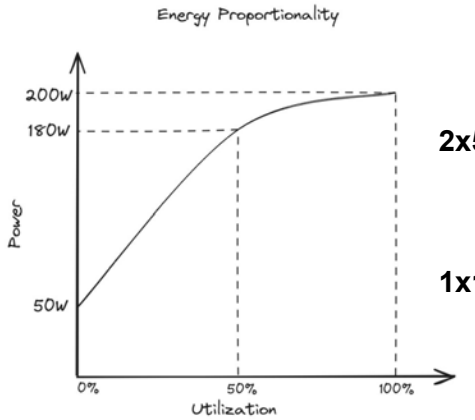


8. Nicht *wie viel* Sie testen, sondern *wie* und *was* ist wichtig!

- Zielgerichtet, effizient & effektiv
- Timeboxing und Session Based Testing als Leitsätze
- Voraussetzung: entsprechendes Requirement Engineering und DoR / DoD

Effizienter und flexibler Betrieb

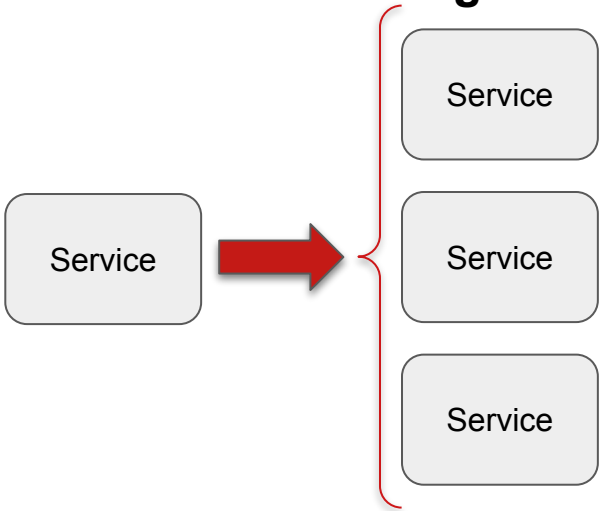
effiziente Energienutzung



2x50% = 360W

1x100% = 200W

flexible Skalierung



Quelle: https://blog.re-cinq.com/posts/cloud-cpu-energy-consumption/

Micro Service Architecture & Container based Hosting

Allgemeine Vorteile von Micro Services:

- unabhängige Entwicklungsteams
- unabhängige Technologien
- leicht skalierbar
- ausfallsicher(er)
  - … wenn ein einzelnes Service ausfällt ist oft nicht die gesamte Applikation betroffen
- → TmT @ Youtube: https://t.ly/fLOp1

Micro Service Architecture

SEQIS

Micro Service Architecture & Container based Hosting

Skalierung & Hosting

- leicht skalierbar
  - nach oben
  - **UND** nach unten!
  - … bis zum kompletten Stillstand
  - regelbasiert und dynamisch
- einfache Verteilung
  - Pod per (physischer) Node
  - bessere Auslastung
- → auch das muss man testen!

Micro Service Architecture

SEQIS

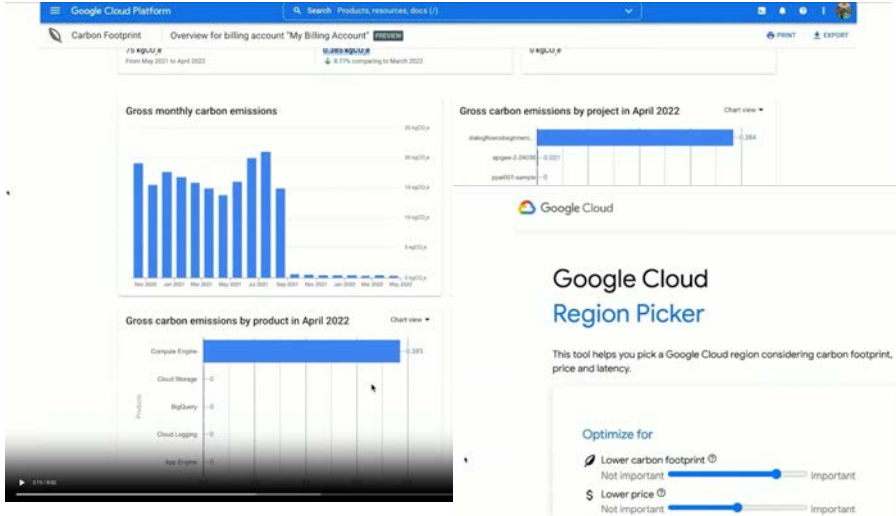
Micro Service Architecture & Container based Hosting

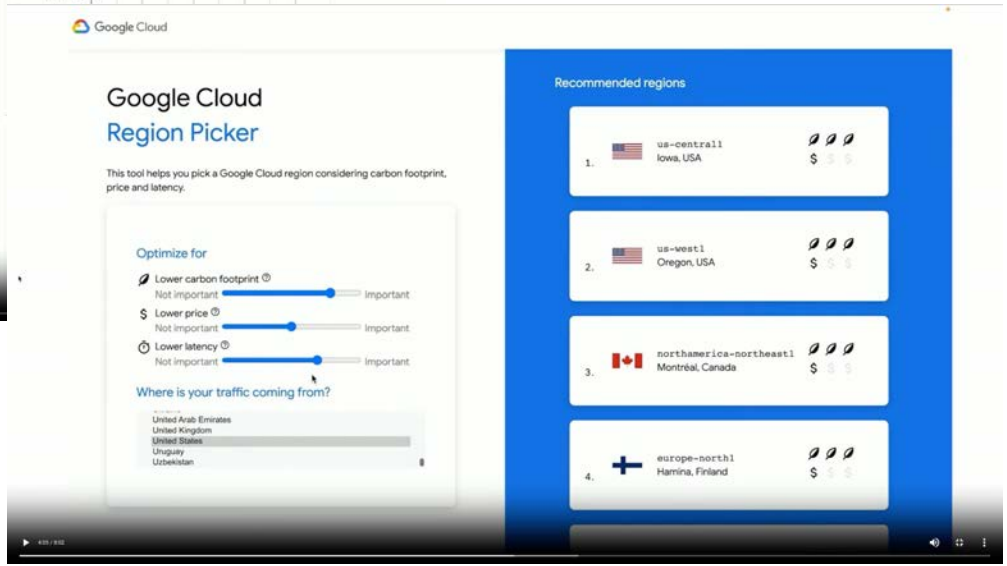
Vorteile der Cloud
gegenüber On Premise Lösungen:

- bessere Ressourcennutzung
- Kostentransparenz
  - CPU, GPU, Memory, Storage, …
- Überschuss Verbrauch
  - Dort auf der Welt hosten, wo gerade Energie grün erzeugt wird
- CO2 reduzierende Tipps und Dashboard @ GCP

Micro Service Architecture

Micro Service Architecture & Container based Hosting





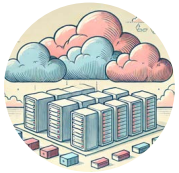
Quelle: https://cloud.google.com

AI in der Cloud

Eigene AI Modelle in der Cloud:

- huggingface.co
  - schnelle und einfache Möglichkeit um AI Modelle zu testen (Prototyping)
- runpod.io
  - umfangreichere Möglichkeiten für Finetuning, UI Anpassungen, …
  - Effiziente Entwicklung mit VSCode Remote Coding

SEQIS



10

9. Container Based Hosting richtig nutzen!

- Dynamische Skalierung und Node affinity einrichten
- Cloud Provider sind effizienter und bieten viele Optimierungsmöglichkeiten
- Kosten transparent machen

Nachhaltigkeit ganzheitlich denken!

Personas definieren & fokussieren

Produktidee & Design

sinnvolle Reduktion, Standardeinstellungen & Use Cases

Meetings strukturieren

Prozesse optimieren

Umsetzungsprozess

bewusst Architekturentscheidungen treffen

Daten (storage & transfer) reduzieren

Clean Code

das Richtige richtig testen

Micro Services & CBH

Umsetzung & Betrieb

Zum Schluss die gute Nachricht:

Jeder von uns kann den Status Quo verbessern!

SEQIS

10. Jetzt mit Nachhaltigkeit anfangen!

- Nachhaltigkeit in der IT ist für alle ein Gewinn!
- In der IT können wir nachhaltig agieren, ohne Nachteile und ohne Einschränkungen, ganz im Gegenteil!
- … weil Nachhaltigkeit kann uns nicht wurscht sein!