Visualisierung der Praktikumsdaten im Praktikumstool

Okan Leenen

Jan Preuße

Inhaltsverzeichnis

Einleitung

Seit 2016 sammelt das Praktikumstool umfangreiche Daten bezüglich der Organisation und Umsetzung verschiedener Informatik-Praktika am Campus Gummersbach. Diese umfassenden Datensätze bergen beträchtliche Informationen, die bisher jedoch größtenteils ungenutzt geblieben sind.

Das vorliegende Projekt zielt darauf ab, Visualisierungen für das Praktikumstool zu erstellen und außerdem zu untersuchen, welche Arten der Visualisierung für die Daten des Praktikumstools sinnvoll und machbar sind. Auf Grundlage dieser Erkenntnisse soll ein Konzept entwickelt werden, welches Beispiele für Visualisierung der Daten aufzeigen und deren Nützlichkeit belegen soll.

Für die Realisierung der Visualisierung wird das Visualisierungs-Framework d3.js im Frontend verwendet.

D3js

D3, auch bekannt als D3.js, ist eine kostenfreie JavaScript-Bibliothek für die Visualisierung von Daten. Seit ihrer Entstehung im Jahr 2011 hat diese Open-Source-Bibliothek einen bemerkenswerten Beitrag zur Datenvisualisierung geleistet. Ihr Ansatz auf niedriger Ebene, basierend auf Webstandards, ermöglicht eine beispiellose Flexibilität bei der Erstellung dynamischer, datengesteuerter Grafiken.

D3 hat nicht nur wegweisende und preisgekrönte Visualisierungen hervorgebracht, sondern auch als Grundbaustein für höherstufige Diagramm Bibliotheken gedient. Die Bibliothek hat eine globale Gemeinschaft von Datenanalytikern inspiriert und gefördert.

Ihre Bedeutung zeigt sich auch in den erhaltenen Auszeichnungen, darunter der Information is Beautiful Test of Time Award 2022 und der IEEE VIS Test of Time Award 2021. Diese Anerkennungen betonen, wie D3 das Wachstum, die Diversifizierung und Kreativität in der Datenvisualisierung vorangetrieben und verändert hat, wie Millionen von Visualisierungen in verschiedenen Kontexten erstellt werden.

Ursprünglich von Mike Bostock ins Leben gerufen, erhielt D3 entscheidende Beiträge von Personen wie Jason Davies und Philippe Rivière. Die Bibliothek wird derzeit von Mike und Philippe bei Observable aktiv gepflegt. Insgesamt bleibt D3 eine zentrale und einflussreiche Kraft in der Welt der Datenvisualisierung, die die Art und Weise, wie Daten präsentiert und verstanden werden, maßgeblich beeinflusst hat.

PG

pg, auch bekannt als pg-promise, ist eine vielseitige JavaScript-Bibliothek, die speziell für die Arbeit mit PostgreSQL-Datenbanken in Node.js-Umgebungen entwickelt wurde. Seit ihrer Veröffentlichung im Jahr 2014 hat sich pg-promise durch ihre Unterstützung von Promises und asynchroner Programmierung als wertvolles Werkzeug für Entwickler etabliert. Die Bibliothek ermöglicht effiziente und reaktionsschnelle Anwendungen und bietet Funktionen wie Transaktionen, benutzerdefinierte Query-Formatierung und Verbindungspooling. Diese Funktionen erleichtern die Verwaltung komplexer Datenbankoperationen erheblich.

Die kontinuierliche Pflege und Weiterentwicklung durch Vitaly Tomilov, den Hauptentwickler, sowie eine engagierte Community sorgen dafür, dass pg-Promise stets auf dem neuesten Stand bleibt und regelmäßig aktualisiert wird. Dank dieser aktiven Unterstützung und der umfassenden Dokumentation hat sich pg-promise zu einem essentiellen Werkzeug für Entwickler entwickelt, die robuste und skalierbare Datenbanklösungen suchen. Die Bibliothek ist besonders geschätzt für ihre Stabilität und Flexibilität, was sie zu einer bevorzugten Wahl in der Node.js-Entwicklung macht.

Konzept

Das Informatik-Projekt beschäftigt sich mit der Visualisierung der Daten aus der Datenbank, die hinter dem Praktikumstool steht.

In diesem Projekt werden Daten aus verschiedenen Tabellen der Postgresql-Datenbank extrahiert, um daraus durch visualisierung einen Mehrwert zu schaffen, indem zum Beispiel Muster in den Daten Aufschluss über Schwierigkeitsgrad von Meilensteinen oder gar der Praktika einzelner Professoren.

Diese Zusammenfassung bietet einen Überblick über die wichtigsten Schritte und Variablen, die im Rahmen des Projekts verwendet werden.

Die Datenbank besteht aus mehreren Tabellen, die jeweils spezifische Informationen für das Praktikumstool bereitstellen. Im folgenden Abschnitt wird festgehalten, welche Tabellen wir für unsere Abfragen genutzt haben:

- **Courses**: Diese Tabelle enthält Informationen zu den Modulen, für uns von besonderem Interesse ist dabei der Fremdschlüssel LECTURER
- **Degrees**: Enthält die Studiengänge.
- **Users**: Enthält Benutzerdaten. Von besonderem Interesse sind hier für uns FIRSTNAME und LASTNAME, da wir daraus die Namen der Professoren zusammenbauen können, um sie als Label an einer Grafik zu verwenden.
- **Semesters**: Speichert Informationen über die verschiedenen Semester, in denen Kurse und Laborarbeiten angeboten werden. Wie beschränken uns auf das Wintersemester 2021/2022.
- **Labwork (Praktika)**: Diese Tabelle gibt Auskunft über die einzelnen Praktika, die in einem Semester für einen bestimmten Studiengang stattfinden. Diese Tabelle enthält mehrere Fremdschlüssel, die wir nutzen können, um nach bestimmten Labworks zu filtern, nämlich SEMESTER, COURSE und DEGREE.
- **Report_Card_Entry (Notenhefte)**: Hier werden die Notenhefte für einen Studierenden in einem bestimmten Praktikum erfasst.
- **Report_Card_Entry_Type (Testate, Anwesenheiten)**: Diese Tabelle enthält Informationen zu Testaten, Anwesenheiten usw. im Zusammenhang mit einem bestimmten Eintrag im Notenheft (Report_Card_Entry), beispielsweise ob ein Testat für Aufgabe 1 vergeben wurde oder Angaben zur Anwesenheit. Von besonderem Interesse sind hier ENTRY_TYPE und BOOL, da wir aus den beiden Werten zusammen ableiten können, ob jemand an einem Meilenstein teilgenommen hat oder ob es sich bei einem Eintrag überhaupt um einen Pflichttermin gehandelt hat.
- **Report_Card_Evaluation**: Enthält Bewertungen und Kommentare zu den Leistungsnachweisen der Studierenden. Von besonderem Interesse ist für uns hier BOOL, da es anzeigt, ob das Praktikum bestanden wurde.
- **Labworkapplications (Praktikazusage)**: Diese Tabelle gibt Auskunft über die einzelnen Praktikaanmeldungen, die in einem Semester für einen bestimmten Studiengang stattfinden.

Diese Tabellen ermöglicht eine Erfassung und Verwaltung von Informationen im Praktikumstool, angefangen bei den Modulen und Studiengängen bis hin zu den einzelnen Praktika, Notenheften und den zugehörigen Testaten und Anwesenheitsdaten.

Implementierung

Im Verlauf des Projekts wurden mehrere Schritte unternommen, um die API der Datenbank, die hinter dem Praktikumstool steht, zu simulieren. Der erste wichtige Schritt war die Installation von PostgreSQL und pgAdmin, um das dumpfile der Datenbank importieren und auf die Daten zugreifen zu können. Mithilfe von pgAdmin ließ sich auch die Struktur einfacher erkennen und erforschen. Anschließend wurden die Dump Files in die lokal laufende Postgresql-Instanz importiert.
Im postgreSQL-Tool unter pgAdmin wurde folgender Befehl ausgeführt:

```
pg_restore -U postgres -d praktikumstool \i
'C:/Users/user/Desktop/Infoprojekt/dump.sql'
```

Unser erster Schritt der API-Simulation bestand in der Extraktion von Daten aus verschiedenen Tabellen der Datenbank mithilfe der Kommandozeile. Diese extrahierten Daten wurden zunächst in JSON-Dateien, also dem standardmäßigen Rückgabeformat von APIs, gespeichert.

Die Abfrage hierzu:

```
SELECT * FROM "REPORT_CARD_ENTRY_TYPE" AS JSON LIMIT 100;
```

Diese Extraktion ermöglichte die Umwandlung der Daten aus der Datenbank in ein Format, welches auch von den meisten APIs verwendet wird. Mit diesen Daten konnten wir also den API-Zugriff emulieren.
Im späteren Verlauf sind wir aber von diesem Konzept weg und dazu übergegangen, unseren eigenen API-Overhead, zu programmieren, statt mit einer kleinen Menge extrahierter Daten zu arbeiten. Dazu lassen wir einen node.js Express-Server laufen, dessen Endpunkte vom Frontend abgerufen werden können. Die Endpunkte triggern eine Funktion, die eine SQL-Query über die Datenbank laufen lässt und die Antwort in JSON. Für die Queries ist jeweils ein Endpunkt angelegt. Die Grundlage für die Implementierung der Visualisierungen bilden die Abfragen, die über verschiedene Tabellen der Datenbank laufen. Diese Tabellen, darunter:

- **COURSES**
- **DEGREES**
- **LABWORK**
- **LABWORKAPPLICATIONS**
- **REPORT_CARD_ENTRY**
- **REPORT_CARD_ENTRY_TYPE**
- **REPORT_CARD_EVALUATION**
- **USERS**
- **SEMESTERS**

enthalten grundlegende Informationen, die für Studenten sowie Professoren relevant sind.

Probleme

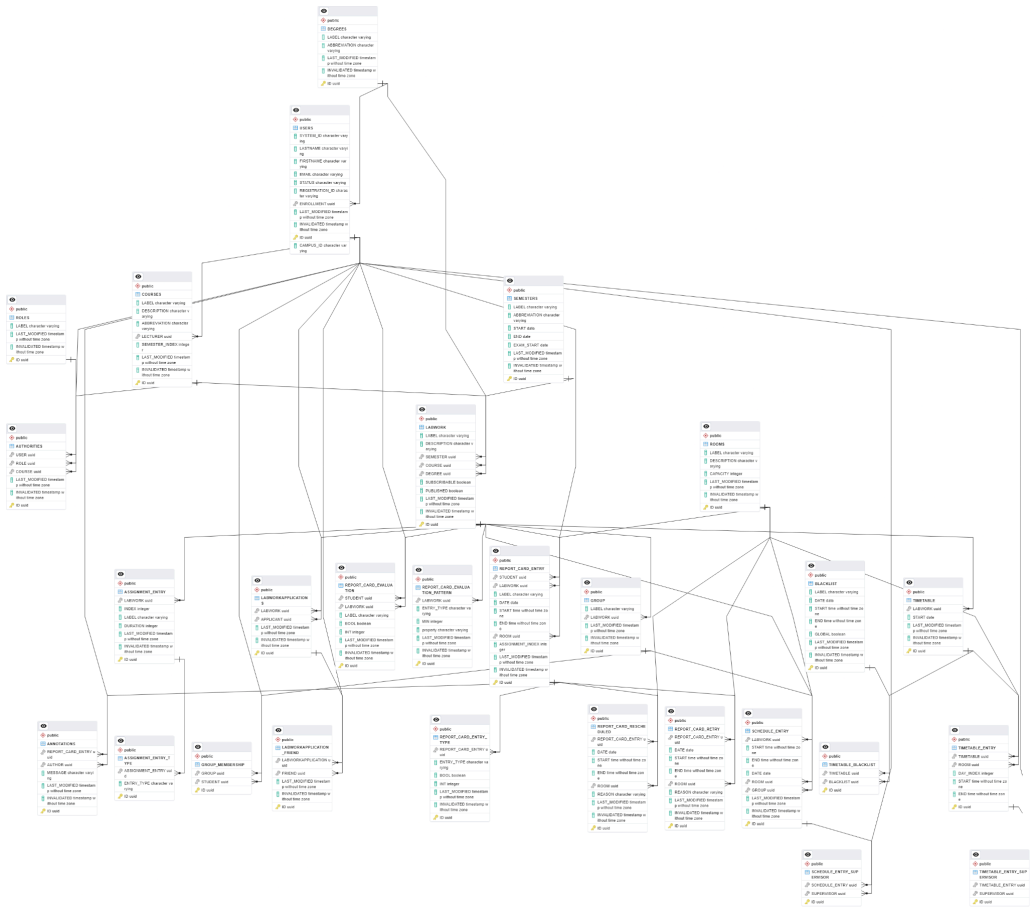
Von dem erhaltenen Dumpfile bis zur Implementierung der Visualisierung traten mehrere Herausforderungen auf.

Zuerst musste aus dem Dumpfile die Datenbank wiederhergestellt werden. Dies gestaltete sich komplizierter als erwartet, da die Daten zunächst in einem JSON-Format hinzugefügt und nach den Anforderungen von PostgreSQL umgeschrieben werden mussten.

Anschließend konnte man nicht einfach auf Import klicken und das Dumpfile hinzufügen. Stattdessen musste man über die Kommandozeile von pgAdmin zum Ordner navigieren und wie in der Implementierung beschriebenen Befehl ausführen:

```
pg_restore -U postgres -d praktikumstool \i
'C:/Users/user/Desktop/Infoprojekt/dump.sql'
```

Nach dem Importieren der Daten sahen wir uns die Struktur der Hochschule Köln-Datenbank an und erkannten bald, dass nicht nur komplexe Queries, sondern auch Stunden der Suche und des Rätselns vor uns standen. Ein Screenshot der gesamten Datenbank als Graph, automatisch generiert von pAdmin, verdeutlichte die Herausforderungen.



Man könnte wohl sagen, dass die Datenbank historisch gewachsen ist, und damit die Probleme.

Im nächsten Schritt beabsichtigten wir, die genannten Tabellen als JSON zu exportieren, um mit der Implementierung der Visualisierung zu beginnen.

Aufgrund der großen Anzahl von Einträgen haben wir den Export auf 100 begrenzt. Das nachfolgende SQL-Statement wurde für den Export verwendet:

```
SELECT * FROM "REPORT_CARD_ENTRY_TYPE" AS JSON LIMIT 100;
```

Diese Schritte verdeutlichen die Probleme, die während des gesamten Prozesses von der Datenaufbereitung bis zur Implementierung aufgetreten sind.

Beim Aggregieren der Daten sind wir vor allem auf ein großes Problem gestoßen, nämlich die Identifikation der durchgefallenen Studenten. Leider gibt es keine direkt ersichtliche Darstellung dazu in der Datenbank. Nach Rücksprache mit dem Projektbetreuer konnten wir aber eine mehr oder weniger zuverlässige Methode zur Klassifizierung erstellen.

Fazit

Während des gesamten Projekts, von der Datenextraktion bis zur Implementierung der Visualisierungen, traten zahlreiche Herausforderungen auf. Die importierten Daten mussten zunächst in ein für PostgreSQL kompatibles Format umgewandelt werden, was zusätzlichen Aufwand erforderte. Der komplexe Aufbau der Datenbankstruktur stellte ebenfalls eine Herausforderung dar, die jedoch durch sorgfältige Analyse und Zusammenarbeit überwunden wurde.

Schließlich haben wir verschiedene Visualisierungstechniken implementiert, um die gesammelten Daten sinnvoll darzustellen. Trotz der aufgetretenen Schwierigkeiten konnten wir wertvolle Einblicke gewinnen und die Grundlage für eine effektive Visualisierung der Praktikumsdaten schaffen.

Diese Visualisierungen können sowohl für Professoren als auch für Studenten von Nutzen sein. So können Professoren beispielsweise ablesen, welche Meilensteine oder Praktika zu einfach oder zu schwer sind und danach gegebenenfalls anpassen beziehungsweise die Auswirkungen von Veränderungen oder äußeren Auswirkungen, wie z.B. der Pandemie.

Auch Studenten können aus den Visualisierungen der Daten Nutzen ziehen. Die Frage zum Beispiel, welcher Professor die meisten Studenten durchfallen lässt, könnte die eine oder andere Entscheidung, welches Wahlpflichtmodul man auswählt, beeinflussen, und tut das heute schon. Heute allerdings basierend auf Gerüchten, in der Zukunft vielleicht auf basierend auf harten Fakten.

Für die Visualisierung würden wir jedoch eher eine andere Library empfehlen, wie z.B. chart.js. Zwar kann man mit d3.js sehr viel mehr machen, allerdings besteht die Funktionalität zur Erstellung sogar von einfachen Graphen bereits aus rund 100 Zeilen Code.