Data Explorer Modul für Orchard CMS

Funktionsübersicht, Seitenstruktur und technische Vision

Verfasst von: Marwen Ben Salem

Abteilung: Team Workflow Systems

Stand: 19. Juni 2025

Version: 1.0

Inhalt

Teil 1: Data Explorer Analysis

1. *Einleitung*

Das Ziel dieses Projekts ist die Entwicklung eines Analyse-Moduls, das als Erweiterung (Modul) in Orchard CMS integriert wird. Das Modul soll es ermöglichen, benutzerdefinierte tabellarische Datenansichten (sog. „Views“) ähnlich wie in Tools wie Qlik darzustellen und flexibel zu konfigurieren. In der ersten Phase wird ein funktionaler Prototyp mit grundlegender Tabellenanzeige entwickelt.

2. *Problemstellung*

Orchard CMS bietet keine native Funktion zur dynamischen Analyse oder tabellarischen Visualisierung von Daten aus verschiedenen Quellen. Für Projekte mit Analysebedarf müssen externe Tools wie Qlik eingebunden werden – diese sind jedoch in der Regel mit hohen Lizenzkosten verbunden.

3. *Projektziele*

- Entwicklung eines flexiblen Analysemoduls als Orchard-Modul zur Ablösung bestehender Qlik-Funktionalitäten
- Unterstützung mehrerer Datenquellen wie SQL-Server, Excel-Dateien und APIs
- Aufbau eines Datenflusses von der Quelle bis zur Darstellung in dynamischen Tabellen
- Einsatz von Orchard-Shapes zur modularen Verarbeitung und Darstellung der Daten
- Vorbereitung auf komplexe Analyse-Szenarien, inklusive Beziehungen zwischen Tabellen (Primary/Foreign Keys)
- Erweiterbarkeit für künftige Funktionen wie Filter, Aggregationen, Visualisierungen und Kennzahlen

4. *Kernfunktionen*

- Auswahl und Anbindung einer Datenquelle durch den Benutzer (SQL, Excel, API usw.)
- Anzeige der verfügbaren Tabellen und Auswahl einzelner Tabellen zur Darstellung
- Darstellung der Tabellendaten in einem interaktiven, filterbaren UI
- Benutzerdefinierte Auswahl und Benennung von Spalten
- Speicherung der Konfiguration als Content-Part oder in einem eigenen Record
- Vorbereitung auf spätere Erweiterungen wie Relationen, Filterfunktionen und Metriken

5. *Einschränkungen und Risiken*

- Orchard hat **limitierte** moderne UI-Komponenten
- **Komplexität** steigt stark bei dynamischen Joins oder relationalen Views
- **Performance-Einschränkungen** bei großen Datenmengen, insbesondere bei direktem Zugriff auf Datenquellen.
- **Sicherheitsrisiken** durch Zugriff auf externe Datenquellen und mögliche Berechtigungsprobleme

- **Begrenzte Funktionalität in der ersten Version**, insbesondere bezüglich komplexer Datenbeziehungen und Analysefunktionen
- **Notwendigkeit kontinuierlicher Weiterentwicklung** zur Anpassung an neue Anforderungen und Datenquellen

6. *Spezifikation der Anforderungen*

❖ *Funktionale Anforderungen*

- Der Benutzer kann eine Datenquelle auswählen (z. B. SQL-Server, Excel-Datei, API).
- Die verfügbaren Datenbanken und Tabellen werden nach Auswahl automatisch geladen.
- Der Benutzer kann Tabellen zur Ansicht auswählen und Spalten manuell ein-/ausblenden.
- Die ausgewählten Daten werden in einem dynamischen Tabellenformat im Frontend angezeigt.
- Konfigurationen (z. B. Datenquelle, Spaltenauswahl) werden gespeichert und wiederverwendbar gemacht.
- Möglichkeit zur späteren Erweiterung um Filter, Metriken, Relationen und Visualisierungen.

❖ *Nicht-funktionale Anforderungen*

- **Modularität**: Das Modul soll unabhängig und erweiterbar innerhalb von Orchard CMS sein.
- **Benutzerfreundlichkeit**: Einfache und intuitive Benutzeroberfläche für technische und nicht-technische Benutzer.
- **Sicherheit**: Zugriff auf Datenquellen nur mit gültigen Berechtigungen, keine ungeschützte Verbindung.
- **Wartbarkeit**: Klare Trennung zwischen Datenzugriff, Geschäftslogik und Präsentation.
- **Leistung**: Effiziente Datenabfragen, um Ladezeiten bei großen Datenmengen zu minimieren.
- **Kompatibilität**: Unterstützung der gängigen Datenformate und -quellen (SQL, Excel, JSON/API).

7. *Projektphasen und Vorgehensweise (mit Zeitplanung)*

Das Projekt folgt einem klassischen Lebenszyklusmodell, das in sieben aufeinanderfolgende Phasen unterteilt ist. Dieser strukturierte Ansatz gewährleistet eine systematische und kontrollierte Entwicklung des Data Explorer-Moduls.

Phase	**Beschreibung**	**Dauer (geschätzt)**
Planungs- und Vorstudienphase	Zieldefinition, Abgrenzung, Technologieauswahl, Voranalyse bestehender Lösungen	3–5 Tage
Anforderungsanalyse	Erhebung und Spezifikation der funktionalen und nicht-funktionalen Anforderungen	5–7 Tage
Konzeption (Technische Spezifikation)	Entwurf der Architektur, Datenmodelle, Datenflüsse, UI-Komponenten	7–10 Tage
Implementierung	Entwicklung des Moduls (Frontend + Backend), inkl. Konfigurationslogik und Datenanzeige	15–25 Tage
Test und Validierung	Funktionstests, UI-Überprüfung, Datenkonsistenz, Fehlerbehebung	5–8 Tage
Deployment und Inbetriebnahme	Integration ins CMS, Tests im Staging, produktives Deployment	3–5 Tage
Wartung und Weiterentwicklung	Fehlerkorrekturen, kleine Anpassungen, Planung künftiger Features (Filter, Visualisierungen)	Laufend / nach Bedarf

Teil 2: Technische Spezifikation und Seitenstruktur

1. Einleitung

Dieser Teil dient als funktionale und konzeptionelle Grundlage für eine **vorgedachte Lösung**, die auf der Idee basiert, ein Analysemodul innerhalb von Orchard CMS zu entwickeln – inspiriert von den Kernfunktionen von Qlik (z. B. Datenquellenanbindung, tabellarische Analyse, Konfiguration von Views).

Die hier vorgestellte Struktur und Seitenlogik stellen **einen ersten Entwurf** dar, der dazu dient, die Anwendungsidee verständlich zu machen und im Team diskutieren zu können. Die beschriebenen Seiten, Datenflüsse und Komponenten sind **nicht final**, sondern sollen als Ausgangspunkt für Rückmeldungen, Überlegungen zur technischen Machbarkeit und zukünftige Optimierungen dienen.

Ziel dieses Abschnitts ist es, **eine klare Vorstellung der potenziellen Anwendung** zu vermitteln und eine gemeinsame Basis für die weitere technische Spezifikation sowie Backend- und Frontendentwicklung zu schaffen.

2. Allgemeine Übersicht der geplanten Seiten

In diesem Abschnitt wird ein Überblick über die zentralen Seiten gegeben, die im Rahmen des Moduls entwickelt werden sollen. Zu den vorgesehenen Seiten gehören:

- die **Index-Seite** (Übersicht und Einstiegspunkt)
- die **Edit-Seite** (Bearbeitung von Inhalten oder Konfigurationen)
- die **Properties-Seite** (Verwaltung von Eigenschaften, z. B. Datenquellen)
- sowie die **View-Frontend-Seite**, die den Endnutzerinnen und -nutzern angezeigt wird

➢ Jede dieser Seiten wird im weiteren Verlauf dieses Dokuments einzeln beschrieben. Dabei wird auf die jeweilige Funktionalität, die logische Rolle im System und eine einfache visuelle Darstellung (Layout-Skizze) eingegangen. Ziel ist es, ein klares Verständnis dafür zu vermitteln, wie jede Seite aufgebaut wird, welche Aufgaben sie erfüllt und wie sie sich in das Gesamtmodul einfügt.

3. *Index-Seite*

Die **Index-Seite** bildet den zentralen Einstiegspunkt für Admin, um mit dem Modul **Data Explorer** zu arbeiten. Sie stellt die erste Anlaufstelle dar und bietet Zugriff auf sämtliche im System erstellten Data Explorer.

Im oberen Bereich der Seite befindet sich eine Schaltfläche **„Create new Data-Explorer Definition“**, mit der ein neuer, leerer Data Explorer angelegt werden kann.

Darunter wird eine **Übersichtsliste aller bestehenden Data Explorer** angezeigt. Jeder Eintrag in der Liste zeigt den Namen des jeweiligen Data Explorers sowie mehrere Aktionsmöglichkeiten:

- **Clone** – Erstellt eine identische Kopie des gewählten Data Explorers.
- **Delete** – Löscht den entsprechenden Data Explorer dauerhaft.
- **Properties** – Führt zur Properties-Seite, auf der zentrale Eigenschaften wie Name und Datenquellen bearbeitet werden können. Diese Seite wird im weiteren Verlauf dieses Dokuments näher beschrieben.
- **Edit** – Öffnet die Bearbeitungsansicht, in der auf Basis der Properties (z. B. Datenquellen) grafische Darstellungen und konfigurierbare Elemente erstellt werden. Auch diese Seite wird im späteren Abschnitt detailliert erläutert.

Diese Seite dient somit als **zentraler Verwaltungsbereich**, um bestehende Einträge zu pflegen und neue Data Explorer zu erstellen. Sie bildet die Grundlage für alle weiteren Arbeitsschritte im Modul.

Beispiel:

Nachfolgend ist eine beispielhafte Darstellung des erwarteten Seitenaufbaus zu sehen:

Abbildung 1: Index View

4. *Properties-Seite*

Die **Properties-Seite** ermöglicht es dem Benutzer, zentrale Eigenschaften eines bestehenden Data Explorers zu bearbeiten. Sie dient als Konfigurationsoberfläche für die Grundparameter, die später in der Analyse und Visualisierung verwendet werden.

Funktionen dieser Seite :

- **Name ändern**: Der Benutzer kann den Namen (Definition) des Data Explorers anpassen, um ihn eindeutig zu identifizieren.
- **Datenquelle auswählen**: Der Benutzer muss angeben, welchen **Typ von Datenquelle** er für die Analyse verwenden möchte – beispielsweise **Excel, SQL, API usw...**
- **Dynamische Eingabefelder**: Abhängig vom gewählten Datentyp werden dynamisch weitere Eingabefelder angezeigt.
  Wenn z. B. **Excel** als Datenquelle ausgewählt wird, erscheint ein Upload-Bereich, über den der Benutzer eine oder mehrere Excel-Dateien hochladen kann.

Die hochgeladenen Dateien werden im Backend gespeichert und bilden die Grundlage für die spätere **Analyse und Visualisierung** auf der **Edit-Seite**, die in einem späteren Abschnitt dieses Dokuments ausführlich behandelt wird.

Beispiel:

Nachfolgend ist eine beispielhafte Darstellung des erwarteten Seitenaufbaus zu sehen:

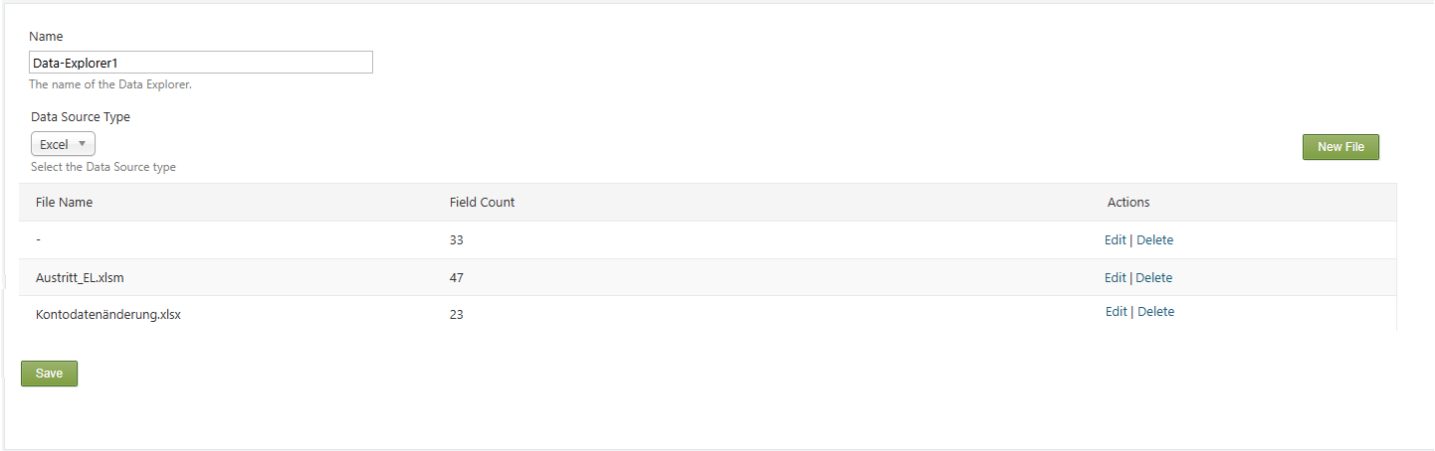

Abbildung 2: Properties View

5. *Edit-Seite*

Die **Edit-Seite** stellt das **Herzstück** des Moduls dar. Sie ermöglicht die visuelle und logische Konfiguration der Datenstruktur, die später zur Analyse und Visualisierung genutzt wird.

Hauptfunktionen dieser Seite :

- **Grafische Darstellung der Tabellen**:
  Ähnlich wie im Workflow-Modul wird hier ein visuelles Interface mit grafischen Elementen angezeigt. Jedes Element repräsentiert eine **Tabelle** – z. B. eine aus einer hochgeladenen Excel-Datei.
- **Verknüpfung von Tabellen**:
  Die Kacheln können miteinander **verbunden** werden. Dabei definiert der Benutzer :
  - eine **Primärschlüssel-Spalte** aus der ersten Tabelle,
  - eine **Fremdschlüssel-Spalte** aus der zweiten Tabelle,
  - sowie eine **Beziehungsart** (Join-Logik), die auf die Verknüpfung angewendet wird.
- **Tabellenstruktur einsehen**:
  Durch einen **Doppelklick** auf ein Element öffnet sich ein Dialogfenster, das die **Spaltennamen** der entsprechenden Tabelle anzeigt.
  Sobald alle Tabellen korrekt verbunden und konfiguriert sind, ist das Backend mit den notwendigen Metadaten vorbereitet, um eine **erste strukturierte Datenansicht** zu generieren – die Grundlage für die spätere **Frontend-Visualisierung**, welche im nächsten Kapitel beschrieben wird.

Beispiel Daten:

Tabelle 1: Clients

ClientID (PK)	Name	Country
1	Alice Dupont	France
2	Max Müller	Allemagne
3	Laura Rossi	Italie

Tabelle 2: Orders

OrdersID	ClientID (FK)	Date	Amount
101	1	2024-06-01	250 €
102	2	2024-06-05	180 €
103	1	2024-06-07	90 €

Beispiel:

Die folgende schematische Darstellung zeigt den Aufbau der Edit-Seite und das Zusammenspiel der Tabellen:

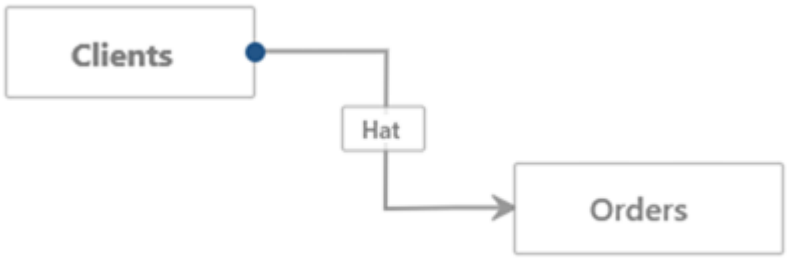

Abbildung 3: Tabellen grafisch verknüpfen

Abbildung 4: Verknüpfen von Tabellen mit Primärschlüssel und Fremdschlüssel

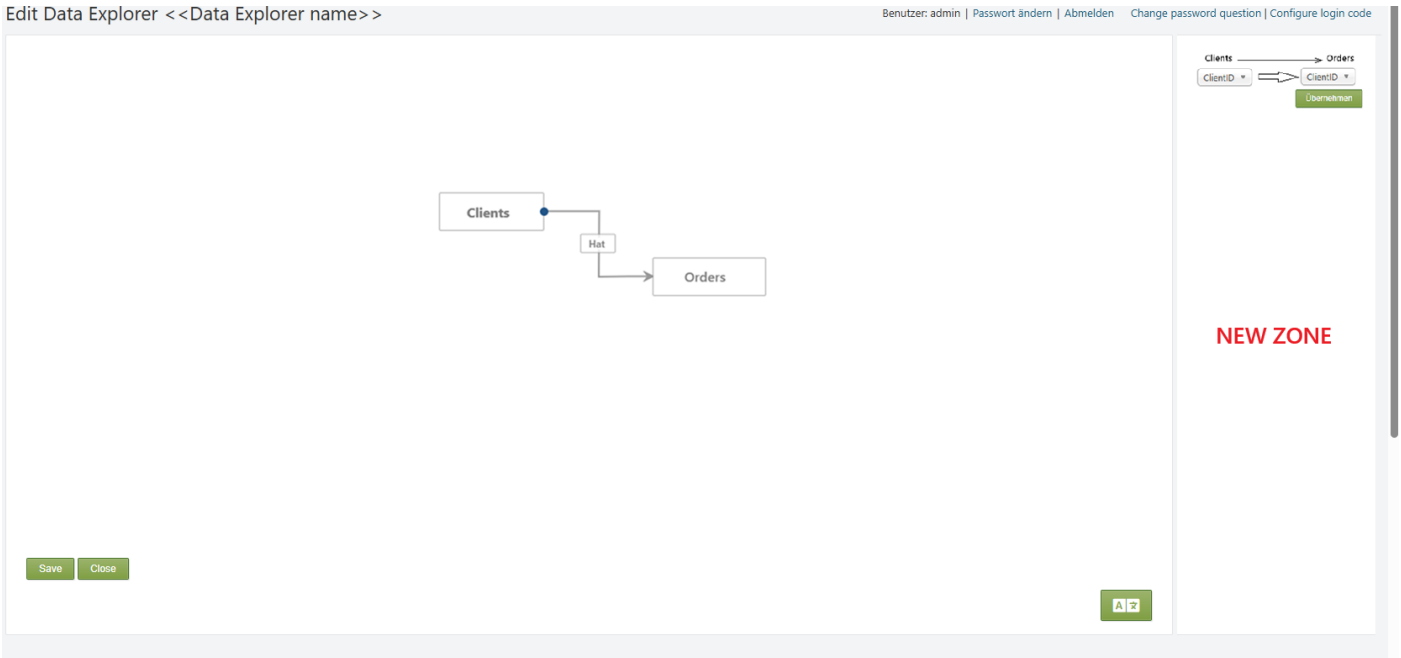

Abbildung 5: Erwartetes Ansicht

6. *View-Seite (Frontend-Anzeige für Benutzer)*

Die **View-Seite** stellt die finale Nutzerschnittstelle dar, auf der die aufbereiteten Daten auf Grundlage der konfigurierten Datenquellen und Tabellenverknüpfungen dargestellt werden. Sie ist ausschließlich für Endanwender bestimmt und dient der **Datenanalyse**, **Visualisierung** und **Interaktivität** – vergleichbar mit bekannten Tools wie **Qlik**.

Ziel dieser Seite:

Dem Benutzer soll eine flexible, dynamische Ansicht der Daten geboten werden, die auf den im Backend definierten Beziehungen basiert. Die Anzeige passt sich an :

- die hochgeladenen Datenquellen (z. B. Excel),
- sowie die in der **Edit-Seite** definierten Verknüpfungen zwischen den Tabellen an.

Hauptfunktionen der View-Seite :

1. **Tabellenansicht**
   Die konsolidierten Daten aus den verschiedenen Quellen werden in einer einzigen **Daten-Tabelle** dargestellt.
2. **Spaltenauswahl**
   Benutzer können **auswählen**, welche Spalten (Datenfelder) in der Tabelle angezeigt werden sollen, um irrelevante Informationen auszublenden.
3. **Daten sortieren**
   Die Tabelle kann nach jeder beliebigen Spalte **sortiert** werden – aufsteigend oder absteigend –, um Muster oder Prioritäten sichtbar zu machen.
4. **Berechnete Spalten hinzufügen**
   Benutzer können eigene Spalten definieren, die auf vorhandenen Daten basieren. Beispiele :
   - Summen,
   - Prozentsätze,
   - mathematische Formeln.
     Diese Funktion erlaubt flexible Datenberechnungen ohne Eingriff ins Backend.
5. **Grafische Visualisierungen erstellen**
   neben der Tabelle können **statistische Diagramme** (z. B. Balken-, Kreis- oder Liniendiagramme) hinzugefügt werden, die bestimmte Felder oder Aggregationen visualisieren.

Beispielhafte Anwendungsfälle :

- Umsatz pro Land in einem Balkendiagramm anzeigen.
- Durchschnittlicher Bestellwert pro Kunde berechnen.
- Tabellenansicht filtern nach bestimmten Zeiträumen.

Beispiel:

Die folgende Grafik veranschaulicht eine mögliche Darstellung der View-Seite mit interaktiven Optionen:

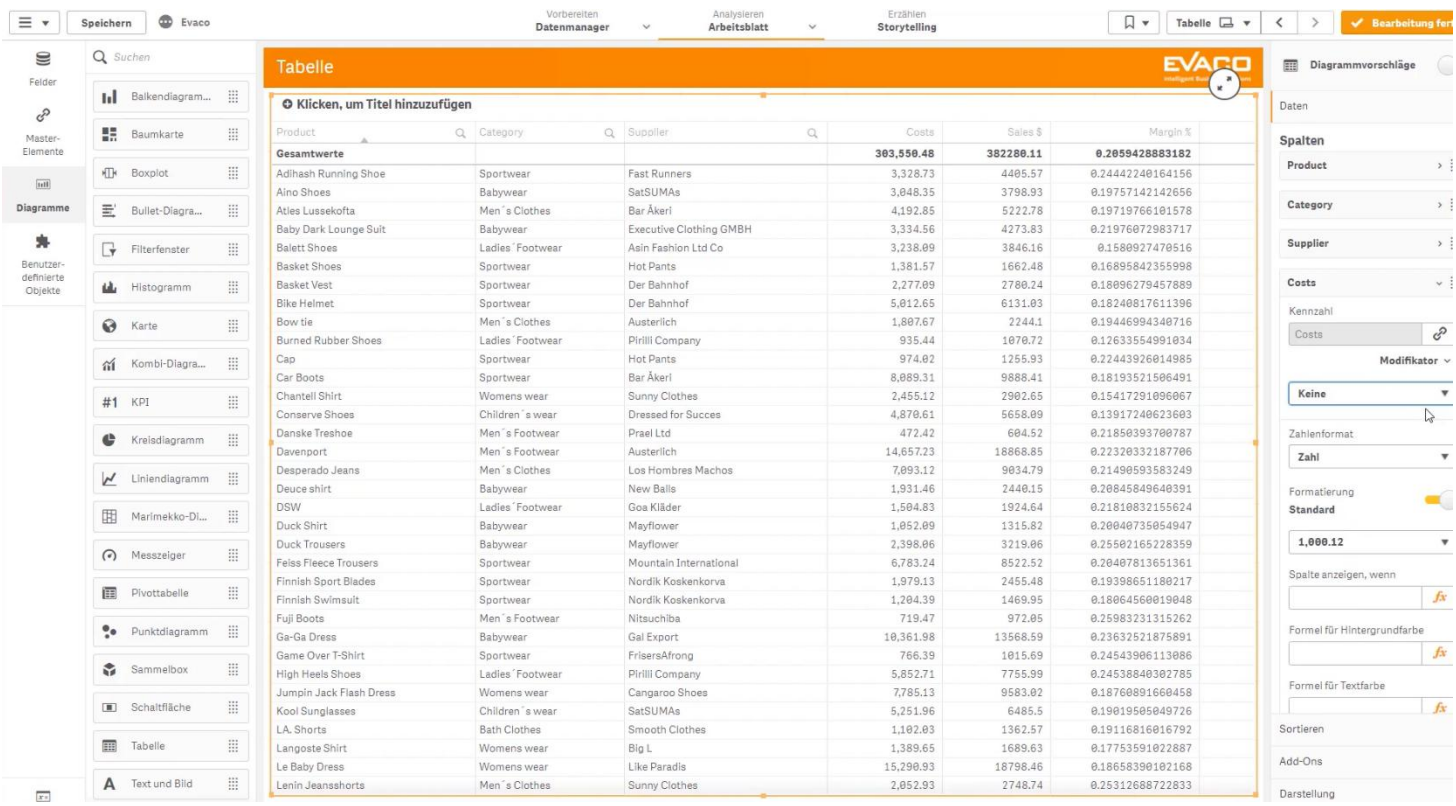

Abbildung 6: Front View base on Qlik

Teil 3: Fragen

Frage 1

❖ Welche **Datenquellen** sollen angebunden werden? (Interne/externe Datenbanken, APIs, etc.)?

Frage 2

❖ Welche Hauptfunktionen von Qlik müssen im neuen Modul auf jeden Fall erhalten bleiben?

Frage 3

❖ Gibt es zusätzliche Funktionen oder Verbesserungen, die im Vergleich zu Qlik erwartet werden?

Frage 4

❖ Gibt es besondere Anforderungen an die **Sicherheit oder Authentifizierung** der Datenzugriffe?

Frage 5

❖ Wie genau ist der Projektumfang? Geht es nur um die Visualisierung oder auch um Datenextraktion und -verarbeitung?

Frage 6

❖ Welche **Mindestfunktionen** sind für den Prototyp erforderlich?

- Einfache Tabellenanzeige ?
- Auswahl von Spalten ?
- Filtermöglichkeiten ?
- Extra spezielle (mathematische) Spalten
- Statistische Diagramme anzeigen (zum Beispiel Balkendiagramm, Kreisdiagramm, Liniendiagramm, Säulendiagramm, Punktediagramm)