Eine für viele

Multi-Tenancy-Software

Sascha Groß
sascha.gross@mathema.de

Agenda

- Warum?
- Was?
- Wie?

Meine Berührungspunkte mit mandantenfähiger Software

- Kreditvergabesystem einer Bank - 4,5 Jahre
- Versicherungsvertrieb - 2 Jahre
- Datenaustauschmiddleware für Bankdienstleister - $\frac{3}{4}$ Jahre
- Unternehmensportal - 7 Jahre
- Backend für Banking für Bankdienstleister - 1 Jahr
- Datenaustauschsystem für CO2 Fußabdrücke - 1 Jahr

- **ca. $\frac{2}{3}$ meines IT-Lebens**

Definition Mandantenfähigkeit

https://de.wikipedia.org/wiki/Mandantenfähigkeit

Als mandantenfähig (auch mandantentauglich) wird Informationstechnik bezeichnet, die auf demselben Server oder demselben Software-System mehrere Mandanten, also Kunden oder Auftraggeber, bedienen kann, **ohne dass diese gegenseitigen Einblick in ihre Daten**, Benutzerverwaltung und Ähnliches haben. Ein IT-System, das dieser Eigenschaft genügt, bietet die Möglichkeit der disjunkten, mandantenorientierten Datenhaltung, Präsentation (GUI) und Konfiguration (Customizing). Jeder Kunde **kann nur seine Daten sehen und ändern**.

Ein System wird nicht mandantenfähig, indem man für jeden Mandanten eine eigene Instanz (Kopie) des Systems erstellt.

Wer kann ein Mandant (Tenant) sein?

- Unternehmen
- Niederlassung
- Abteilung
- Organisationseinheit
- Behörde
- …

Keine Mandanten sind

- Benutzer
- Benutzergruppen

Warum Mehrmandantenfähige Software?

- Wiederverwendung von Software & Hardwareresourcen
- Minimierung Betriebskosten
- Minimierung Wartungsaufwand
- Minimierung Erstellungskosten

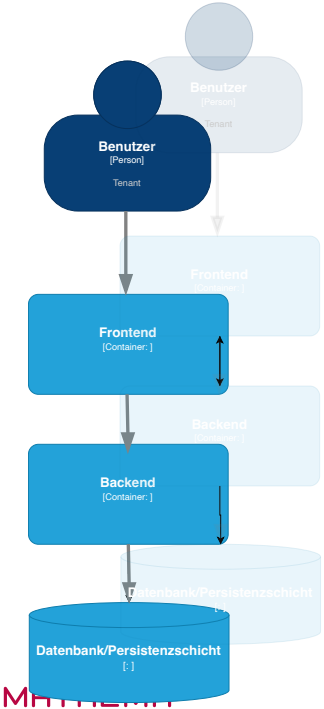
- **Wirtschaftliche und organisatorische Gründe**

Scheinbare Nachteile

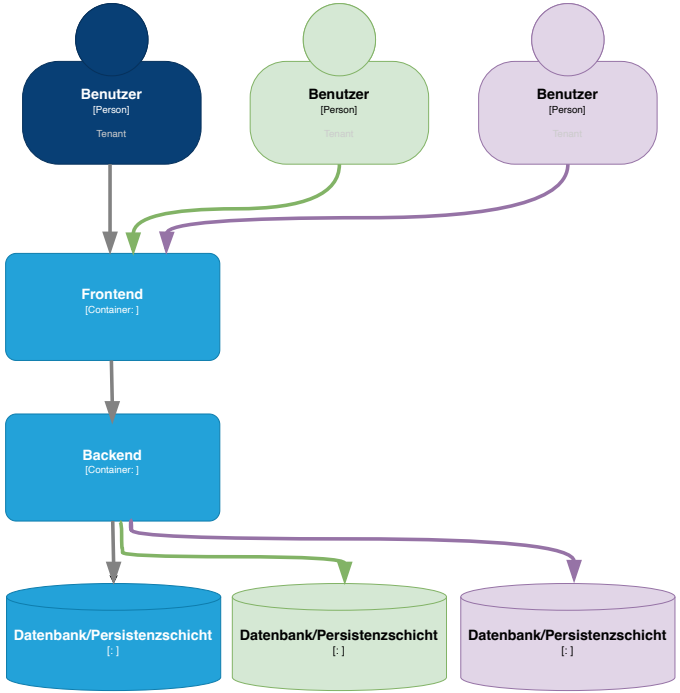
- Feature für einzelen Mandanten
  - Lösung: Featuretoggle & Konfiguration
- Gleicher Softwarestand bei allen Mandanten
- Bugs bei allen Mandanten
- Ausfall bei allen Mandanten

Architekturen

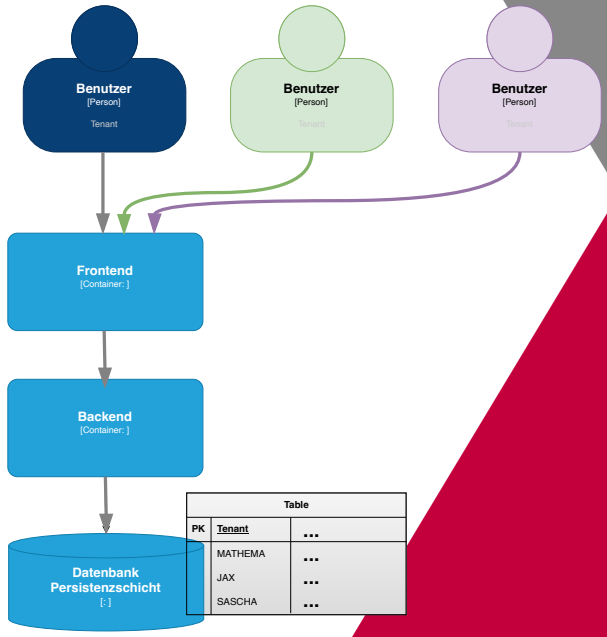
Silo model



Bridge model



Pool model



Silo model

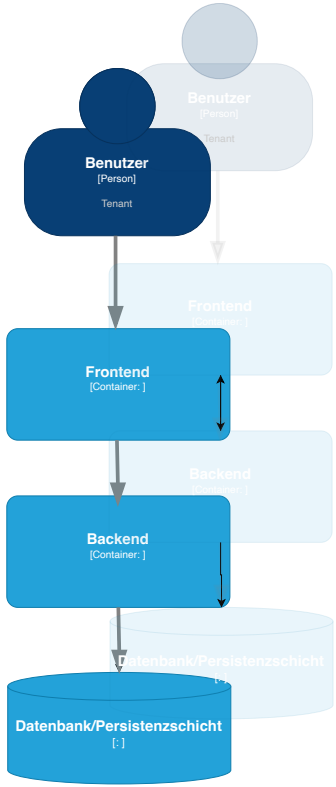
Getrennte Systeme pro Mandant

Vorteile

- Nur ein Mandant, also *normale* Software
- Strikte Trennung von Daten (rechtliche Aspekte)
- Jeder Mandant hat eigene Instanzen
- Hohe Verfügbarkeit, da nur ein Mandant

Nachteile

- *Copy & Paste* der Infrastruktur
- Verwaltungsaufwand/Überwachung der Resourcen

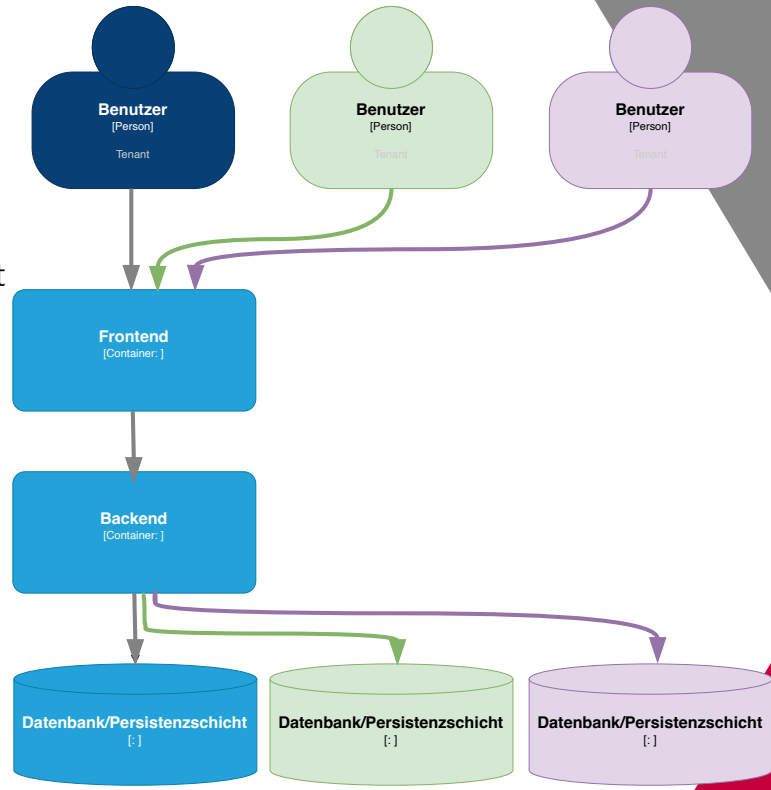


Bridge model

- Gemeinsame Benutzung aller Ressourcen
- Gemeinsame Benutzung der Applikationsschicht
- Gemeinsame Benutzung der Datenhaltungsschicht
- Trennung der Datenhaltung über getrennte Datasoures/Datenbankschemata (in der gleichen Datenbank)

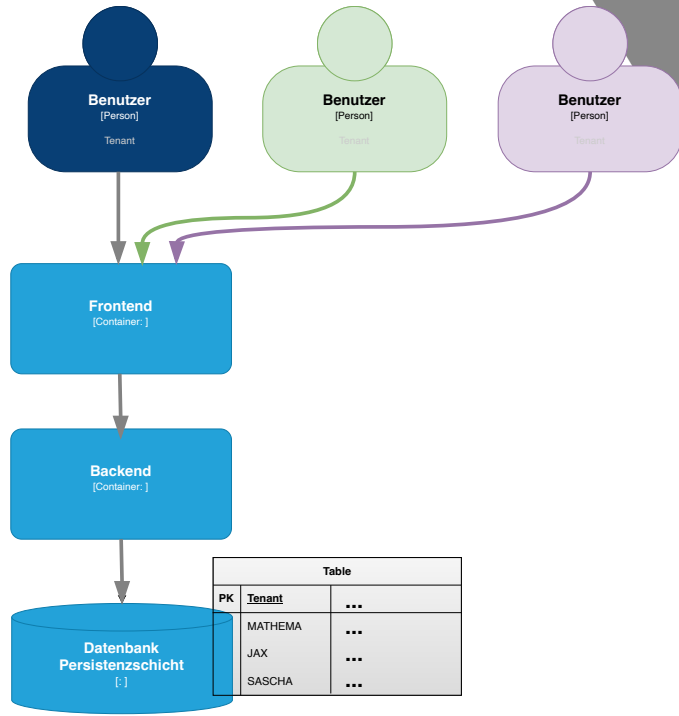
Nachteile:

- Getrennte Datenbankschemata und somit erhöhte Anzahl von Verbindungen



Pool model

- Gemeinsame Benutzung der Infrastructure Ressourcen
- Gemeinsame Benutzung des Applikationsschicht
- Trennung der Datenhaltung über Diskriminator Spalte in Datenbank



Technische Umsetzung - Mandantenerkennung

- HTTP Header
- Query Parameter `http://mein-super-service.de/super/?tenant= < tenant >`
- Url - Subdomain `http://< tenant >.mein-super-service.de`
- JSON Web Token (JWT)
- …

Datenbank

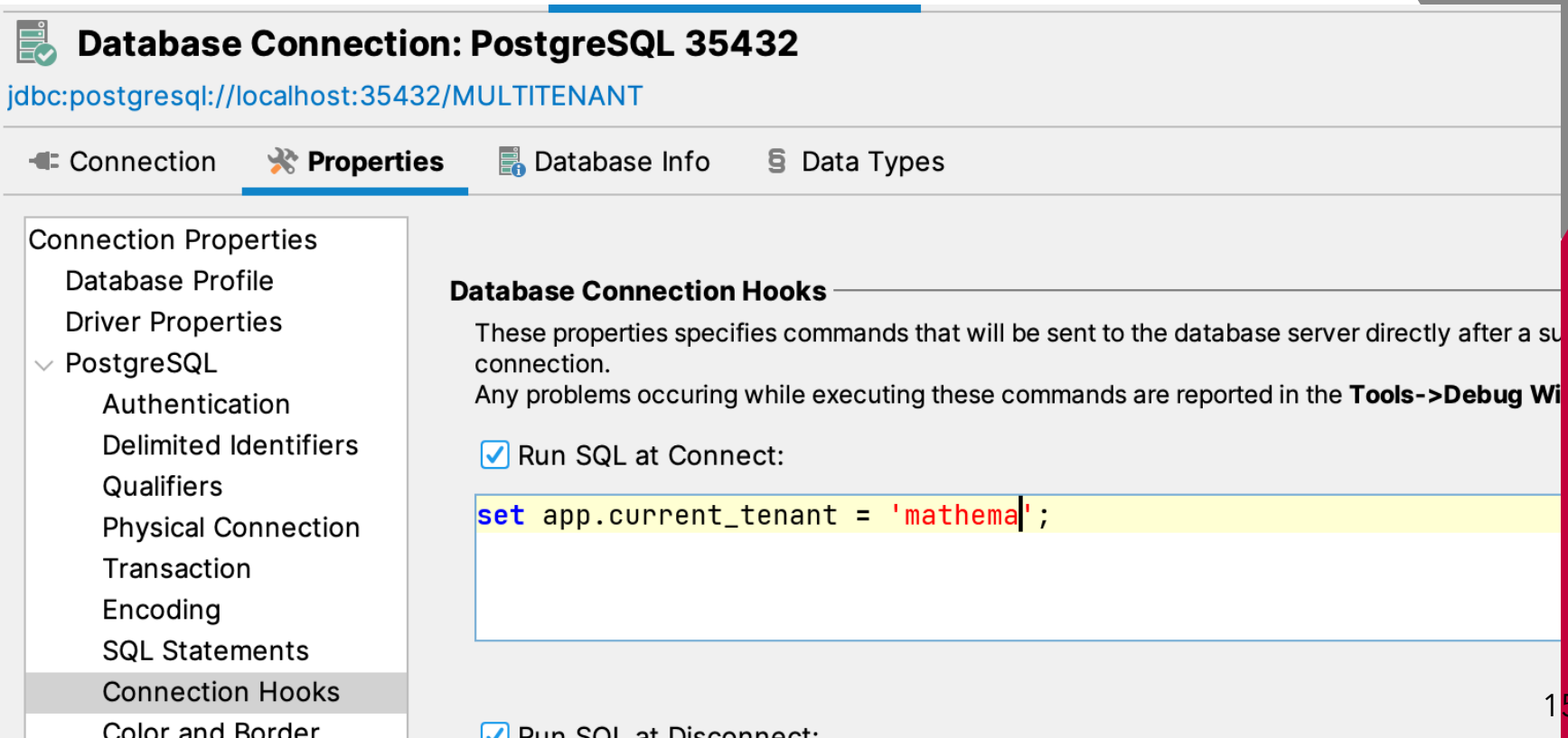
- Spalte ***tenant*** in Tabelle
- Row level Security (RLS)
  - Erlaubt Zugriff auf Zeile nur, wenn konfigurierte Bedingung erfüllt ist

Datenbank - RLS - Postgres

```
1  CREATE TABLE IF NOT EXISTS asset
2  (
3  id     BIGINT NOT NULL,
4  tenant VARCHAR(255) NOT NULL,
5   ...
6  PRIMARY KEY (id, tenant)
7  );
8
9  ALTER TABLE ONLY asset FORCE ROW LEVEL SECURITY;
10 ALTER TABLE asset ENABLE ROW LEVEL SECURITY;--do not forget to enable it
11
12 CREATE POLICY tenant_isolation_policy ON
13   asset USING (((tenant)::text = current_setting('app.current_tenant'::text)));
```

Datenbankzugriff

```
1   set app.current_tenant = '< tenant >';
```



Datenbank - RLS - Applikationsseitig

```
1   public Connection getConnection() throws SQLException {
2       Connection connection = super.getConnection();
3       try (Statement sql = connection.createStatement()) {
4          sql.execute("SET app.current_tenant = '" + tenantService.getCurrentTenant() + "'");
5       }
6       return connection;
7   }
```

Java - TenantService

```java
public class TenantService {
    ThreadLocal<String> TLS_TENANT = new InheritableThreadLocal<>();

    void setCurrentTenant(String tenant) {
        TLS_TENANT.set(tenant);
    }
    void remove() {
        TLS_TENANT.remove();
    }

    public String getCurrentTenant() {
        return TLS_TENANT.get();
    }
}
```

MATHEMA

Java - TenantFilter - JaxRS

```java
@Provider
public class TenantFilter implements ContainerRequestFilter {

    @Inject
    private TenantService tenantService;

    @Override
    public void filter(ContainerRequestContext requestContext) {
        var headers = requestContext.getHeaders();
        var tenantList = headers.get("tenant");
        if (tenantList == null) {
            tenantList = Collections.emptyList();
        }

        var tenant = tenantList.stream().findFirst();

        if (tenant.isPresent()) {
            tenantService.setCurrentTenant(tenant.get());
        } else {
            tenantService.setCurrentTenant("default");
        }
    }
}
```

Batch - Dunkelverarbeitung

```
1       private static final List<String> TENANTS =
2           List.of("mathema", "campus", "jax", "sascha");
3
4       public void start() {
5           TENANTS.forEach(this::doIt);
6       }
7
8       private void doIt(String tenant) {
9           try {
10              tenantService.setCurrentTenant(tenant);
11
12              doIt();
13          } finally {
14              tenantService.remove();
15          }
16      }
17
18      private void doIt() {
19          ...
20      }
```

Annotation bei JPA (EclipseLink)

```
1  @Entity
2  @org.eclipse.persistence.annotations.Multitenant
3          (value = SINGLE_TABLE | TABLE_PER_TENANT)
4  @org.eclipse.persistence.annotations.TenantDiscriminatorColumn(name = "tenant")
5  public class ...
```

Annotation bei JPA (Hibernate)

```
@Entity
public class ...

    @org.hibernate.annotations.TenantId
    private String tenant
```

```
interface org.hibernate.context.spi.CurrentTenantIdentifierResolver<T> {
    T    resolveCurrentTenantIdentifier()}
     ...
```

MATHEMA

MATHEMA CAMPUS

Eine für viele - Multi-Tenancy-Software

Dankeschön / Fragen?

https://github.com/saschagr/multitenant-service

Sascha Groß
sascha.gross@mathema.de