Team 2 - DEPAZS

Digitale Energie Preis Auskunfts Zentralstelle

Jan Rüger, Laurin Scholtysik, Sven Stocker

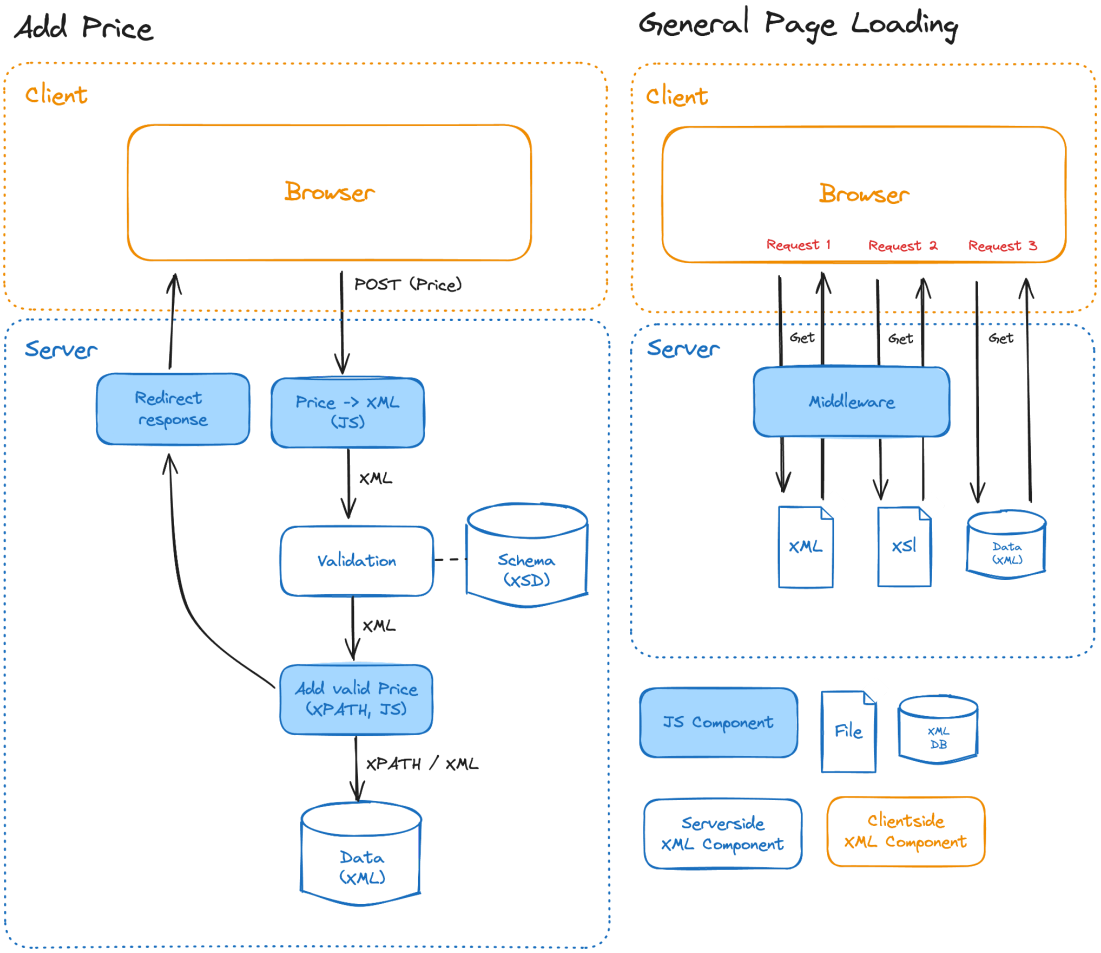
Konzept

Unsere digitale Plattform “DEPAZS” bietet eine innovative Lösung für die aktuellen Herausforderungen im Energiesektor, die sich aus der drohenden Energiekrise nach der COVID-19-Pandemie ergeben hat. In einer Zeit, in der die Energiepreise unaufhörlich steigen und die Energielieferanten mit der Bewältigung dieser Krise kämpfen, ermöglicht unsere Plattform eine effiziente Verbreitung der Preise und eine transparente Kommunikation zwischen den Energieunternehmen und der Bevölkerung.

Die Plattform richtet sich an zwei Hauptkunden: die Bevölkerung, die transparente Informationen über ihre Energiekosten erhalten möchte, und die Energiekonzerne, welche eine effiziente Verwaltung und Kommunikation der Preise benötigen. Unser Service bietet eine digitale Verwaltung und Kommunikation der aktuellen Ereignisse im Energiesektor und trägt dazu bei, die Interaktion zwischen Energieunternehmen und Verbrauchern zu verbessern.

Durch die Schaffung einer zentralen Verwaltungsstelle bieten wir eine kostengünstige Lösung, die die Kommunikation und Verwaltung der Preise optimiert. DEPAZS unterstützt Energieunternehmen dabei, Prozesse zu optimieren und Kosten zu senken, während sie Verbrauchern eine transparente und benutzerfreundliche Möglichkeit bietet, die aktuellen Energiepreise zu verfolgen.

Architekturdiagramm



Verwendete Frameworks

Framework / Library	Verwendungszweck(e)
express	• Webserver für Hosting von XML Files • I/O-Operationen mit dem DB-File • Anbindung HTTP an XSL
dotenv	Management von *.env*-Files (Umgebungsvariablen, z.B. Server Port)
morgan	Request Logger für Development & Debugging
libxmljs2	Library zum verarbeiten von XML-Files: • Parsing • Validierung
leaflet	Library zum Darstellen von interaktiven Maps
font-awesome	UI-Icon-Library
w3.css	CSS-Framework

Technische Stolpersteine

Die zwei unserer grössten technischen Stolpersteine waren letztlich auf den Custom Namespace in der Datenbank XML zurückzuführen:

Zuerst war es uns nicht möglich, im XSL-File die Datenbank mit *document()* korrekt zu laden und dann die Templates darauf anzuwenden. Dieses Problem konnten wir beseitigen, indem wir den Namespace im Pfad des *match*-Attributes des Template-Nodes angegeben haben.

Das zweite Problem hatten wir, als wir versuchten, das "Add Price" Feature zu implementieren. Dort funktionierte das Laden und Suchen mit der *libxmljs2* Library nicht. An diesem Zeitpunkt beschlossen wir, den Namespace unserer Datenbank zu entfernen, da dieser im aktuellen Use-Case keinen Mehrwert bringt (Schema wird lediglich Applikations-intern verwendet und ist keine öffentliche Schnittstelle) .

Bekannte Limitationen

Verwendet der User einen Firefox-Browser, so wird die Leaflet-Map auf der *Plant Overview* Seite nicht dargestellt. Auch nach einer detaillierten Analyse konnten wir nicht eruieren, wo der Fehler genau liegt. Dies zumal auch, da keine Exception oder Ähnliches in der JavaScript-Konsole erscheint und das Rendering der Map “still” fehlschlägt.

Diskrepanzen Kriterienkatalog

Laut Kriterienkatalog ist eine Konformität zu XHTML gefordert. Jene erfüllen wir auf allen Seiten bis auf *Plant Overview*. Sobald auf dieser Seite der default Namespace von XHTML gesetzt wird, crasht LeafletJS beim Aufbau der Map. Dies ist unter Anbetracht des Stacktrace und Caller Codes womöglich daran geschuldet, dass sich bei einem XHTML-Dokument die JavaScript *document*-APIs von denen von einem HTML5 Dokument unterscheiden. Da wir jedoch nicht auf die visuelle Map als zusätzliches Feature verzichten wollten, liessen wir den XHTML Namespace auf dieser Seite bewusst weg.

Nicht-XML Technologien

In den folgenden Bereichen wurde supplementär zu den XML-Technologien noch JavaScript (client- oder serverseitig) verwendet:

- **[Server] Web Server:** File Hosting & Bereitstellung statischer Assets
  Für das Hosting der XML-Files sowie die Bereitstellung von statischen Assets (z.B. CSS-Files) und der Datenbank wurde *express* verwendet, da sich ein solcher Use-Case nicht mit reinen XML-Technologien abdecken liesse. Hierbei wurden für ein professionelleres Auftreten zusätzlich die XML-Files manuell auf dedizierte Pfade gemappt, sodass die Dateiendungen in der URL nicht mehr ersichtlich sind.
- **[Server] Dynamische XSL Parameter:** Zugriff auf URL-Query-Argumente
  Um einen dynamischen Wechsel des Anzeige Jahres der Preis-Charts zu ermöglichen, musste mit der Express-Middleware *injectUrlParams* die Möglichkeit geschaffen werden, Platzhalter in XLS-Files dynamisch durch Argumente aus dem URL Query String zu ersetzen. Dies wäre mit reinen XML-Technologien nicht möglich gewesen.
- **[Server] Hinzufügen von Preisen:** Datenbankmanipulation & serverseitige Validierung
  Um einen neuen Preis-Datensatz aktiv gegen das Schema zu validieren und anschliessend im Datenbank-File (I/O Operation) einzufügen, wurde eine Express-API-Endpoint (*api-routes*) geschrieben. XML-Technologien besitzen diese Fähigkeiten nicht.
- **[Client] Leaflet:** Anbindung Map & Interaktion mit Layern
  Leaflet ist eine JavaScript Library zum Rendering von interaktiven Maps. Dementsprechend erfolgt das Laden und konfigurieren ebenfalls mit JavaScript Code, was die Abdeckung durch XML-Technologien ausschliesst. Der JS-Code liesse sich auch via XSLT generieren, jedoch macht dies keinen wesentlichen Unterschied. Deshalb entschieden wir uns aus semantischen Gründen für ein dediziertes JS-File (*plant-overview.js*) für die Leaflet-Anbindung.
- **[Client] Sidebar:** Navigation zu den einzelnen Features
  Für eine optisch ansprechende Navigation zwischen den Seiten wurde zusätzlich Javascript verwendet, um das Anzeigen und Verschwinden der Sidebar zu realisieren. Da es sich hierbei um dynamische DOM-Manipulationen handelt, wurde diese Funktionalität mit JavaScript anstatt XML-Technologien realisiert.

Fazit

Das Projekt war für alle Teammitglieder äusserst interessant und lehrreich. Zwar waren gewisse Namensraum-Herausforderungen ein bisschen nervend und zeitraubend, aber trotzdem empfanden wir die Arbeit am Projekt alles in allem positiv. Durch den Unterricht und das intensive Repetieren für die Prüfung haben wir viel gelernt. Dieses Wissen konnten wir im Projekt noch einmal zusätzlich erweitern und vertiefen.
Es überraschte uns alle, wie vielseitig XML ist und dass man mithilfe von XML-Technologien praktisch eine gesamte Webseite erstellen kann, auch wenn bestimmte Aspekte in der realen Welt wahrscheinlich so nicht umgesetzt werden würden.