Caching verstehen

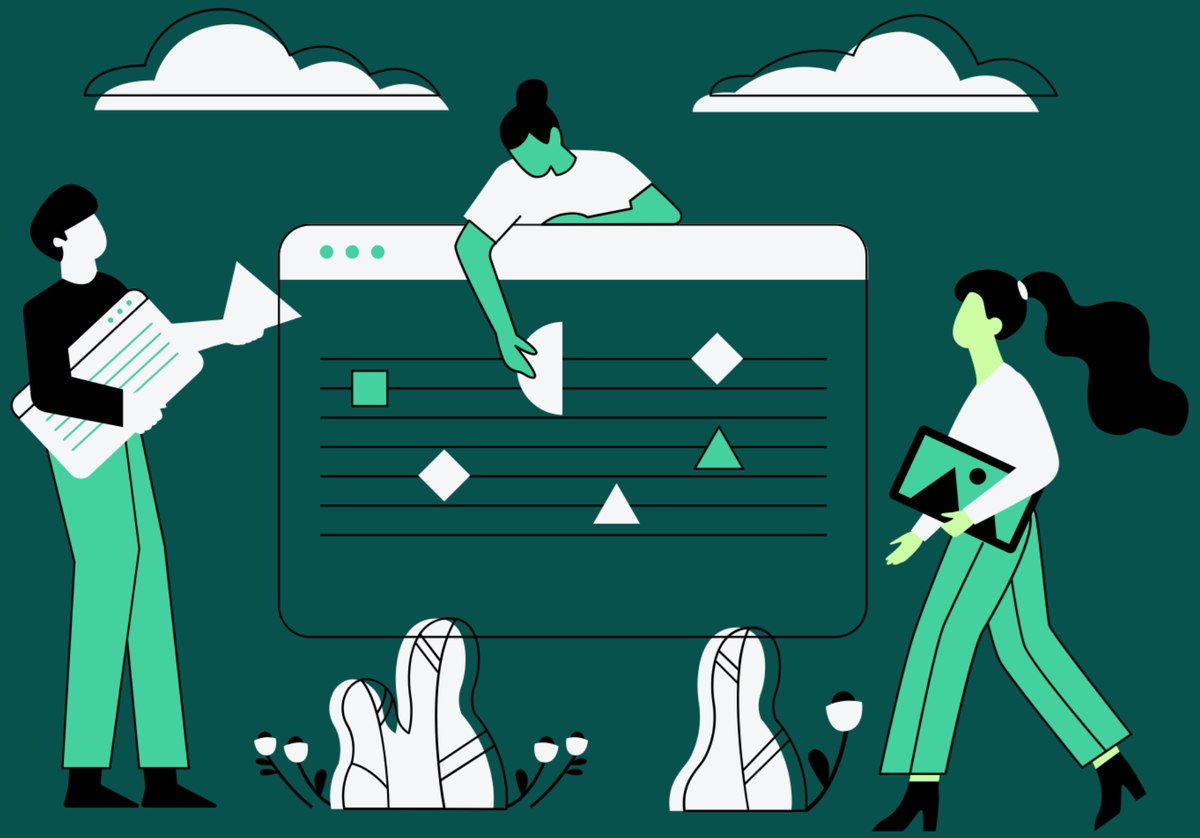
Schneller, effizienter, sicher

Torsten Steinmüller

16. Januar 2026

Kamal Badawi – 5236028

Agenda

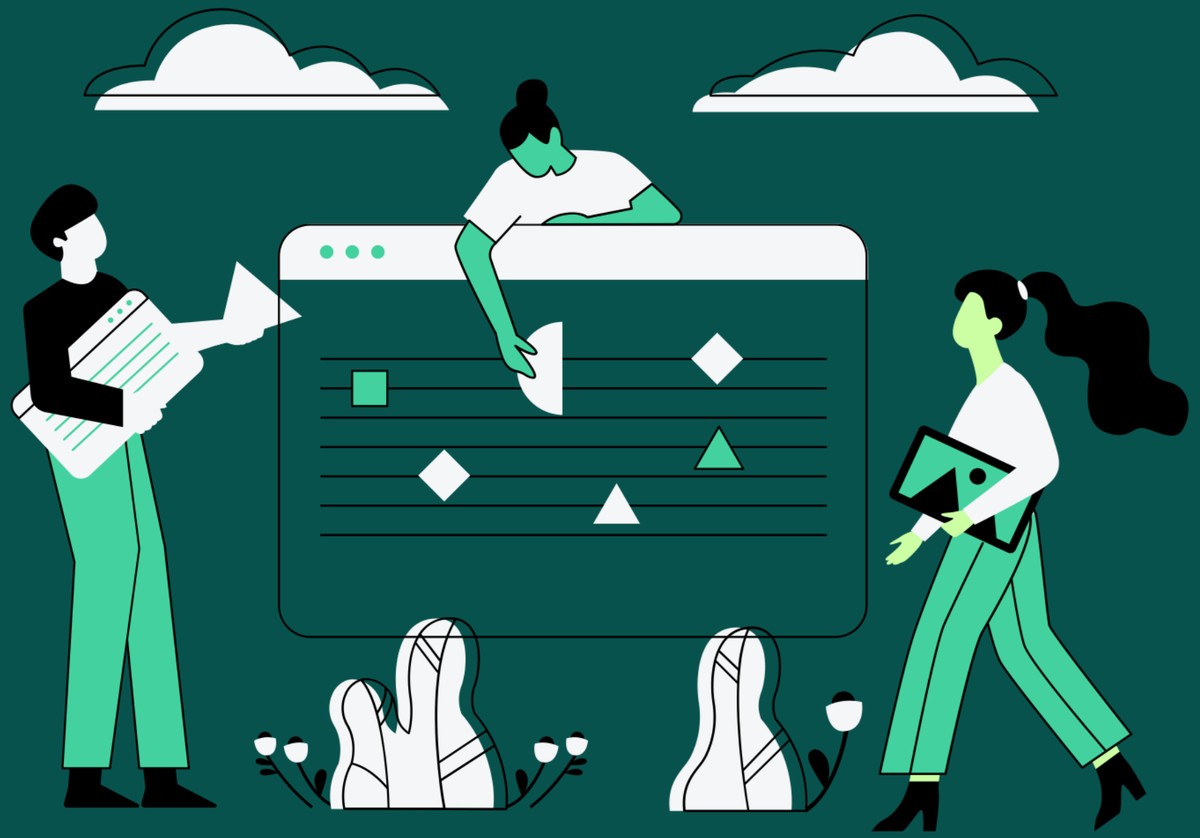


01 Einleitung

02 Ziele des Cachings

03 Architekturebenen

Agenda

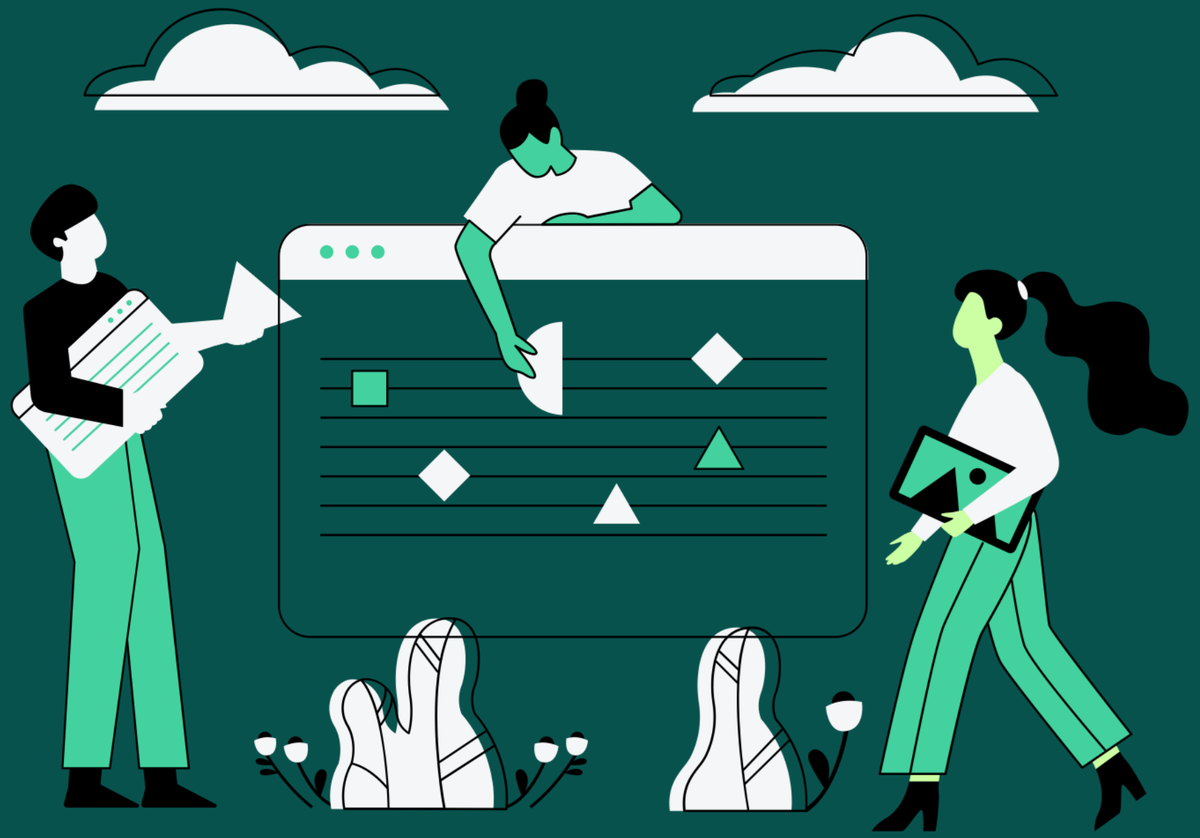


04 Herausforderungen

05 Best Practices

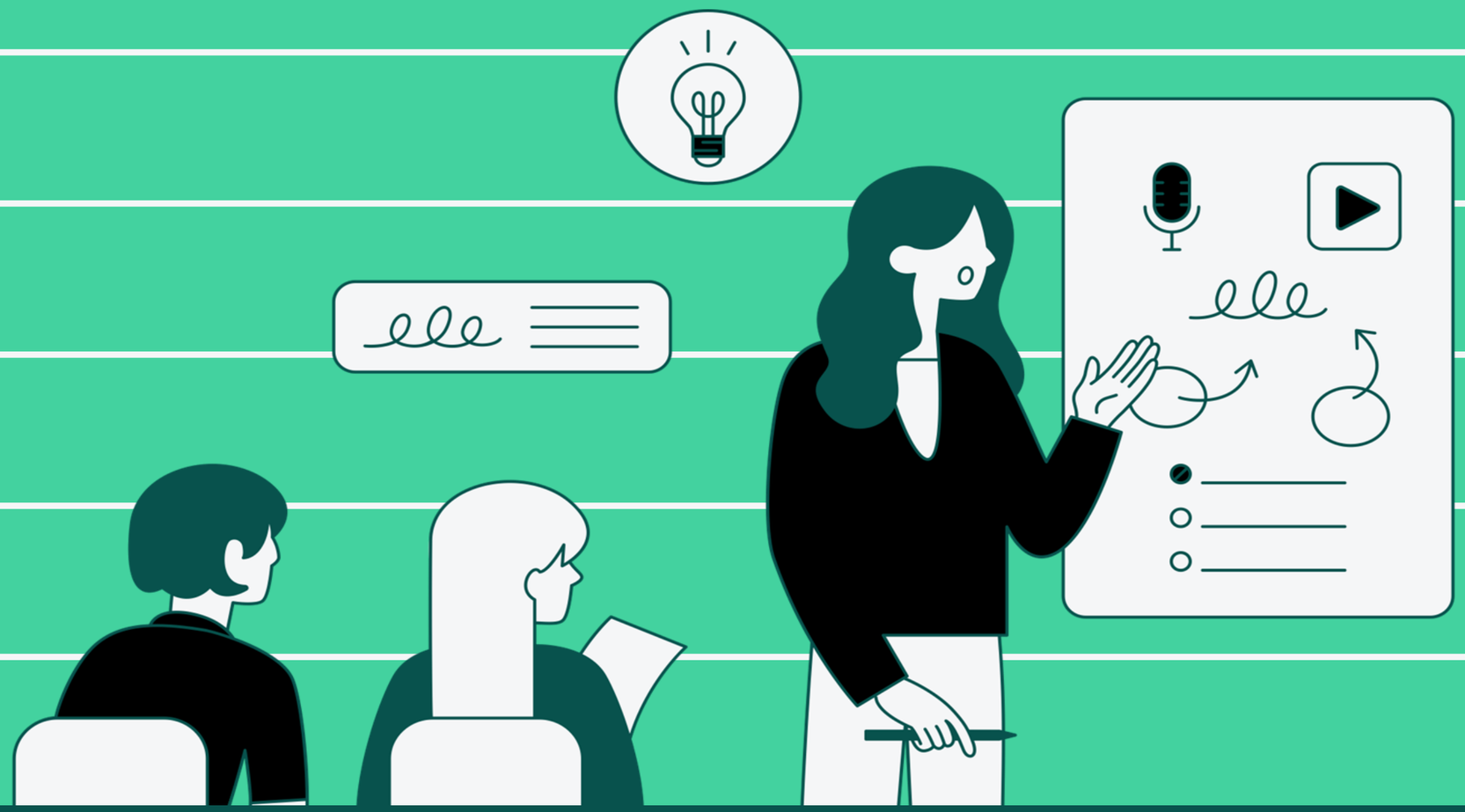
06 Demo

Agenda



07 Fazit

08 Quiz



01 Einleitung

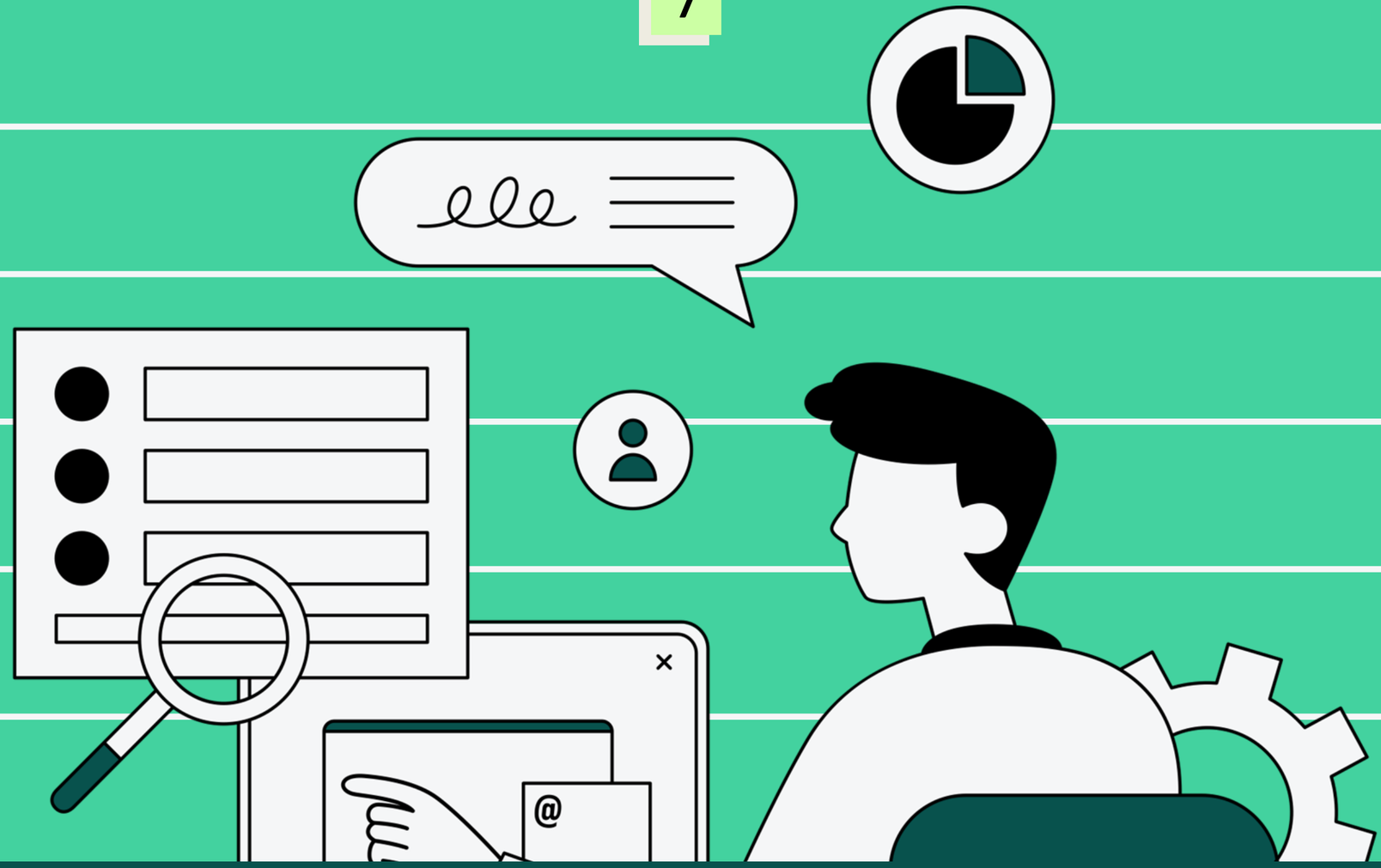
Definition – Analogie

Definition

Temporäre Zwischenspeicherung ressourcenintensiver Ergebnisse

Analogie

Regionallagerprinzip: schneller zum Kunden, weniger Belastung für die Zentrale – so funktioniert Caching.



02 Ziele des Cachings

Nutzen & Wirkung

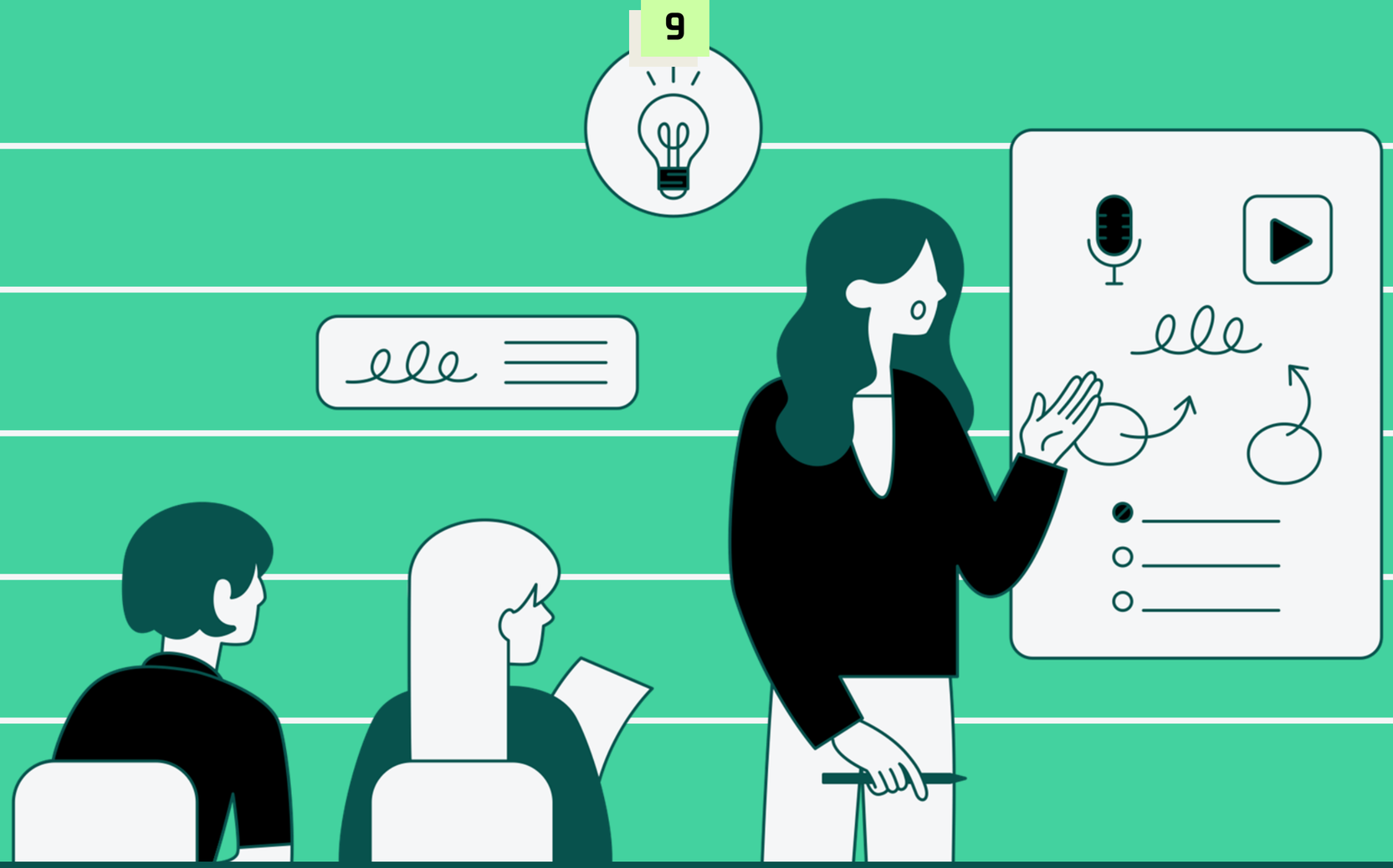
Performance steigern: Daten schneller verfügbar machen, Zugriffszeiten minimieren

Backend entlasten: Weniger direkte Anfragen an Server und Datenbanken

Systemstabilität erhöhen: Vermeidung von Überlastung und Ausfällen

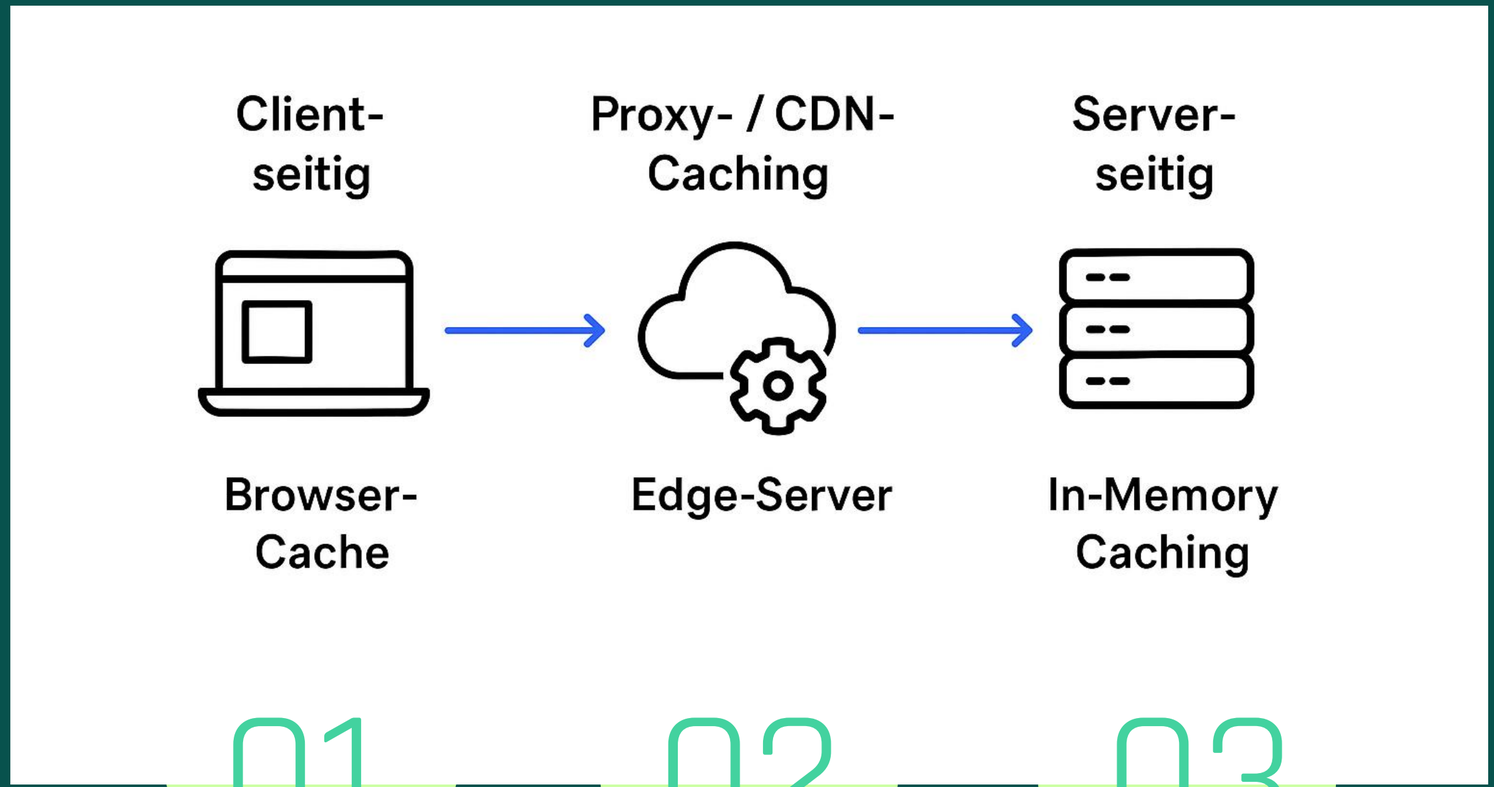
Skalierbarkeit unterstützen: Systeme können mehr Nutzer oder Daten effizient verarbeiten

Benutzerzufriedenheit verbessern: Schnelle Reaktionen erhöhen die Nutzererfahrung



03 Architekturebenen

Architekturebenen des Cachings



Client-Side Caching

Speicherort

Lokal auf Endgerät

Vorteile

- Sehr schnelle Ladezeiten
- Keine erneute Serveranfrage notwendig

Anwendungsfall

Statische Ressourcen (CSS, JS, Bilder, Fonts)

Steuerung

HTTP-Header (z. B. Cache-Control: max-age=60)

Proxy / CDN-Caching

Speicherort

Geografisch verteilte Edge-Server eines CDN-Anbieters

Vorteile

- Reduzierte Netzwerklatenz
- Deutliche Entlastung des Origin-Servers

Anwendungsfall

Globale Auslieferung statischer oder semi-statischer Inhalte

Steuerung

CDN- und Cache-Regeln (TTL, Cache Keys, Invalidation)

Server-seitiges Caching

Speicherort

Arbeitsspeicher von Applikationsservern oder In-Memory-Stores (z. B. Redis, Memcached)

Vorteile

- Maximale Kontrolle
- Hohe Performance
- Gezielte Aktualisierung möglich

Anwendungsfall

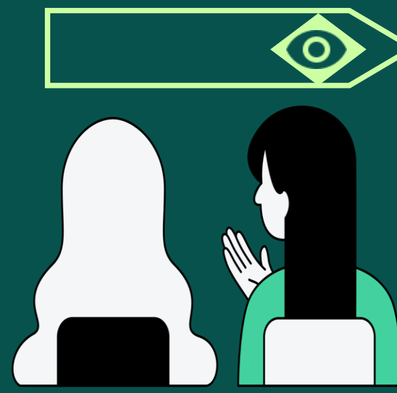
Zwischenspeicherung komplexer oder häufig ausgeführter Berechnungen (z. B. Datenbankabfragen, gerenderte Templates)

Steuerung

Direkte Kontrolle durch die Anwendungslogik (TTL, explizite Invalidierung)

Vergleich

Kriterium	Client-seitiges Caching (Browser)	Proxy / CDN-Caching	Server-seitiges Caching
Speicherort	Lokal auf dem Endgerät des Nutzers	Geografisch verteilte Edge-Server	Arbeitsspeicher von Applikationsservern oder In-Memory Stores
Zweck	Zwischenspeicherung statischer Ressourcen (CSS, JS, Bilder, Fonts)	Globale Auslieferung statischer oder semi-statischer Inhalte	Zwischenspeicherung komplexer oder häufiger Berechnungen
Typische Inhalte	Frontend-Ressourcen, UI-Dateien	Medien, Downloads, öffentliche Inhalte	Datenbankabfragen, Berechnungsergebnisse
Vorteil	Sehr schnelle Ladezeiten, keine Serveranfrage nötig	Reduzierte Netzwerklatenz, Entlastung des Origin-Servers	Maximale Kontrolle, gezielte Invalidierung möglich
Kontrolle	Gering (über HTTP-Header)	Mittel (über CDN-Konfiguration)	Hoch (direkt in der Anwendungslogik)
Nähe zum Nutzer	Sehr hoch	Hoch	Gering
Risiko bei Fehlkonfiguration	Veraltete Inhalte (Stale Data)	Datenschutzprobleme (Data Leakage)	Inkonsistente Daten



04 Herausforderungen

Konsistenzprobleme (Stale Data)

Problem:
Der Cache liefert veraltete Daten, obwohl die Originaldaten bereits aktualisiert wurden

Ursachen:
- Zu lange Cache-Laufzeiten
- Fehlende oder falsche Invalidierungsstrategie

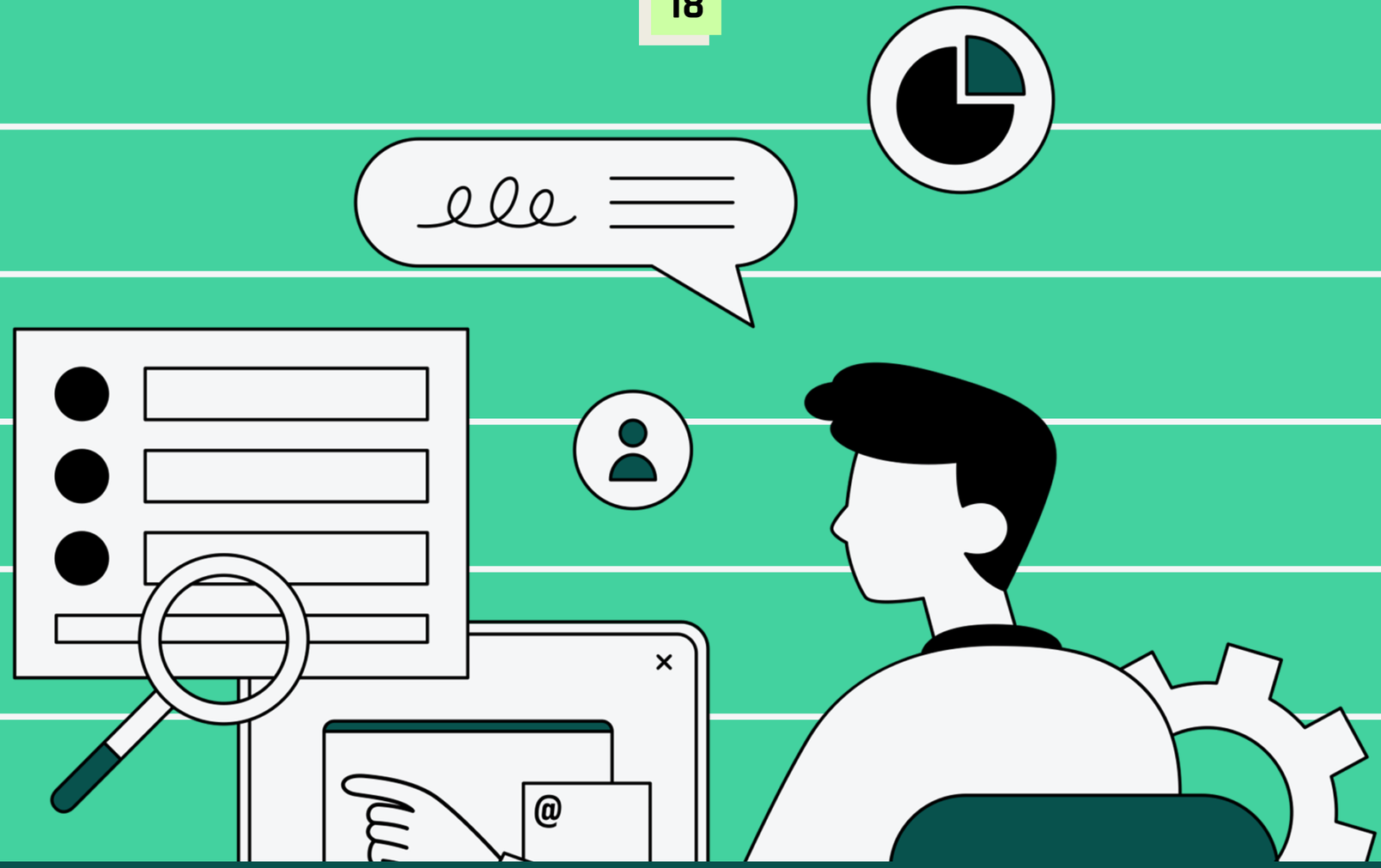
Lösungen:
- Time-to-Live (TTL): Automatische Ablaufzeit für Cache-Einträge
- Ereignisgesteuerte Invalidierung: Cache wird bei Datenänderungen gezielt gelöscht oder aktualisiert

Datenschutz & Sicherheit (Data Leakage)

Problem:
Sensible oder nutzerspezifische Daten werden in öffentlichen Caches gespeichert

Risiko:
Unbefugte Nutzer erhalten Zugriff auf fremde Inhalte

Lösungen:
- Cache-Control: private: Speicherung nur im Browser-Cache erlauben
- Vary Header: Trennung von Cache-Versionen nach Cookie oder Authorization



05 Best Practices

Planung & Datenkonsistenz

Cache-Strategie sorgfältig planen:

- Klare Entscheidung, was, wo und wie lange gecacht wird
- Unterschiedliche Inhalte erfordern unterschiedliche Cache-Zeiten

Automatisierte Invalidierung implementieren:

- Veraltete Daten müssen zuverlässig entfernt werden
- Einsatz von Time-to-Live (TTL) oder ereignisgesteuerten Mechanismen

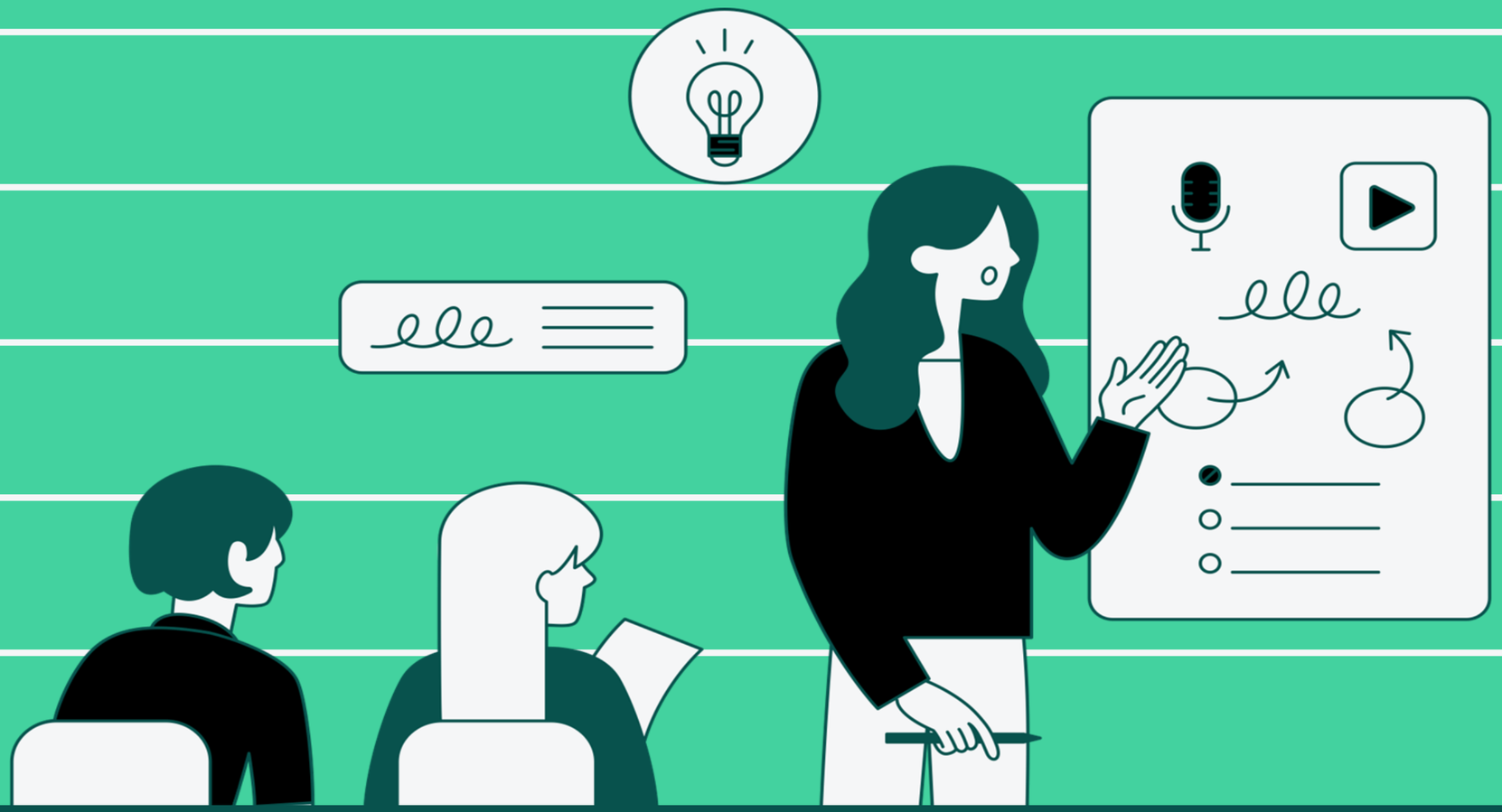
Sicherheit & Monitoring

Sensible Daten absichern:

- Keine personenbezogenen oder sessionbezogenen Daten in öffentlichen Caches
- Nutzung von Cache-Control: private und Vary Headern

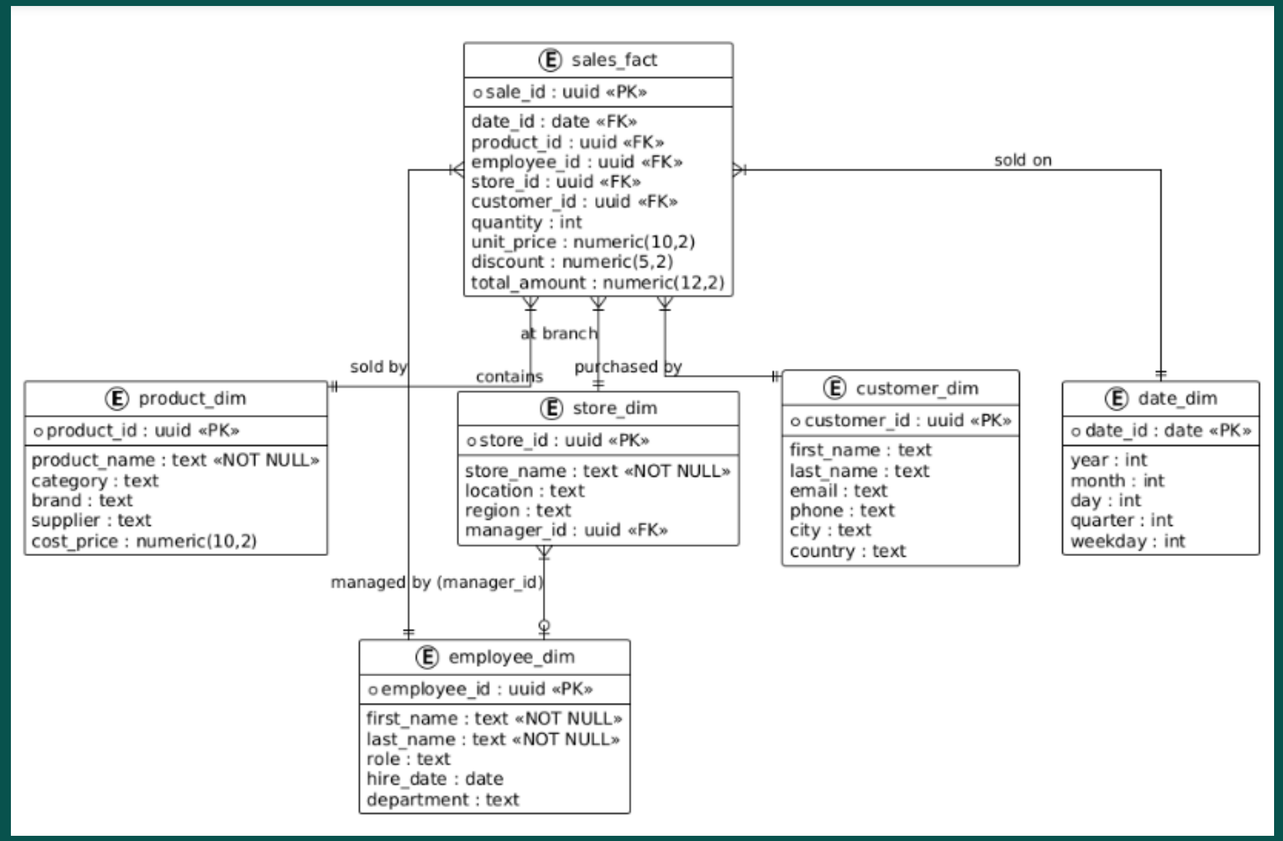
Monitoring & Performance-Analyse:

- Überwachung von Cache-Hit- und Cache-Miss-Raten
- Analyse der Performance-Auswirkungen
- Frühzeitiges Erkennen von Fehlkonfigurationen

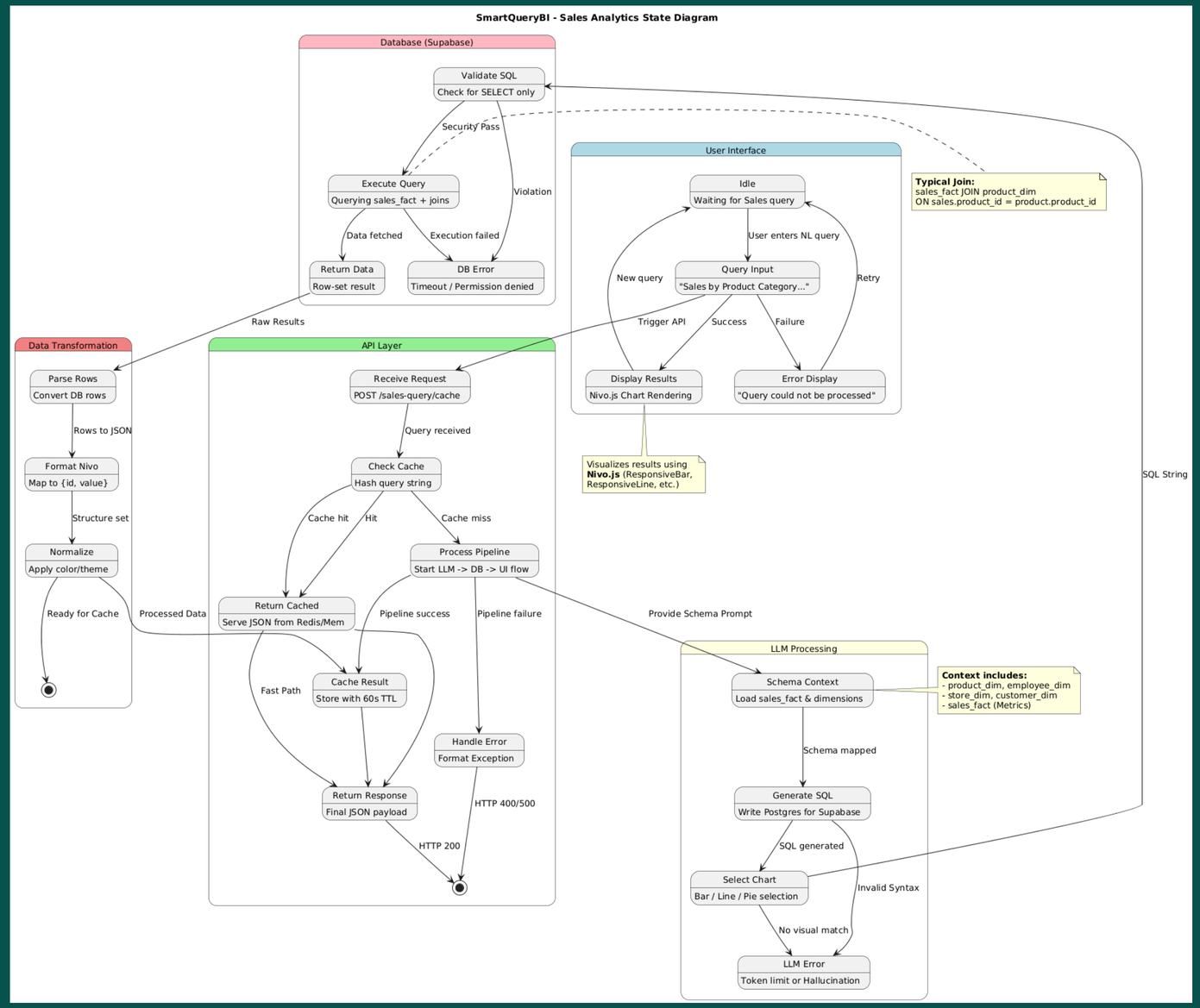


06 Demo

ER-Modell


sales_fact
sale_id : uuid «PK»
date_id : date «FK»
product_id : uuid «FK»
employee_id : uuid «FK»
store_id : uuid «FK»
customer_id : uuid «FK»
quantity : int
unit_price : numeric(10,2)
discount : numeric(5,2)
total_amount : numeric(12,2)
sold on
at branch
sold by
contains
purchased by
product_dim
product_id : uuid «PK»
product_name : text «NOT NULL»
category : text
brand : text
supplier : text
cost_price : numeric(10,2)
store_dim
store_id : uuid «PK»
store_name : text «NOT NULL»
location : text
region : text
manager_id : uuid «FK»
customer_dim
customer_id : uuid «PK»
first_name : text
last_name : text
email : text
phone : text
city : text
country : text
date_dim
date_id : date «PK»
year : int
month : int
day : int
quarter : int
weekday : int
managed by (manager_id)
employee_dim
employee_id : uuid «PK»
first_name : text «NOT NULL»
last_name : text «NOT NULL»
role : text
hire_date : date
department : text

Zustandsdiagramm



07 Fazit

Wichtige Erkenntnisse

Zentrale Bedeutung:

Caching ist ein wesentliches Mittel, um die Performance moderner Systeme zu steigern.

Mehrstufiges Caching:

- Reduziert Latenzen
- Entlastet Backend-Systeme
- Jede Cache-Ebene hat eine klar definierte Rolle im Gesamtsystem

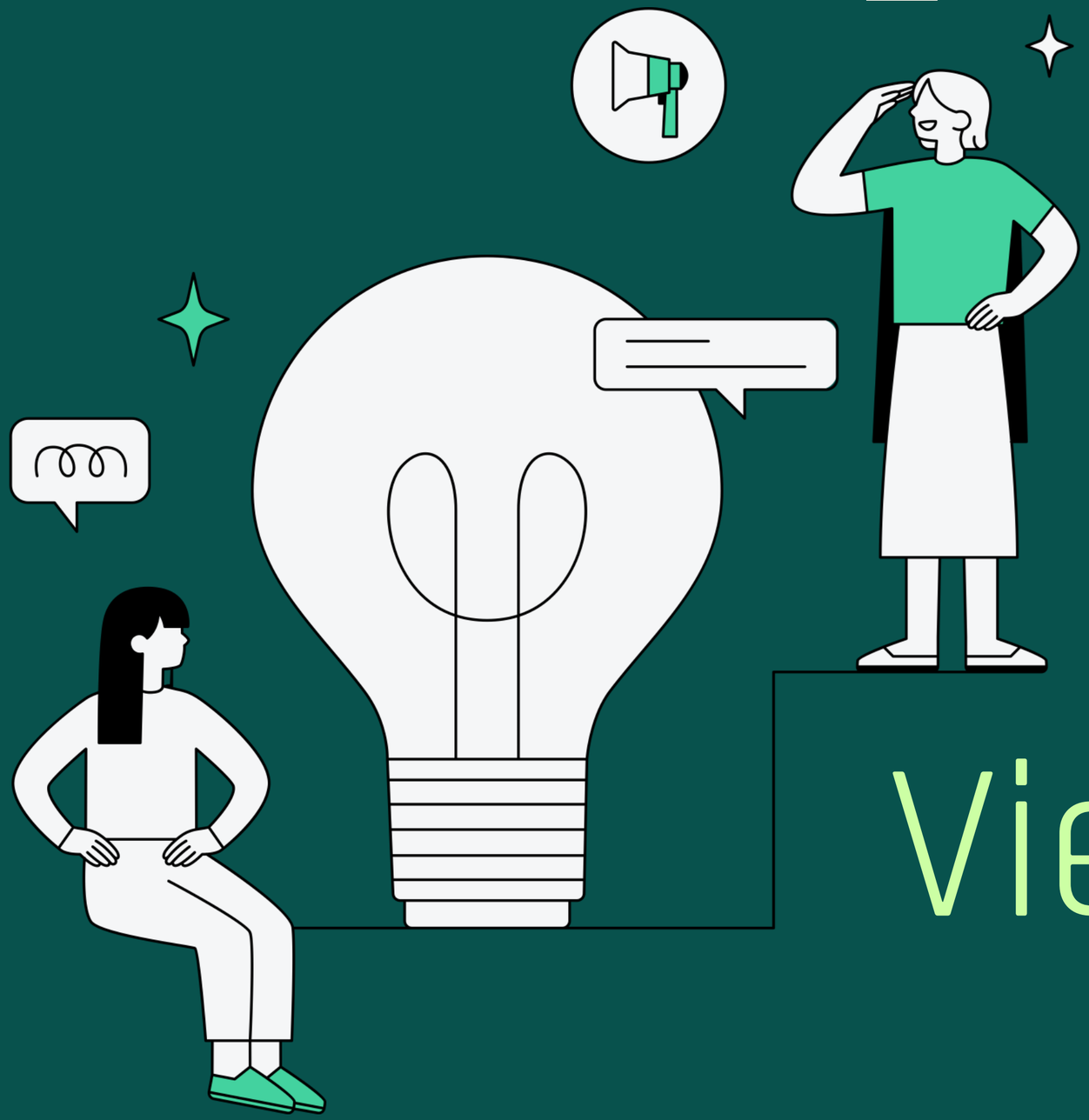
Erfolgsfaktoren:

- Durchdachte Invalidierungsstrategien
- Sorgfältiger Umgang mit sensiblen Daten

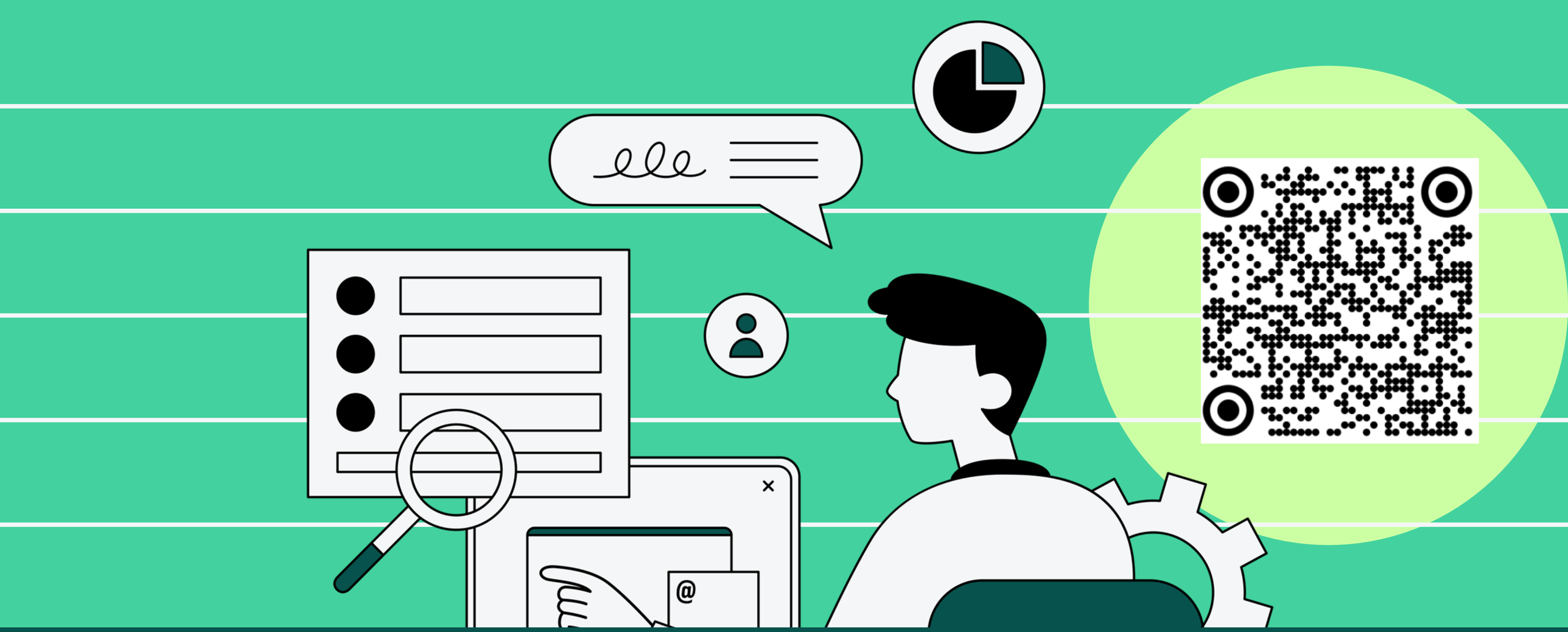
Nutzen bei richtiger Umsetzung:

- Höhere Benutzerzufriedenheit
- Verbesserte Skalierbarkeit
- Erhöhte Systemstabilität

08 Quiz



Vielen Dank



GitHub