Interaktive web-basierte Visualisierungen für große netzwerkartige Daten

Daniel Graf Hoyos, 29. September 2014 München

Software Engineering für betriebliche Informationssysteme (sebis)
Fakultät für Informatik
Technische Universität München

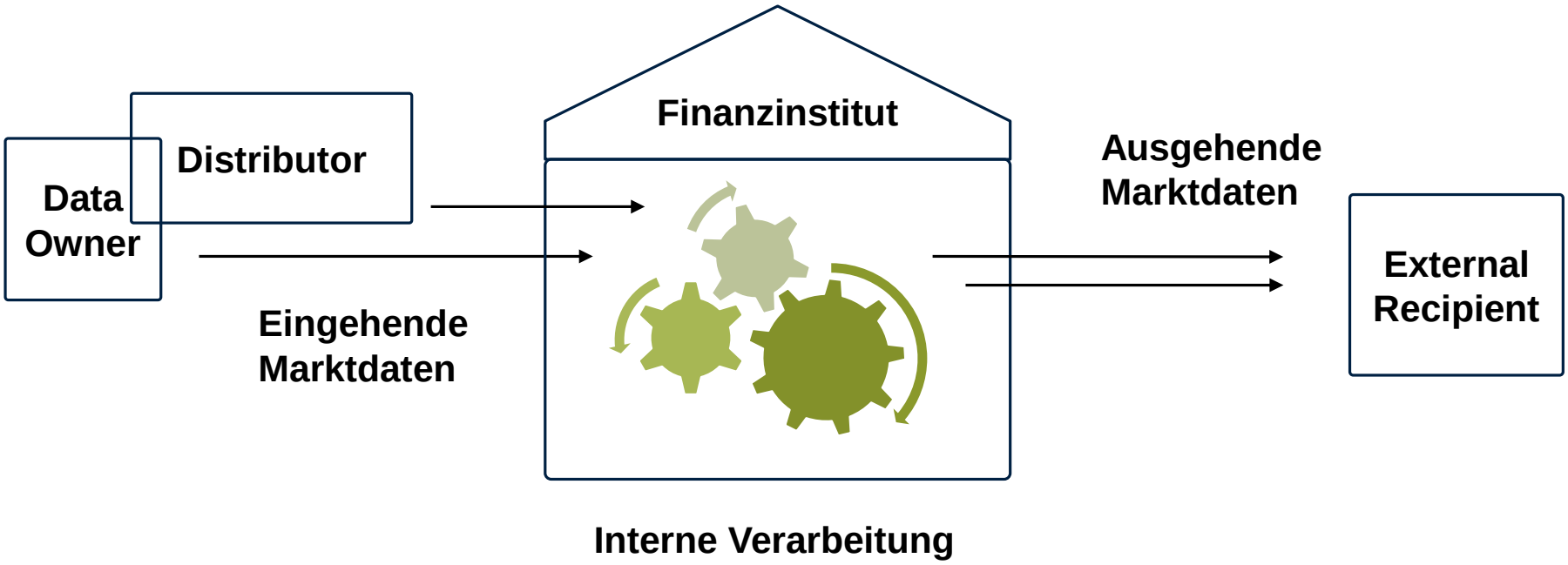
wwwmatthes.in.tum.de

Überblick

sebis

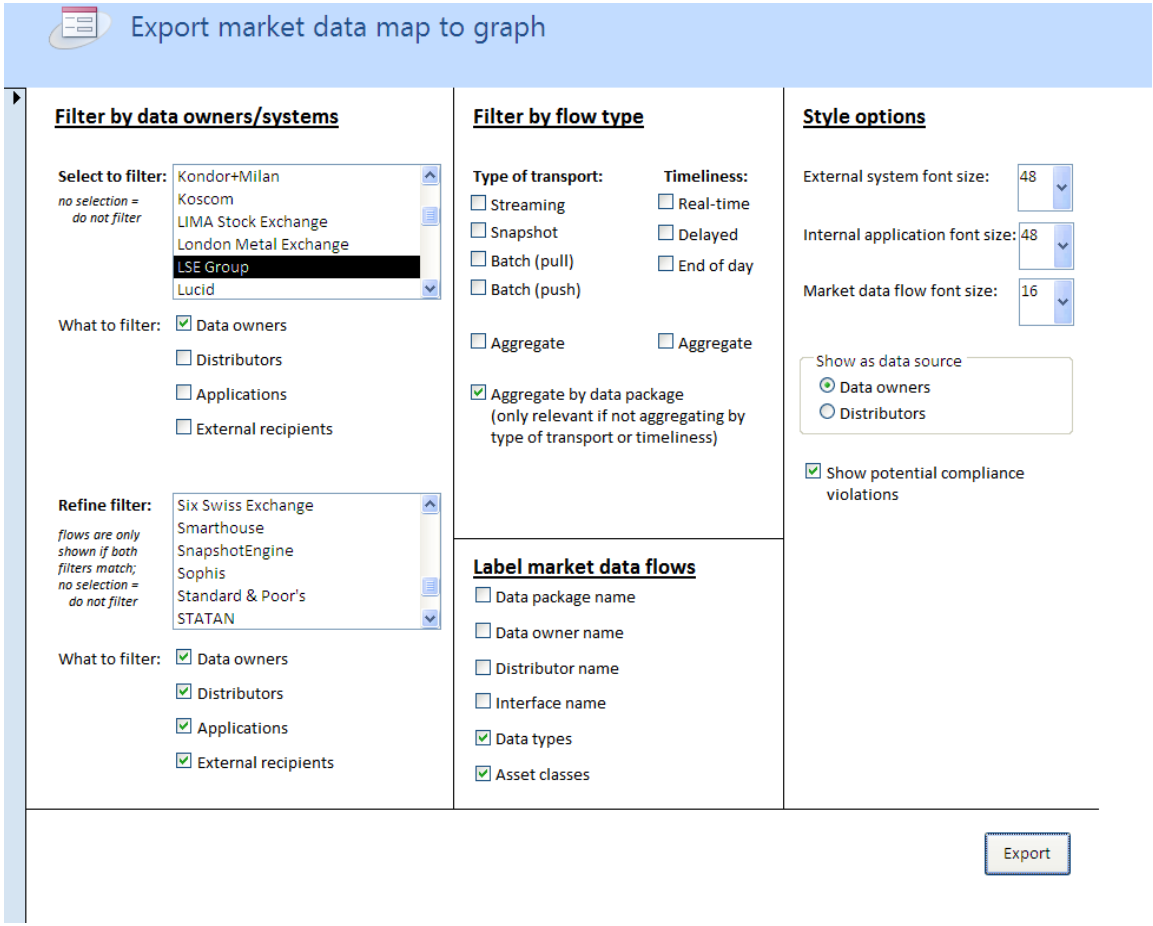
1. **Szenario**
   - Bisherige Lösung
   - Anforderungen
2. **Softwarearchitektur**
   - OData
   - Breeze
   - AngularJS
3. **Komponenten**
   - Filter
   - Graph
   - Tabellen
4. **Ausblick**
5. **Prototyp Vorstellung**

Szenario

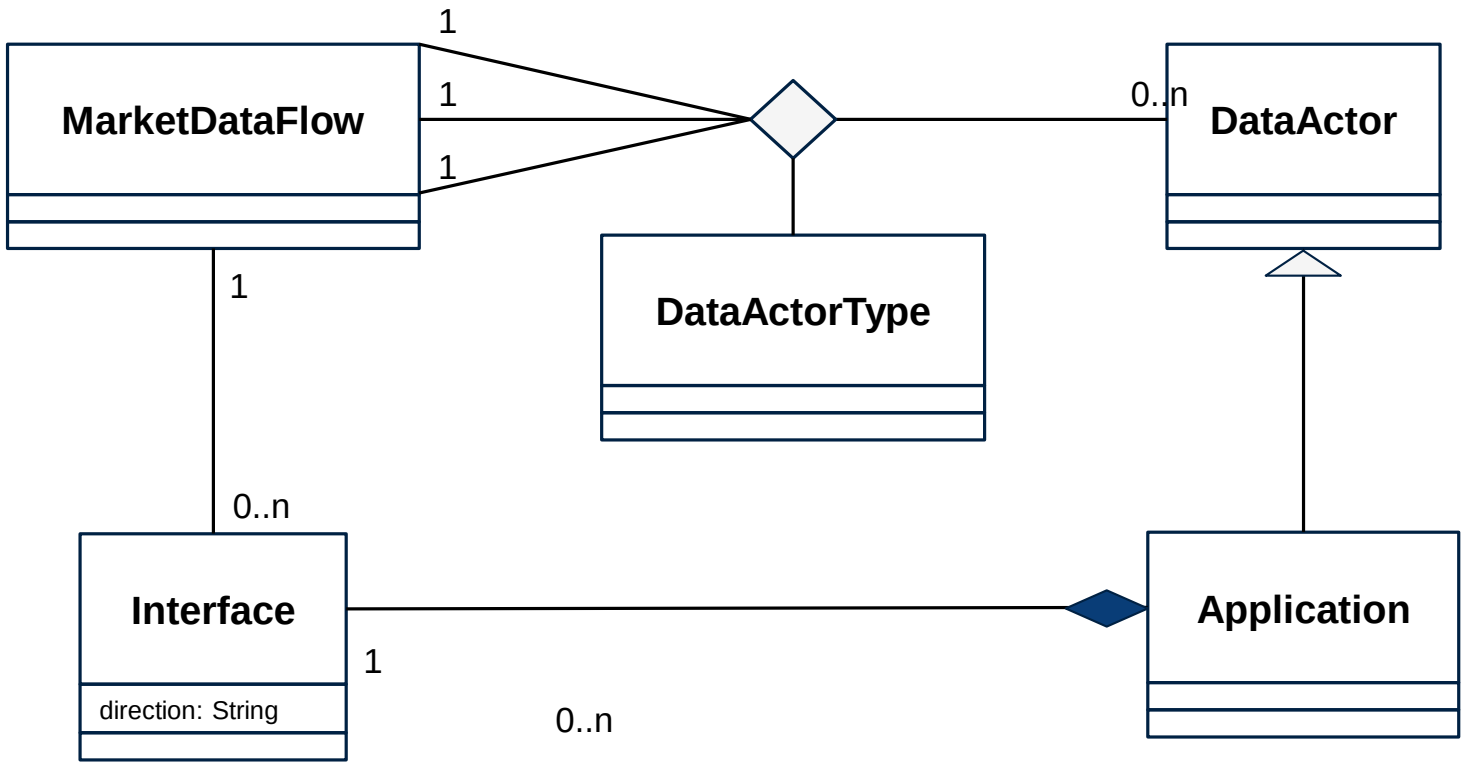


Bisherige Lösung

- Excel Fragebögen
- Access Daten Import
- Filter
  - Graphviz markup
- Market data map



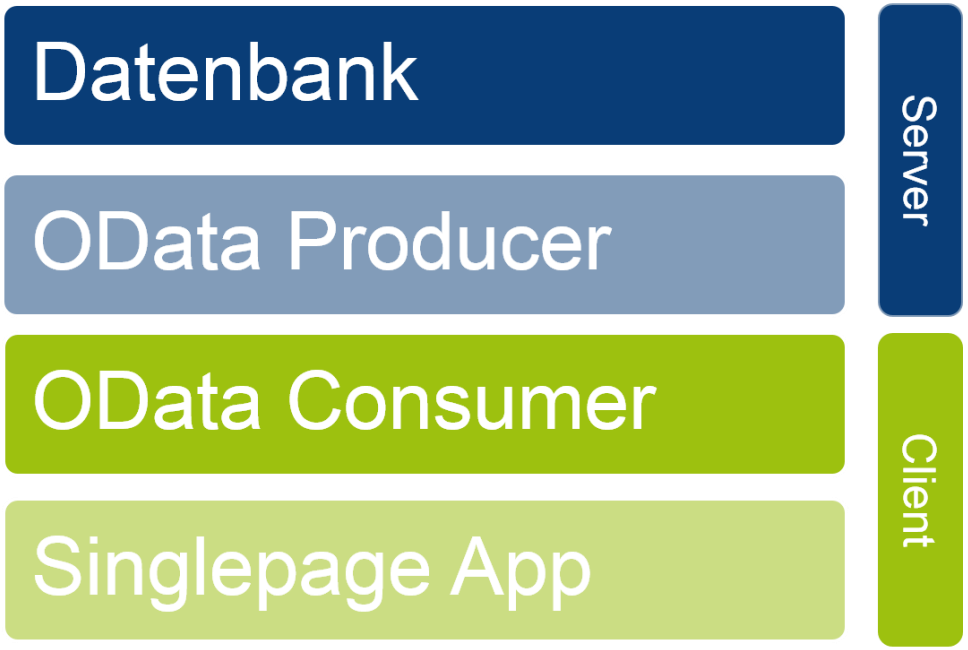
Datenmodell



Anforderungen

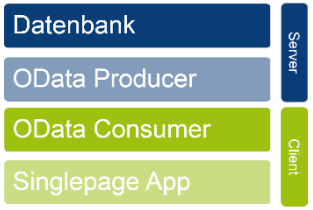
- Optische Unterscheidung (Entitäten und Datenflüsse)
- Navigierbarkeit der Graphen
- Konfigurierbare Darstellung (z.B. Kantenbeschriftungen)
- Filterung von Kanten und Knoten
- Aggregation von Kanten
- Definition und Anwendung von Regeln
- Detailinformationen von Knoten und Kanten
- Ausdruck oder Export
- CRUD Funktionalität
- Erweiterbarkeit des Schemas

Technology Stack

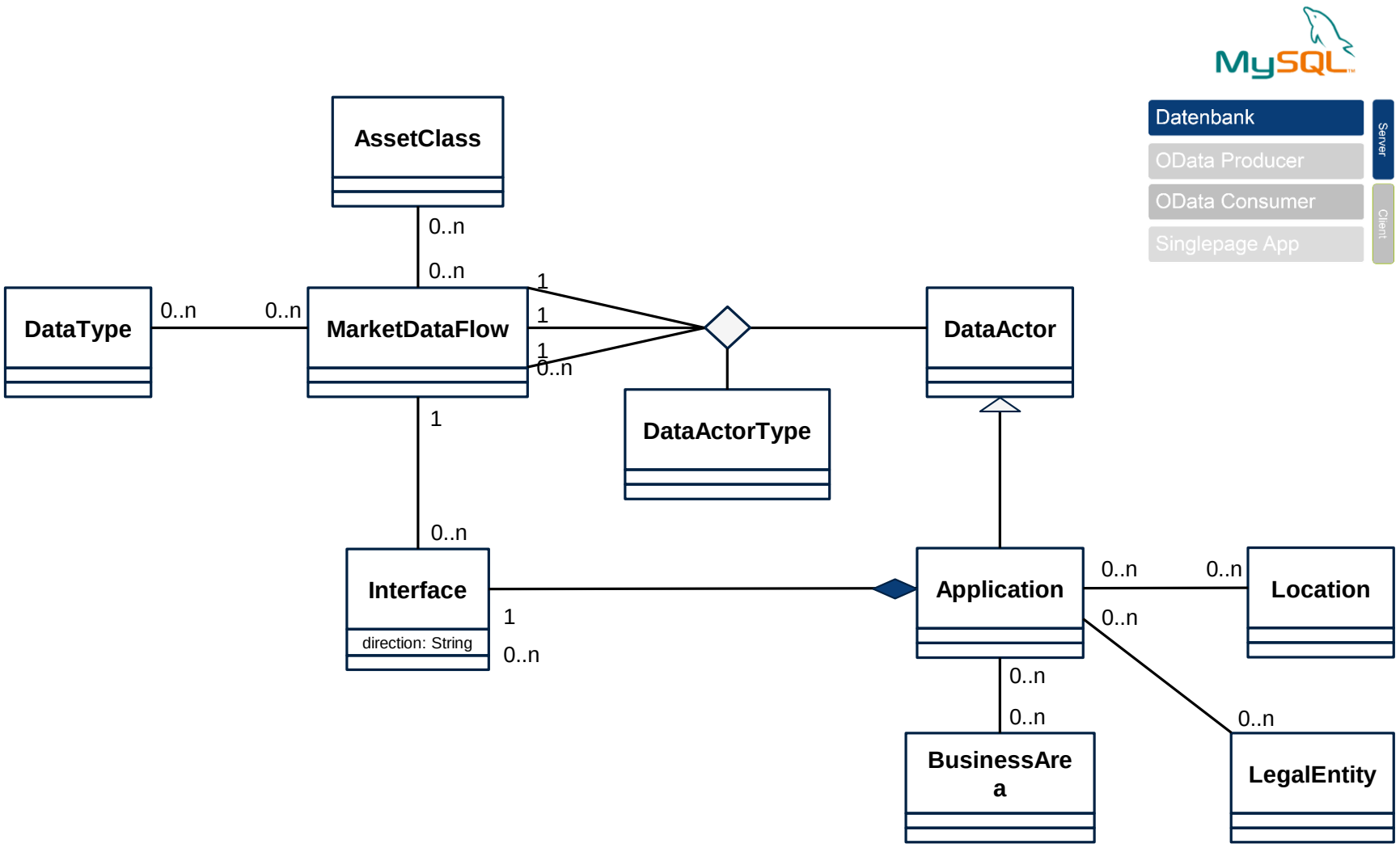


Verwendete Tools und Libraries

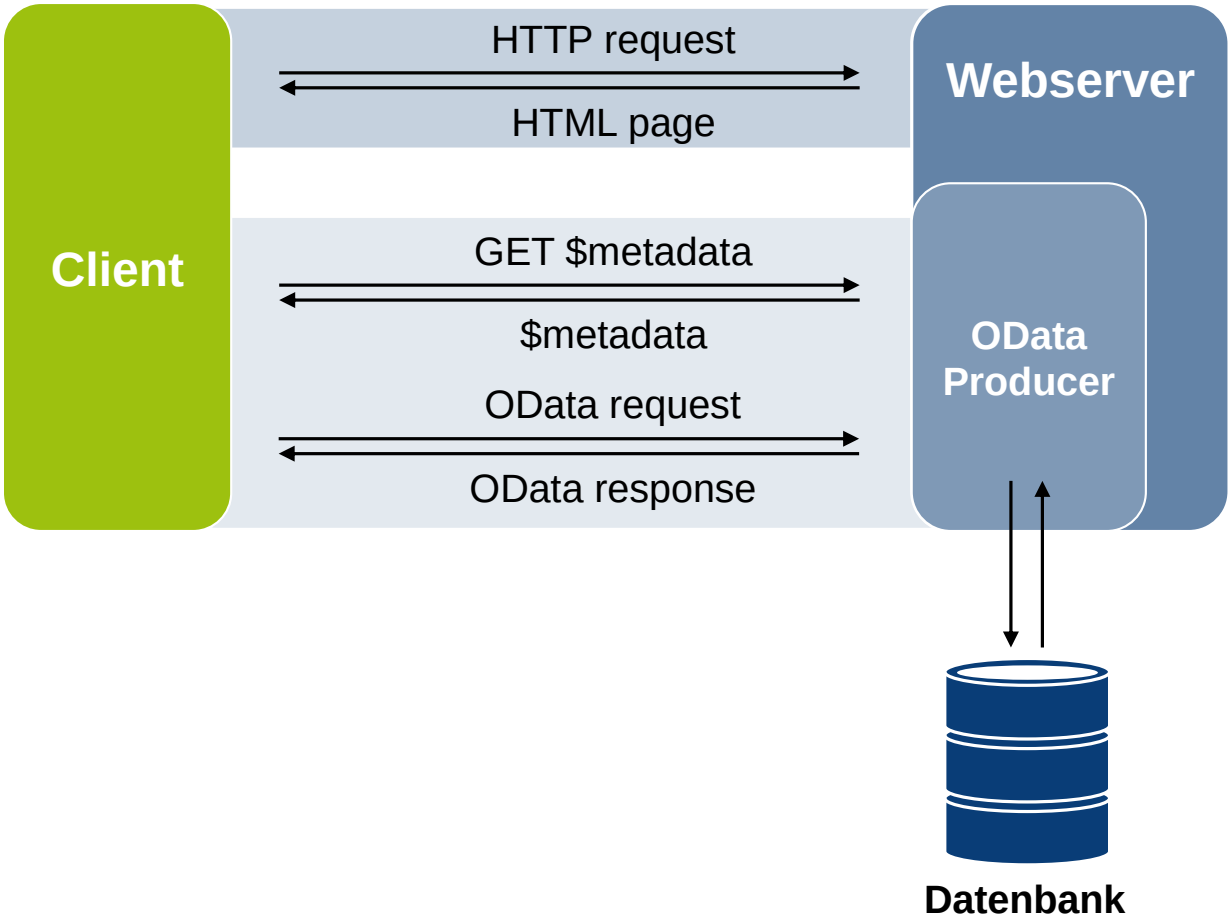
• MySQL	Datenbank
• Apache Olingo	OData Producer
• Tomcat	Java Application Server
• breezeJS	OData Consumer
dataJS	HTTP Kommunikation
• AngularJS	MVC Framework
xeditable	Detail Formulare
• D3	SVG Verwaltung
• Dagre	Graph Berechnung
• Twitter Bootstrap	CSS Design
• Font Awesome	Icons

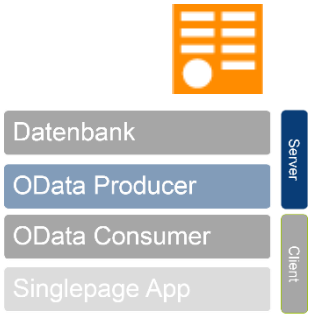


Datenbank – Schema

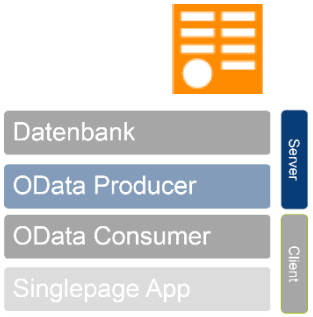


OData


Client
HTTP request
HTML page
Webserver
GET $metadata
$metadata
OData request
OData response
OData Producer
Datenbank


Datenbank
OData Producer
OData Consumer
Singlepage App
Server
Client

OData Producer – Apache Olingo



- OData 2.0 Unterstützung
- OASIS OData 4.0 in Entwicklung
- JPA model processor
- Entwickler: Apache Foundation unterstützt durch SAP

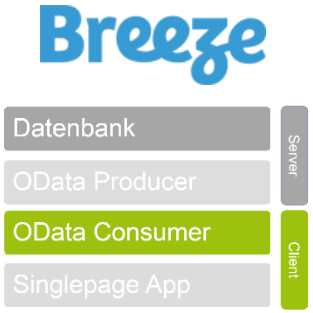
REST Operationen

- GET, POST, PUT, PATCH, DELETE
- $batch für gesammelte http anfragen

Nicht unterstütze Funktionen

- Keine groupBy unterstützung
- JoinColumns (nur olingo)

OData Consumer – BreezeJS



- JavaScript Bibliothek
- Asynchrone Kommunikation (Promises)
- Nachverfolgen von Änderungen
- Lokale Datenbankanfragen im Cache

Asynchrone Anfrage

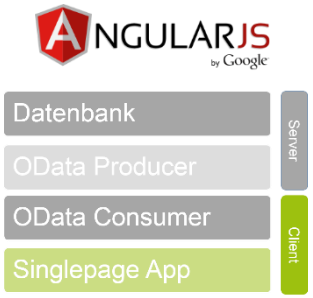
```
query = new breeze.EntityQuery()
    .from('Interfaces')
    .where('applicationID', 'Eq', 1234);
manager.executeQuery(query)
    .then(function(data) {
    interfaces = data.results;
});
```

Synchrone Anfrage

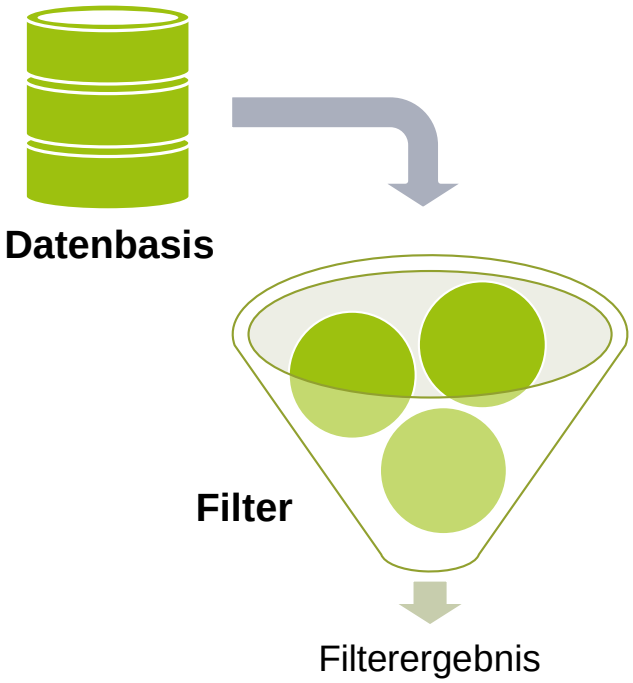
```
interfaces = breeze.EntityQuery
    .from('Interfaces')
    .where('applicationID', 'Eq', 1234)
    .using(manager)
    .executeLocally();
```

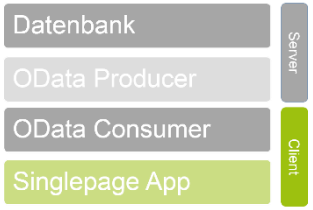
Singlepage App – Angular JS

- JavaScript Framework
- HTML = static web → AngularJS = dynamic web
- Model-View-Controller pattern (MVC)
  - Separation of concerns
  - Austauschbare Komponenten
  - Geschäftslogik getrennt von Anzeigelogik
- Custom HTML Tags (directives)
  <myTag></myTag>



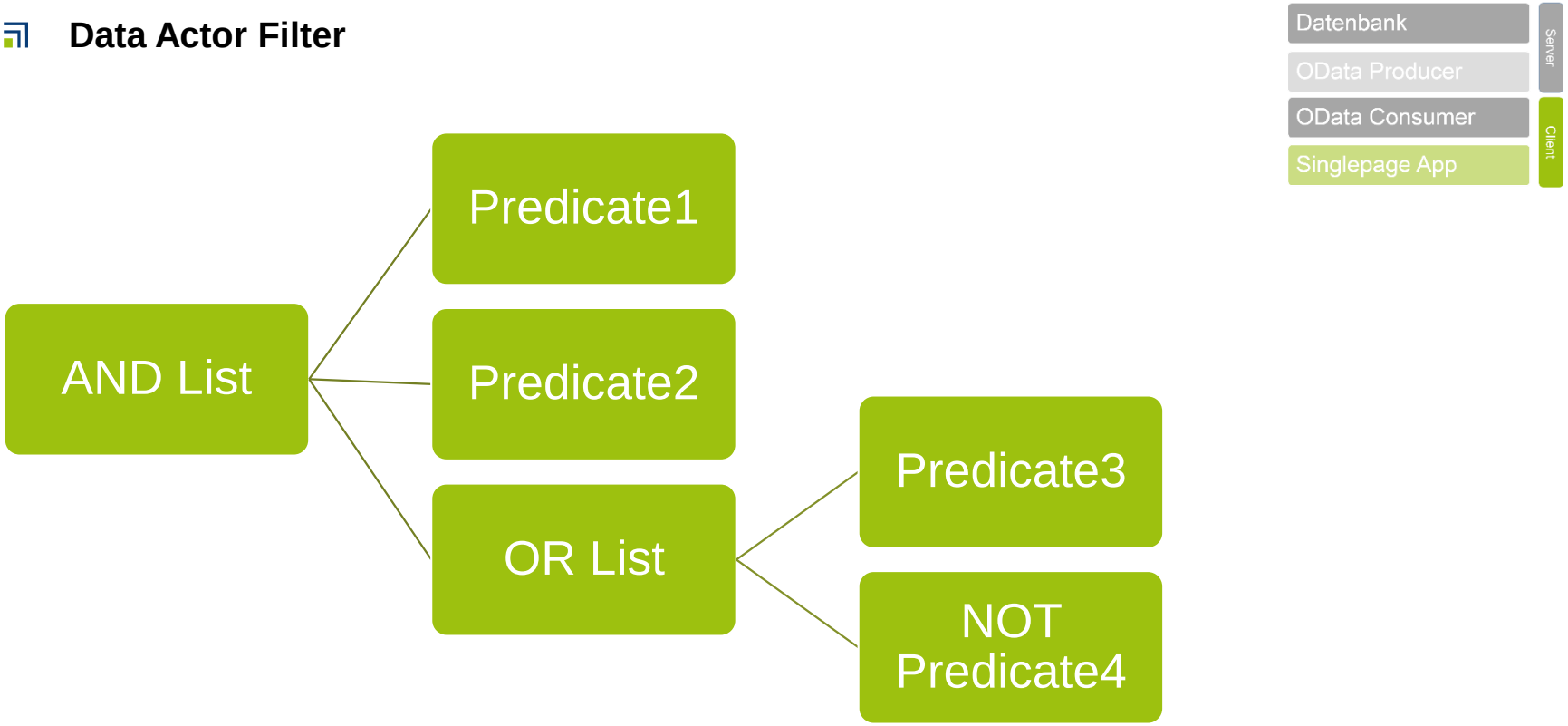
Singlepage App – Komponenten





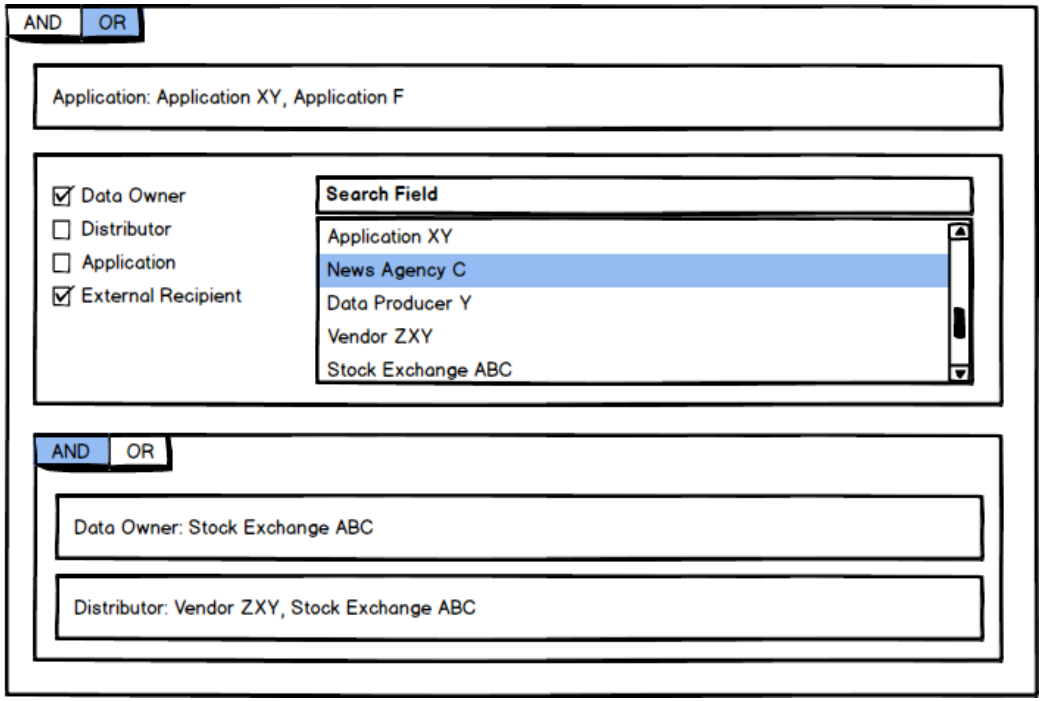
Komponenten – Filter

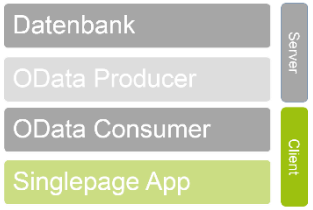
- **Data Actor Filter**



Komponenten – Filter

Data Actor Filter

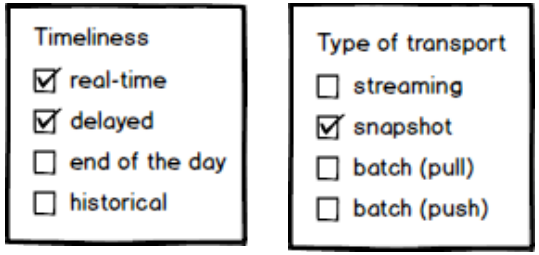




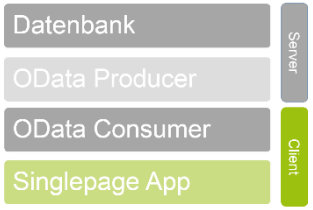
- Alternative Data Actor Rollen wählbar
- Alternative Data Actors wählbar
- AND bzw. OR Verknüpfungen
- Jede Ebene negierbar
- Fehlerwarnungen

Komponenten – Filter

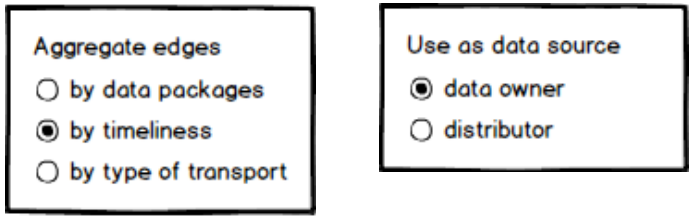
Markdatenfluss Filter



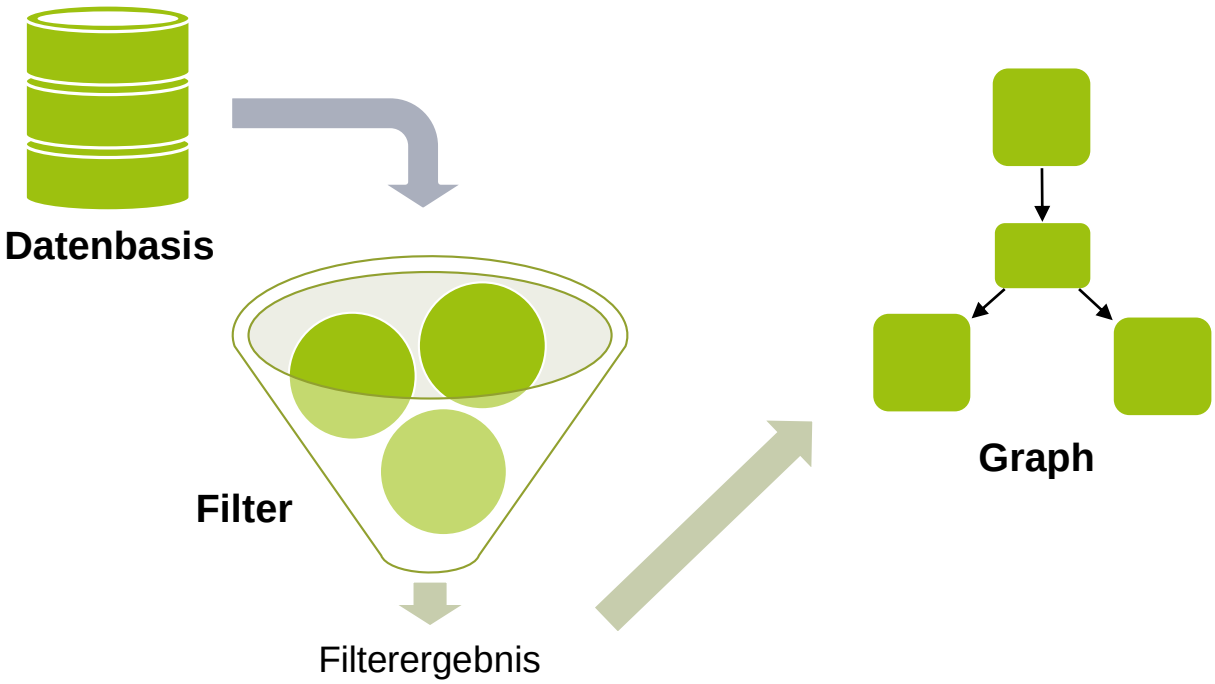
- Alternative Aggregierung oder Aggregieren aller Kanten
- Frei kombinierbare Labels
- Graphenorientierung wählbar
- Fehlerwarnungen

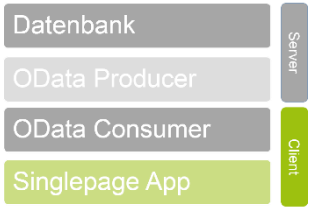


Graph Einstellungen

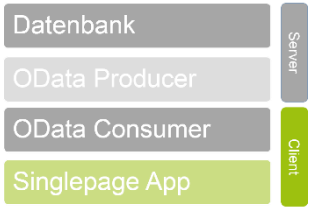


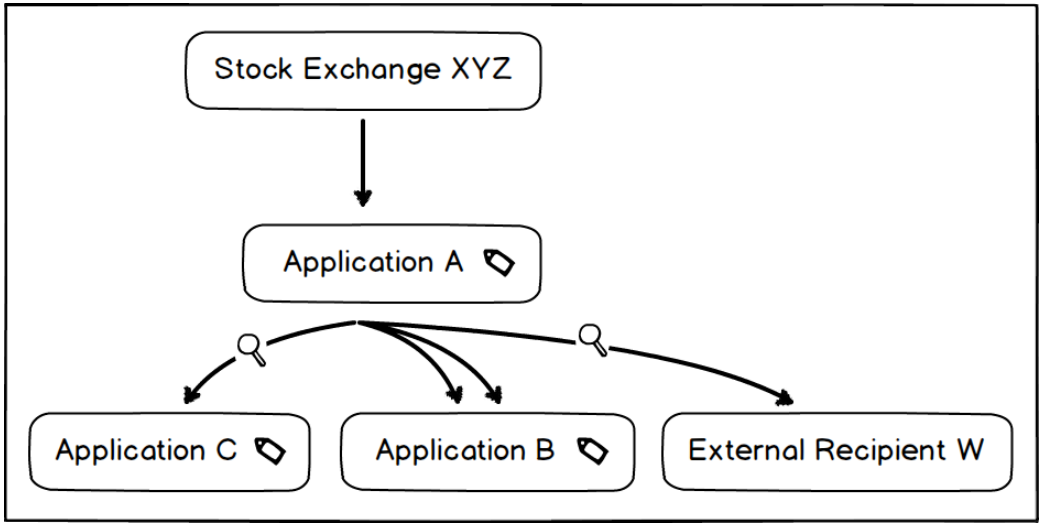
Singlepage App – Komponenten





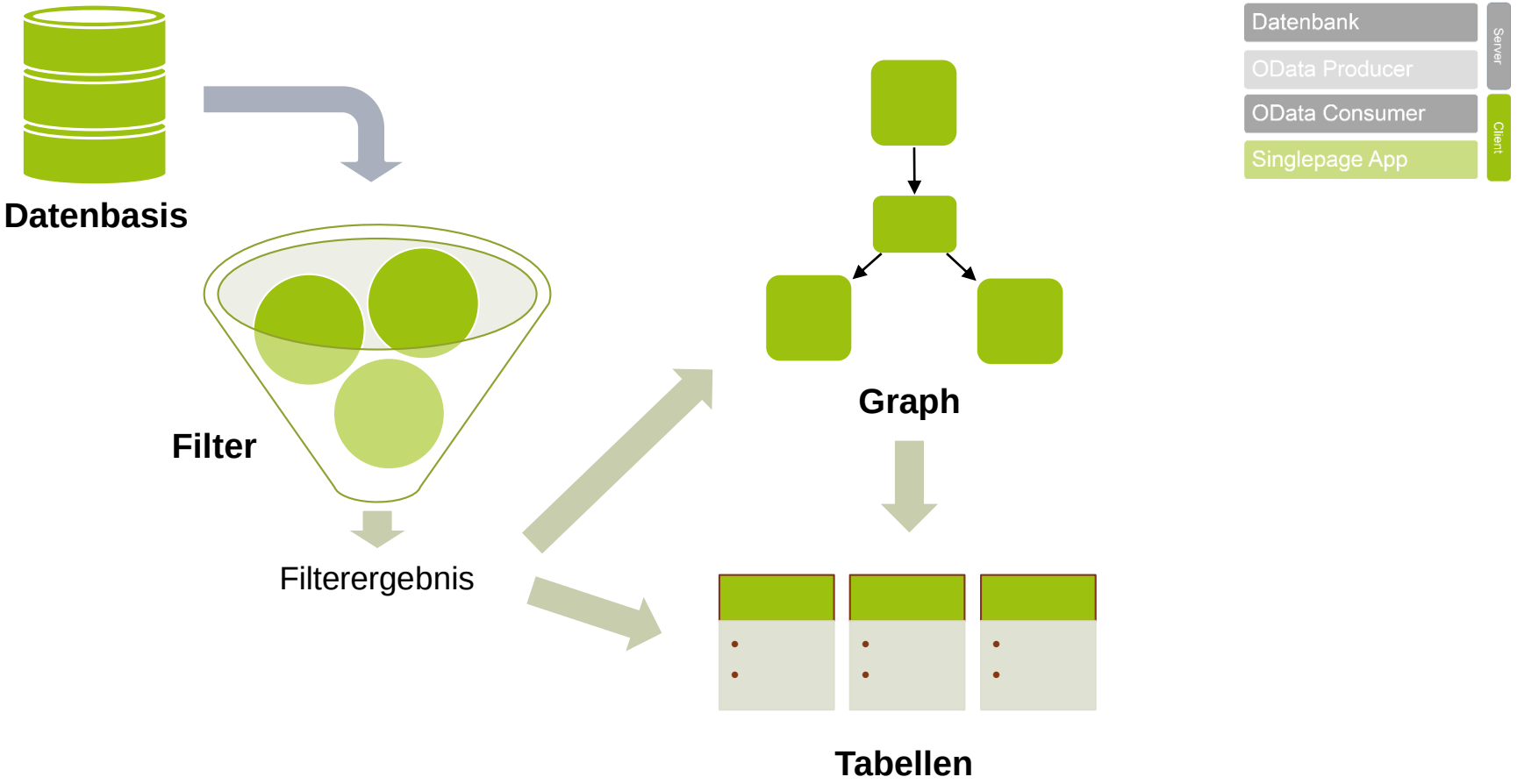
Komponenten – Graph



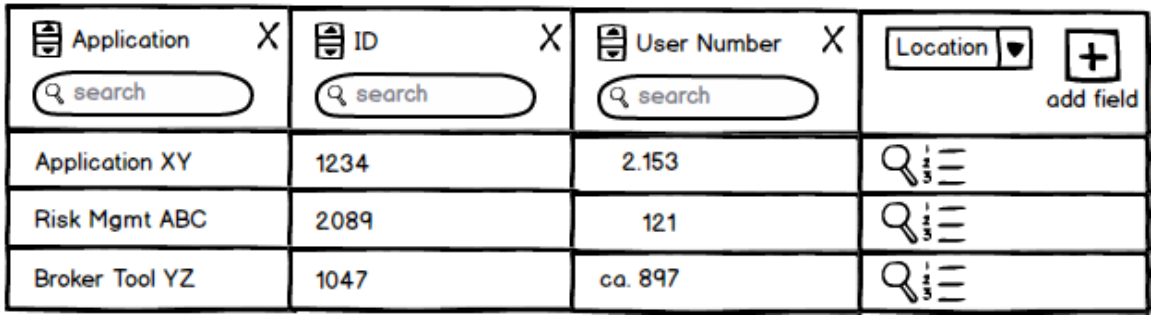


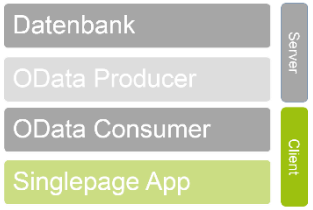
- Data Actors sind Knoten
- Marktdatenflüssen sind gerichtete Kanten
- Applikationen sind editierbar
- Einzelne Kanten sind editierbar
- Ursprung einer Kante ist Data Owner oder Distributor
- Navigierbar (drag‘n‘zoom)

Singlepage App – Komponenten



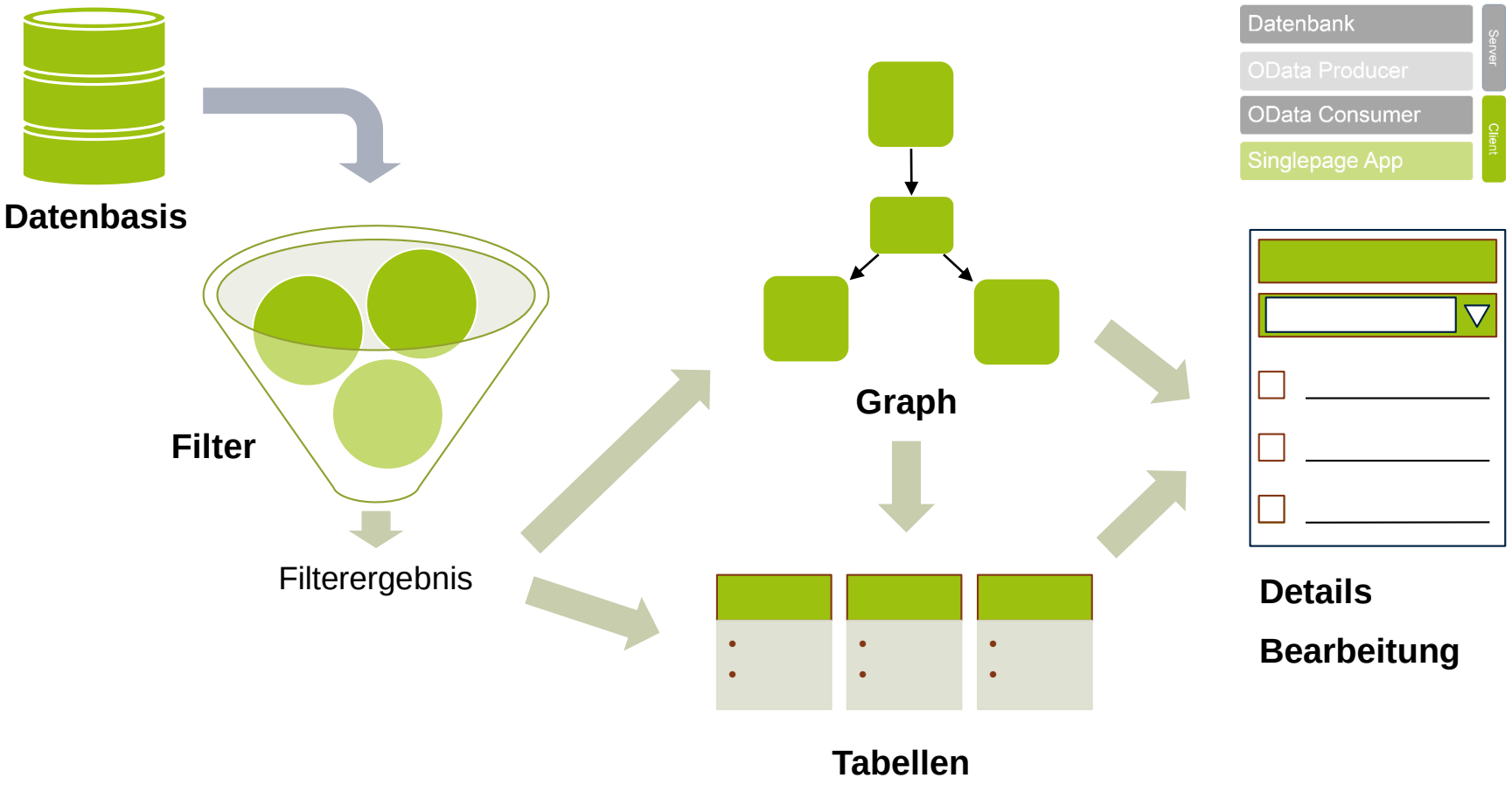
Komponenten – Tabellen



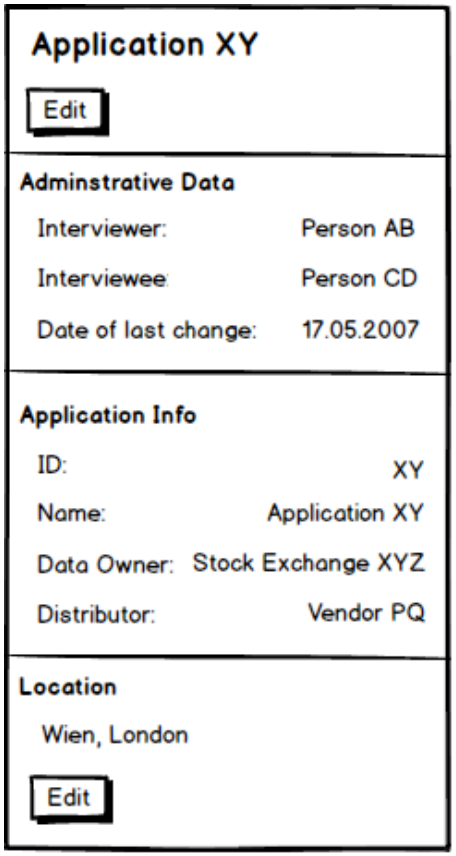


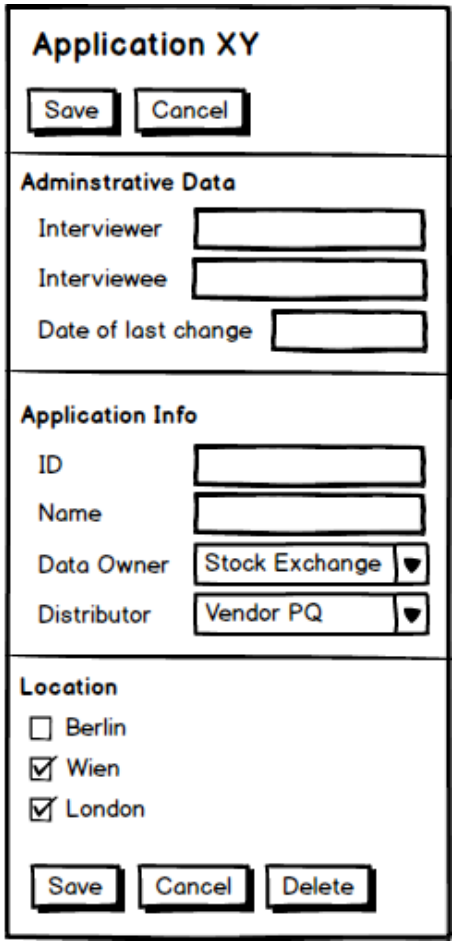
- Konfigurierbare Spalten
- Nach Feldern sortierbar
- Mehrfaches Eingrenzen durch Suchfelder
- Detailansichten
- Sublisten

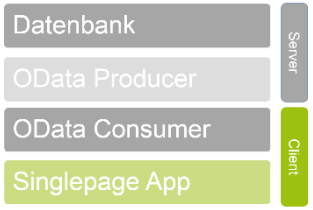
Singlepage App – Komponenten



Komponenten – Details & Edit

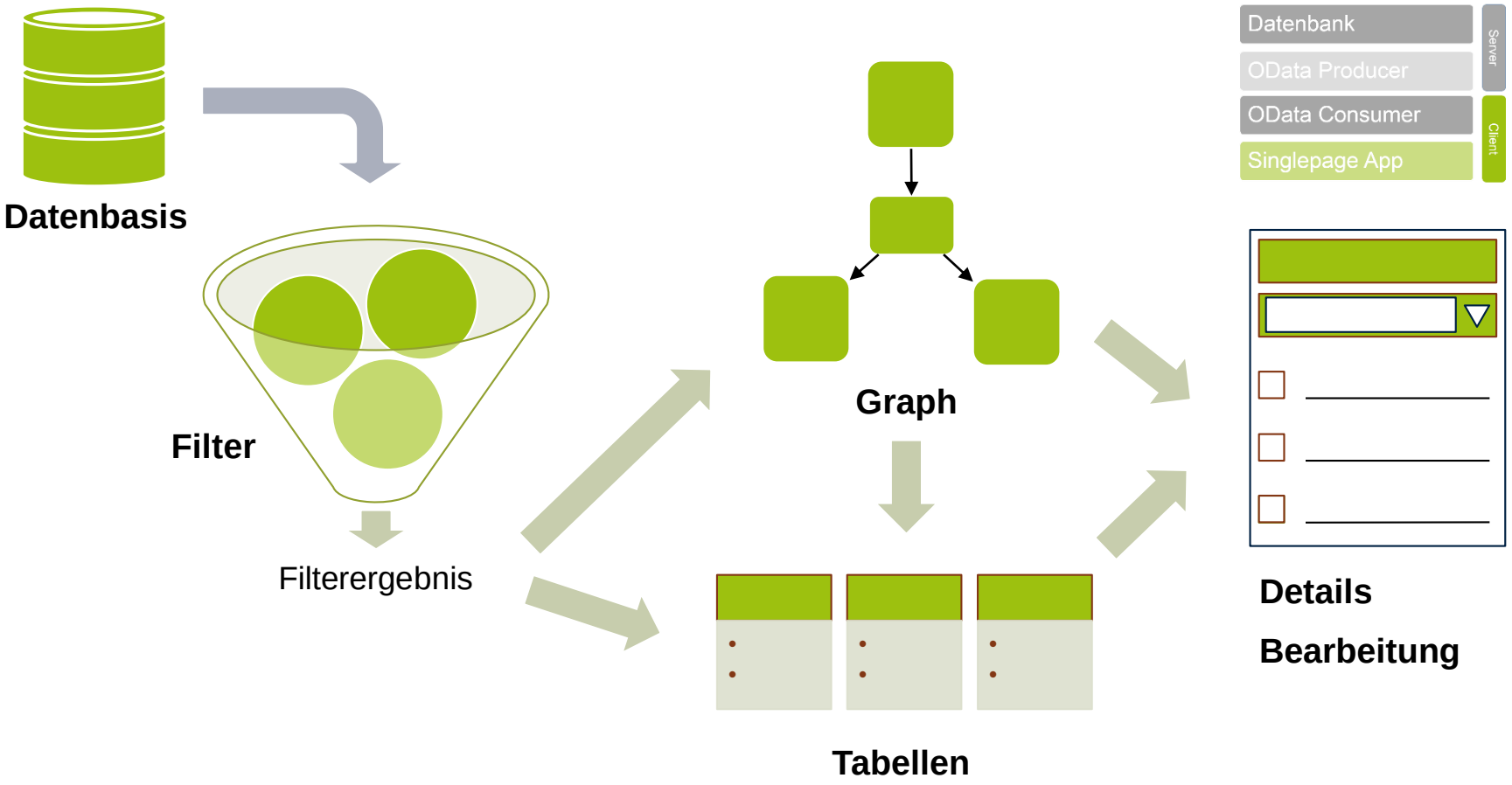






- Temporäres bearbeiten
- Freitextfelder
- Dropdown
- Checkbox und CheckList
- RadioList
- Löschen

Singlepage App – Komponenten

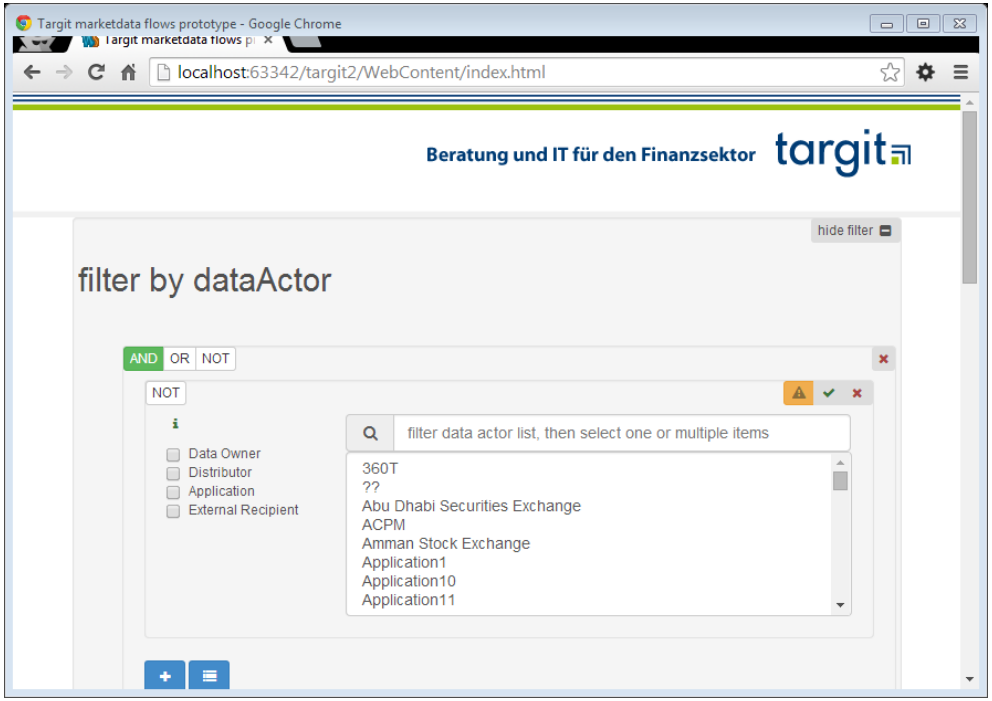


Ausblick

Mögliche Erweiterungen

- Rechteverwaltung
- Filter Templates speichern / exportieren / importieren
- Mini Map zur besseren Navigierbarkeit
- Komplexere Kantenbeschriftungen bei Aggregierung

Prototyp Vorstellung



Fragen – Anmerkungen – Diskussion

Daniel Graf Hoyos
daniel.hoyos@tum.de

Technische Universität München
Fakultät für Informatik
Lehrstuhl Software Engineering für betriebliche Informationssysteme

Boltzmannstraße 3
85748 Garching bei München

Tel +49.89.289.
Fax +49.89.289.17136

wwwmatthes.in.tum.de