



LV-Titel: Wissenschaftliches Projekt

LV-Leiter: Prof. Dr. Stefan Edlich

Semester: Wintersemester 2012/13

Studiengang: Online Medieninformatik Master

Projektdokumentation:

Google Fusion Tables:

Daten-Cockpit für Berliner Weihnachtsmärkte 2012

Projektteam/Verfasser:

Patrick Böhm (783318), E-Mail: patty.boehm@gmail.com

Manuel Wisniewski (786598), E-Mail: manuel.wisniewski@gmail.com

letzte Änderung: 10.02.2013

DOKUMENT-HISTORIE

Version	Datum	Änderungen	Bearbeiter
0.1	13.11.2012	Projektbeschreibung in GoogleDocs angelegt	Patrick Böhm, Manuel Wisniewski
1.0	02.12.2012	Dokumentation begonnen	Patrick Böhm, Manuel Wisniewski
1.1	15.12.2012	Inhalte Strukturiert	Patrick Böhm
1.2	16.12.2012	Formatierung angepasst	Manuel Wisniewski
1.3	18.12.2012	Projektbeschreibung und Informationen zu Kapitel 2 ergänzt	Patrick Böhm
1.4	6.1.2013	Kapitel 4.2 <i>Umsetzung</i> hinzugefügt	Manuel Wisniewski
1.5	11.01.2013	Google Fusion Tables Web Apps und API-Beschreibung, Erfahrung dokumentiert.	Patrick Böhm
1.6	24.01.2013	Ergänzungen Kapitel 2 „Google Fusion Tables“	Manuel Wisniewski
1.7	25.01.2013	Anpassung Struktur	Patrick Böhm, Manuel Wisniewski
1.8	26.01.2013	Ergänzung HTML-Prototypen, Anpassung im Kapitel 3.	Patrick Böhm
1.9	05.02.2013	Überarbeitung Kapitel 1 & 2 (komplett) und 3 (begonnen)	Manuel Wisniewski
1.10	10.02.2013	Abschluss Kapitel 4.5, 4.6, 4.7	Patrick Böhm
1.11	10.02.2013	Kapitel 5, Überarbeitung, Verzeichnisse	Manuel Wisniewski

ZEITPLANUNG

Meilenstein	Plan	Ist
Einarbeitung	01.11.2012 - 13.11.2012	13.11.2012
Spezifikation / Lastenheft	14.11.2012 - 30.11.2012	04.12.2012
Vorbereitung Datenbeschaffung	01.12.2012 - 13.12.2012	13.12.2012
Umsetzung	14.12.2012 - 31.12.2012	15.01.2013
Dokumentation der verwendeten Technologien	01.01.2013 - 14.01.2013	10.02.2013
Abschlusspräsentation	28.01.2013	11.02.2013

ABBILDUNGSVERZEICHNIS

Abbildung 1: Nutzungswege von Google Fusion Tables.....	11
Abbildung 2: Google Fusion Tables Web App (New Look).....	12
Abbildung 3: Google Fusion Tables API Reference.....	13
Abbildung 4: Google Maps Info-Window.....	15
Abbildung 5: Wireframe des Weihnachtsmarkt-Datencockpits.....	20
Abbildung 6: Berlin Open Data-Portal der Senatsverwaltung Berlin.....	22
Abbildung 7: Neue Tabelle importieren (1): Importeinstellungen.....	26
Abbildung 8: Neue Tabelle importieren (2): Erfolgreicher Import.....	27
Abbildung 9: Balkendiagramm: Anzahl Weihnachtmärkte pro Bezirk.....	28
Abbildung 10: Markierung der Adressdaten in Google Maps.....	29
Abbildung 11: Anzeige von Polygondaten in Google Maps.....	30
Abbildung 12: Polygon-Flächendarstellung in Google Maps.....	30
Abbildung 13: Menü: "Tools - Change mal styles..."	30
Abbildung 14: Change map styles: Show a gradient.....	31
Abbildung 15: Heatmap: Anzahl Weihnachtmärkte pro Bezirk.....	32
Abbildung 16: Prototyp: Filterlisten.....	34
Abbildung 17: Prototyp: Eingebettete Google Map mit Positionsmarkern.....	35
Abbildung 18: Geokodierbare Location-Angaben in der Spalte "geo_adresse"	36
Abbildung 19: Überlagerte Darstellung mehrerer Layer in Google Maps.....	40
Abbildung 20: Finaler Prototyp.....	41

TABELLENVERZEICHNIS

Tabelle 1: HTTP-Requests für die Ressourcen Tabelle, Spalte, Template und Style.....	16
Tabelle 2: SQL-Queries für HTTP-Request zum Auslesen von Zeilendaten.....	16
Tabelle 3: Weihnachtsmarktdetails des Berlin Open Data-Portals.....	23
Tabelle 4: Spalten der Tabelle „Wibo - Weihnachtsmarkt Finder 2012 – Weihnachtmärkte“	25
Tabelle 5: Spalten der Tabelle: „Wibo - Weihnachtsmarkt Finder 2012 - Bezirke Auswertung“.....	25

INHALTSVERZEICHNIS

1 Einleitung.....	5
2 Google Fusion Tables.....	6
2.1 Systemarchitektur.....	6
2.1.1 Speicher-Stack.....	7
2.1.2 Anwendungsdatenspeicher.....	7
2.1.3 Datenabfrage.....	8
2.1.4 Transaktionsmanagement.....	8
2.2 Datenvisualisierung.....	9
2.2.1 Clientseitige Visualisierung von Diagrammen.....	9
2.2.2 Serverseitige Visualisierung von geografischen Features.....	9
2.3 Entwicklungsumgebung und API.....	10
2.3.1 Google Fusion Tables Web App.....	10
2.3.2 Google Fusion Tables API 1.0.....	12
3 Anwendungsspezifikation	18
3.1 Motivation.....	18
3.2 Suchfunktionalität	19
3.3 Schematischer Entwurf.....	20
3.4 Entwicklungsinfrastruktur.....	21
4 Technische Umsetzung.....	22
4.1 Datenerhebung.....	22
4.2 Datenimport und -aufbereitung.....	24
4.2.1 Exkurs: Probleme beim Importieren der Daten.....	26
4.3 Konfiguration der Visualisierungen.....	27
4.3.1 Diagramme mit Fusion Tables.....	27
4.3.2 Karten und Heatmaps mit Fusion Tables.....	29
4.4 HTML-Prototyp.....	32
4.4.1 Dynamische Auswahllisten.....	33
4.4.2 Google Map und Fusion Tables.....	35
4.5 Interaktive Anzeigesteuerung.....	37
4.5.1 Filterfunktionalität.....	37
4.5.2 Einblenden der statistischen Werte.....	39
4.6 Einschränkungen/Ausblick.....	40
5 Zusammenfassung.....	42

1 EINLEITUNG

Fusion Tables ist Googles umfangreiches Web-Toolkit zur Speicherung, Verwaltung und Visualisierung (alpha)numerischer Daten. Durch die Verknüpfung mit anderen Google Services – allen voran Google Maps – ergeben sich interessante Anwendungsmöglichkeiten, insbesondere durch die Verknüpfung statistischer Daten mit Geodaten in der kartografischen Projektion. Das Ergebnis sind visuell aufbereitete, interaktive Datenbilder bzw. Landkarten, die neue und aufschlussreiche Einblicke in das Datenmaterial ermöglichen.

Das vorliegende Projekthandbuch dokumentiert die Konzeption und Entwicklung einer Fusion Tables-Anwendung in Form einer Informationsplattform für Berliner Weihnachtsmärkte: Auf einer Berlinkarte werden zunächst alle gemeldeten Weihnachtsmärkte der Saison 2012 angezeigt. AnwenderInnen können dann die Zielmenge anhand mehrerer Kriterien (z.B. nach Bezirk, Eintrittspreis, etc.) nach belieben einschränken und einige beispielhafte statistische Auswertungen vornehmen.

Die Umsetzung des Projektes beginnt mit der theoretischen Skizzierung der Google Fusion Tables-Infrastruktur (Kapitel 2) sowie einer groben Spezifikation der Datencockpit-Anwendung (Kapitel 3). Im Rahmen der Entwicklung müssen dann zunächst die notwendigen Daten und insbesondere die Geokoordinaten beschafft und in Fusion Tables importiert werden. Im Anschluss werden die Datentabellen mit Google Maps kombiniert und in einen HTML-Prototypen integriert (Kapitel 4). Betrachtungen zu den Stärken und Schwächen von Fusion Tables sowie reflektierende Überlegungen zur Arbeit mit der Architektur schließen das Projekt ab (Kapitel 5).

Zu den vollständigen Projektergebnissen gehören neben der vorliegenden Dokumentation sowohl der Prototyp des Datencockpits als auch die aufbereiteten Datentabellen. Die Quelltexte, Verweise und Links zu den eingesetzten Ressourcen sind im projektbegleitenden github-Repository vollständig aufgelistet:

- Datencockpit Weihnachtsmarkt-Finder 2012: <http://tinyurl.com/xmas-city>
- github-Repository: https://github.com/borste/WiBo_GoogleFusionTables.

2 GOOGLE FUSION TABLES

Fusion Tables ist Googles 2009 gestarteter cloud-basierter Webdienst zur Verwaltung, Integration und insbesondere zur Visualisierung tabellenbasierter Daten. Der Service unterstützt AnwenderInnen und Unternehmen bei der internen wie externen Bereitstellung von Daten und hat zum Ziel, die Zusammenarbeit von NutzerInnen aus verschiedenen Organisationseinheiten zu vereinfachen. Für eine möglichst benutzerfreundliche Datenauswertung hält das Framework Filter- und Aggregationsfunktionalitäten bereit und schlägt auf Basis der hinterlegten Daten automatisch passende Visualisierungsformen vor. Neben herkömmlichen Diagrammen ist speziell auch die geografische Projektion der Daten in das Kartenmaterial der Geodatendienste Google Maps und Google Earth möglich. UserInnen können ihre Daten zudem für die Zusammenarbeit mit anderen AnwenderInnen (und für Suchmaschinen) freigeben und dank einer Diskussionsfunktion sogar bis auf Zellenebene detailliert kommentieren¹. Google Fusion Tables erlaubt dabei den Upload verschiedenster Datenformate (csv, kml, etc.) über den Service Google Drive (bis zu 250 MB). Neben der Integration verschiedener Datenquellen können zu Auswertungszwecken auch Datentabellen verschiedener NutzerInnen verbunden werden.

Gemäß [4] folgt die Entwicklung von Fusion Tables drei zentralen Leitlinien:

1. Vereinfachung der Datenverwaltung für AnwenderInnen ohne Erfahrung im Umgang mit Datenbanksystemen,
2. Erhöhung der Attraktivität von Data-Sharing und Informationsintegration durch aussagekräftige Visualisierungen sowie
3. Unterstützung von cloud-basierter Kollaboration, die über reine Datenabfrage hinausgeht, z.B. durch Diskussions- und Abstimmungsfeatures.

Die folgenden Abschnitte skizzieren grob die grundlegende Systemarchitektur des Fusion Tables Frameworks sowie die Visualisierungskonzepte und den groben Aufbau der Entwicklungsschnittstelle *Fusion Tables API v1.0*.

2.1 Systemarchitektur

Die Systemarchitektur von Fusion Tables gliedert sich in drei Schichten: *Frontend Dispatcher*, *Query Processing Module* und *Backend*. Anfragen (Requests) können grundsätzlich von verschiedenen Clients gestellt werden (Fusion Tables-Website, Stand-Alone-Anwendungen, eingebettete Karten, etc.). Der Frontend Dispatcher übersetzt die Anfragen und leitet die standardisierten Queries an das Query Processing Module weiter. Das Processing Module erstellt einen Query Plan und über gibt diesen ans Backend. Im Backend werden die Queries abgearbeitet und die Ergebnisse über

¹ Der Fusion Tables Entwicklungsstatus ist gegenwärtig noch experimentell und bietet zwei Darstellungsoptionen für die Anwendungsoberfläche im Web: *Classic Look* und *New Look*. Die Kommentarfunktion steht derzeit nur im *Classic Look* zur Verfügung. Ob und wann eine solche Funktionalität auch im angepassten UI zur Verfügung steht, bleibt abzuwarten.

das Query Processing Module schlussendlich wieder an das Frontend zurückgeliefert. Als Persistenzschicht dienen dabei synchron replizierte Bigtable-Server (siehe Kapitel 2.1.1). Die größte Herausforderung der Datenbankschicht besteht in der Bearbeitung hunderttausender Tabellen mit unterschiedlichen Schemata, Größen und Anfragelast-Statistiken.

2.1.1 Speicher-Stack

Innerhalb des Google Speicher-Stacks operiert die Fusion Tables-Architektur auf den beiden Ebenen *Bigtable* und *Megastore*:

1. *Bigtable* ist Googles proprietäres Hochleistungs-DBMS und speichert in erster Linie Tabellen als komplexe und nicht zwingend normalisierte (key, value)-Tupel. Diese werden nach Schlüsselwerten sortiert über mehrere Server verteilt. Jede Tabelle ist mehrdimensional, wobei jeder Eintrag mithilfe eines Timestamps versioniert wird.
2. *Megastore* ist ein auf Bigtable aufbauendes Speichersystem, das die Vorteile von NoSQL-Datenbanken (in erster Linie die Skalierbarkeit) mit den Vorzügen traditioneller relationaler DBMS verbindet. Megastore bietet u.a. konsistente Sekundärindizes, mehrzeilige Transaktionen und konsistente Replikation.

Übersichtliche Ansatzpunkte für die ausführliche Beschreibung von Bigtable und Megastore-Architektur liefern [2] und [1].

2.1.2 Anwendungsdatenspeicher

Der Anwendungsdatenspeicher der Fusion Tables-Infrastruktur unterteilt sich in vier Bereiche: Zeilenspeicher (*Row Store*), Schema-Speicher (*Schema Store*), Sichten-Speicher (*View Store*) und Kommentar-Speicher (*Comment Store*).

1. Zeilenspeicher:

Die Zeilen aller Datentabellen werden gemeinsam in einer einzigen Bigtable „Rows“ gespeichert. Dabei repräsentiert jede Zeile genau eine Zeile in einer Nutzertabelle. Der Primärschlüssel einer Zeile wird aus der individuellen, intern generierten Tabellen-ID und der jeweiligen Zeile zusammengesetzt. Dank der guten Skalierbarkeit der Bigtable-Architektur bleibt der Zugriff auf die „Rows“-Tabelle auch bei Millionen von Anwendertabellen performant, da die Tabelle in einzelne Tabellenfragmente aufgeteilt wird, die wiederum auf verschiedenen Servern lagern.

Eine Tabellenzeile besteht aus dem Zeilen-Schlüssel sowie je einem Satz indizierter und nicht-indizierter Properties. Ein Property setzt sich aus Property-Name und Property-Wert zusammen. Da folglich jeder Property-Name redundant in der „Rows“-Tabelle hinterlegt wird, können in den einzelnen Zeilen unterschiedliche Property-Sets – d.h. unterschiedliche Tabellenschemata – gespeichert werden. Standardmäßig werden Property-Werte als

Strings gespeichert. Ein Annotationsservice identifiziert automatisch Datentypen wie Zahlen, Datum oder Geodaten. Je nach Datentyp schlägt das Fusion Tables-Framework in weiterer Folge passende Visualisierungsformen vor. Zum effizienteren Zugriff werden ausgewählte Properties indiziert.

2. Schema-Speicher:

Tabellenschemata von Benutzertabellen werden ebenfalls gemeinsam in einer zentralen Bigtable gespeichert. Dabei dient die Tabellen-ID als Primärschlüssel. Für jede Tabelle werden Informationen zu Spalten und Berechtigungen abgelegt, dazu gehören Spaltenname, Datentyp und Access Control Lists (ACLs). Dabei werden für jede Tabelle separate Listen für NutzerInnen mit Lese- (Viewer) und/oder Schreibberechtigung (Collaborator) geführt. Öffentliche Tabellen haben einen speziellen Vermerk, dass sie für jeden sichtbar sind. Tabellenschemata können verändert, Spalten können hinzugefügt oder entfernt und automatisch identifizierte Datentypen können nach Belieben angepasst werden.

3. View-Speicher:

Einzelne Tabellen verschiedener BenutzerInnen können in Fusion Tables zusammengeführt werden. Zusammengesetzte Tabellen werden *Views* genannt und nicht materiell repliziert, sondern verweisen auf die verlinkten Ausgangstabellen. Lediglich ihre Zusammensetzung und Zugriffsrechte werden gespeichert, wobei die gleichen Anwenderrollen gelten wie bei Tabellen, erweitert nur um die Contributor-Rolle für NutzerInnen, welche die View-Definition editieren können.

4. Kommentar-Speicher:

NutzerInnenkommentare zu Datensätzen werden ebenfalls in einer einzigen Bigtable zusammengefasst. Als Schlüssel dient hierbei das 3er-Tupel (Tabelle, Zeile, Spalte). Gespeichert werden der Kommentartext sowie Autor und Datum des Kommentars.

2.1.3 Datenabfrage

Der Tabellenzugriff in Fusion Tables erfolgt mittels SQL. Entsprechende Queries können aus klassischen Abfragen (SELECT), Aggregat-Funktionen (GROUP BY) und Verknüpfungen (JOIN) zusammengestellt werden. Aus Performancegründen werden die SQL-Statements auf Standardoperationen von Bigtable (*Key Lookup*, *Prefix Scan* und *Range Scan*) gemappt. Dabei kommt ein Property-Index zum Einsatz, der in einer eigenen Tabelle alle Schlüssel zusammenfasst. Eine Überblicksdarstellung der gängigen Zugriffsmodi liefert [3].

2.1.4 Transaktionsmanagement

Da Fusion Tables in erster Linie als Visualisierungsplattform eingesetzt wird und nicht auf großen Datendurchsatz ausgelegt ist, kommt dem Transaktionsmanagement eine niederrangige Rolle zu.

Im Rahmen von Megastore erfolgt Transaktionskontrolle daher lediglich auf Tabellenebene mittels *Write Ahead Logging (WAL)* und optimistischem Locking (*Optimistic Concurrency Control*) zur Gewährleistung von Atomarität und Dauerhaftigkeit. Für eine detailliertere Darstellung der Transaktionskontrolle sei ebenfalls auf [3] verwiesen.

2.2 Datenvisualisierung

Die zentrale Funktionalität von Fusion Tables ist die Datenvisualisierung. Verfügbare Visualisierungsmethoden werden auf Grundlage der verwendeten Datentypen automatisiert berechnet. So sind beispielsweise Streudiagramme nur verfügbar, wenn mindestens zwei Spalten mit numerischen Werten vorhanden sind. Analog dazu ist die Kartenvizualisierung nur auf Basis von Geodaten möglich. Bei der Visualisierung werden zwei Konzepte unterschieden: clientseitige Visualisierung für die Darstellung von Diagrammen und serverseitige Visualisierung für das Rendern von geografischen Darstellungen in Google Maps.

2.2.1 Clientseitige Visualisierung von Diagrammen

Die clientseitige Visualisierung wird mittels Google Visualization API realisiert. Das Rendern der Visualisierung erfolgt mittels Javascript oder Flash direkt im Browser. Über das Fusion Tables-Framework können dabei Datentabellen abgefragt und Ergebnismengen direkt an die Visualisierungsinfrastruktur übergeben werden. Zudem können AnwenderInnen ihre Visualisierungen individuell konfigurieren, wobei das Framework anhand der zur Verfügung stehenden Daten automatisch Vorschläge generiert. Erstellte Datendiagramme können durch Einsetzen weniger Zeilen JavaScript-Code in beliebige Webseiten eingebunden werden. Die Visualisierungen sind dabei direkt mit der Datenquelle verbunden und werden bei Änderungen im Datensatz automatisch aktualisiert.

2.2.2 Serverseitige Visualisierung von geografischen Features

Für geografische Visualisierungen über den Service Google Maps wird hingegen die serverseitige Berechnung der Darstellung angewandt. Dabei werden geokodierbare Daten (in erster Linie Adressen) auf sogenannte *Map Layers* gerendert. Die Repräsentation von Daten in Google Maps erfolgt stets in Form von Layern, die sich aus einzelnen Bildkacheln (*Tiles*) zusammensetzen. Die Entscheidung für eine serverseitige Berechnung liegt im erheblichen Datenaufwand begründet, der ohne Kenntnis über die aktuellen Anzeigeeinstellungen zur Visualisierung an den Client übertragen werden müsste, sowie in der begrenzten Leistungsfähigkeit der Browser. Fragt die/der AnwenderIn die Anzeige einer Karte bzw. Map an, wird ein entsprechender Request zum Backend-Server mit Informationen über die aktuell sichtbaren Layer, die Geokoordinaten und den gegenwärtigen Zoom Level gesendet. Das Backend setzt nun aus allen Layer-Informationen einzelne Bild-Tiles zusammen und sendet diese zurück an den Browser.

In Fusion Tables werden Geodaten in einen Raum-Index (*spatial index*) eingetragen. Für die Arbeit

mit Fusion Tables bedeutet dies, dass zwischen strukturierten (*structured*) und räumlichen (*spatial*) Datenabfragen unterschieden wird. Bei Abfragen wie z.B. „alle Weihnachtsmärkte in Berlin (*spatial query*) ohne Eintritt (*structured query*)“ werden beide Anfragen parallel verarbeitet und die Ergebnismengen verschränkt. Structured queries werden klassisch mittels SQL beantwortet während spatial queries mithilfe des spatial index erfolgen. Für Details zum räumlichen Mapping von Informationen auf einzelne Tiles und zum genauen Verfahren der spatial queries sei an dieser Stelle auf [3] verwiesen.

Um schnelle Antwortzeiten der Kartenvisualisierung zu ermöglichen, ist die Anzahl der darstellbaren Features auf einer Tile begrenzt. Im Streben um bessere Übersichtlichkeit trifft das Visualisierungs-frame-work eigenständig eine Auswahl darzustellender Details in Abhängigkeit vom aktuellen Zoom Level. Fusion Tables erlaubt auch das unkomplizierte Rendern von *Heat Maps*, wobei unterschiedliche Kartenbereiche in Abhängigkeit von statistischen Daten unterschiedlich stark eingefärbt werden. Genau wie bei der Client-Visualisierung können gerenderte Karten auch auf individuellen Webseiten veröffentlicht werden: Durch Einsetzen weniger Zeilen JavaScript können Karten angezeigt werden, die live mit Tabellendaten verlinkt sind.

2.3 Entwicklungsumgebung und API

Google bietet für das Arbeiten mit Google Fusion Tables einerseits eine *Web App* als browsergesteuerte Anwendung² und andererseits den Zugriff über eine API³. EntwicklerInnen erhalten so die Möglichkeit, die bereitgestellte Technologie in eigenen Projekten einzusetzen und um eigene Funktionalitäten zu erweitern. Der Zugriff auf Fusion Tables verläuft dabei nahezu wie bei einer Datenbank: Die API unterstützt die Abfrage und Manipulation von Daten mittels gängiger SQL-Statements. Auch das Anlegen neuer Tabellen über die API ist möglich. Der Zugriff auf das Framework wird mithilfe standardisierter Authentifizierungsmechanismen für Google-Services geregelt. Die folgenden Abschnitte geben einen kurzen Einblick in den Aufbau und die Funktionsweise von Web App und Fusion Tables API.

2.3.1 Google Fusion Tables Web App

Innerhalb der Fusion Tables Web App, die in die Webservice-Suite *Google Drive* integriert ist, können neue Tabellen direkt im Browser erstellt, bearbeitet und befüllt werden. *Google Drive* ist eine von Google angebotener Webdienst, der eine Art Online-Office-Anwendung mit verteiltem Dateisystem für die Synchronisation von Dateien zwischen verschiedenen BenutzerInnen verbindet. Neben Fusion Tables gibt es noch weitere Anwendungsbereiche wie etwa Tabellenkalkulation, Textverarbeitung, Formularerstellung, Bildschirmpräsentation oder Zeichnungen.

Um die Arbeit mit Fusion Tables zu beginnen, besteht neben dem Anlegen leerer Fusion Tables die Möglichkeit, zunächst einfache Google Spreadsheets zu erzeugen oder Excel-Tabellen (xls, csv,

2 URL: <http://www.google.com/drive/start/apps.html#fusiontables> [16.12.2012]

3 URL: <https://developers.google.com/fusiontables/docs/v1/reference/> [16.12.2012]

etc.) bzw. sogar Geokoordinaten im kml-Format zu importieren. Auf den ersten Blick erinnert eine leere Google Fusion Table an ein „abgespecktes“ Tabellenkalkulationsdokument. Auf den zweiten Blick fällt jedoch auf, dass die Tabellenköpfe selbst keine Überschriften darstellen, sondern eigene Dateitypen für die Werte in der jeweiligen Spalte repräsentieren. Neben dem Namen gibt es somit die Möglichkeit aus vordefinierten Dateitypen zu wählen. Zur Auswahl stehen dabei *Date/Time*, *Text*, *Number* und *Location*. Die Auszeichnung als ein spezieller Typ hilft Google dabei, die Daten entsprechend zu interpretieren und weiterzuverarbeiten.



Abbildung 1: Nutzungswege von Google Fusion Tables

Im Typ *Location* liegt auch eine der Stärken von Fusion Tables: Bei Spalten vom Typ *Location*, ist es notwendig, dass die in den Zeilen hinterlegten Werte eine Basis für die Generierung von Geokoordinaten bieten. Das können neben konkreten Geokoordinaten wie Longitude und Latitude auch vollständige Adressdaten wie eine Anschrift (inklusive Postleitzahl, Ort und Land) sein. Außerdem ist es möglich, vollständige Polygon-Netze bzw. Umrisse im kml-Format zu hinterlegen (z.B. Bezirksgrenzen). Im Reiter *Map*, welche per Default bei jeder Fusion Table vorhanden ist, interpretiert Google anschließend die als *Location* hinterlegten Daten und stellt sie direkt auf der Karte dar. Des Weiteren können ausgewählte Tabellendaten wie in bekannten Tabellenkalkulationsprogrammen über Funktionen wie „min“, „max“ und „average“ kumuliert und „auf Knopfdruck“ als Chart visualisiert werden. Mit Hilfe von Google Chart lassen sich verschiedene Arten von interaktiven Diagrammen auf Basis einer Fusion Table visualisieren. Neben den genannten Funktionen bietet Fusion Tables natürlich auch die Möglichkeit, Tabellen selbst zu teilen oder zu publizieren.

Berliner Weihnachtsmärkte 2012: SimpleSearch CSV-Export (20130801)									
Imported at Tue Jan 08 15:59:21 PST 2013 from Berliner_Weihnachtsmaerkte_2012_SimpleSearch_A... more >									
Add Attribution - Edited on January 9, 2013									
File	Edit	Tools	Help	Rows 1	Cards 1	Map of bezirk	+	Share	Manuel Wisniewski
Filter	No filters applied								
1-81 of 81									
id	bezirk	name	strasse	plz_ort	von	bis	veranstalter	oeffnungszeiten	email
1	Charlottenburg-Wilmersdorf	Weihnachtsmarkt vor dem Schloss Charlottenburg	Spandauer Damm	14059 Berlin	11/26/2012	12/26/2012	werbeteam berlin/Tommy Erbe Tel.: 261 20 12, Fax...	Mo-Do 14:00-22:00 Fr-So 12:00-22:00	info@werbeteam-berlin.de
3	Charlottenburg-Wilmersdorf	29. Weihnachtsmarkt an der Gedächtniskirche	Breitscheidplatz	10789 Berlin	11/26/2012	1/1/2013	Arbeitsgemeinschaft e.V., Tel.: 262 95 92, Fa...	So-Do 11:00-21:00 Fr-Sa 11:00-22:00	info@svbev.de
5	Charlottenburg-Wilmersdorf	Weihnachtsmarkt in der Fußgängerzone Wilmersdorfer...	Wilmersdorfer Straße	10627 Berlin	11/28/2012	12/29/2012	Wilmersdorfer Arcaden Centermanagement Tel.: 31 8...	Mo-So 12:00-20:00 24.12. 10:00-14:00	mfi.wilmersdorf@mfi.eu
7	Charlottenburg-Wilmersdorf	Weihnachtsmarkt des Johannischen Sozialwerks	St.-Michaels-Heim, Bismarckallee 23	14193 Berlin	11/30/2012	12/2/2012	Johannisches Sozialwerk e.V. Tel.: 89 68 81 88, F...	Fr 16:00-20:00 Sa 14:00-20:00	
9	Charlottenburg-Wilmersdorf	Schwedischer Weihnachtsbasar	Landhausstr. 26	10717 Berlin	12/1/2012	12/2/2012	Schwedische Victoriatagemeinde Tel.: 864 95 90	Sa 10:00-19:00 So 12:00-18:00	
11	Charlottenburg-Wilmersdorf	Dänischer Weihnachtsmarkt	Briener Str. 12	10713 Berlin	12/1/2012	12/2/2012	Dänische Gemeinde Berlin, Tel.: 873 44 30	Sa 12:00-18:00 So 12:00-17:00	
13	Charlottenburg-Wilmersdorf	Weihnachtsmarkt rund um die Grunewaldkirche	Bismarckallee 28 B	14193 Berlin	12/2/2012	12/2/2012	Frau Wecke, Tel.: 89 73 33 48	11:00-19:00	

Abbildung 2: Google Fusion Tables Web App (New Look)

2.3.2 Google Fusion Tables API 1.0

Um eine breite Masse an AnwenderInnen zur Arbeit mit Fusion Tables zu animieren, bietet Google eine eigenständige Fusion Tables API in der Version 1.0. In Form eines umfangreichen Webservices gewährt die API EntwicklerInnen auf Basis des Representational State Transfer-Verfahrens (kurz: REST) Zugriff auf die Fusion Tables-Funktionalität.

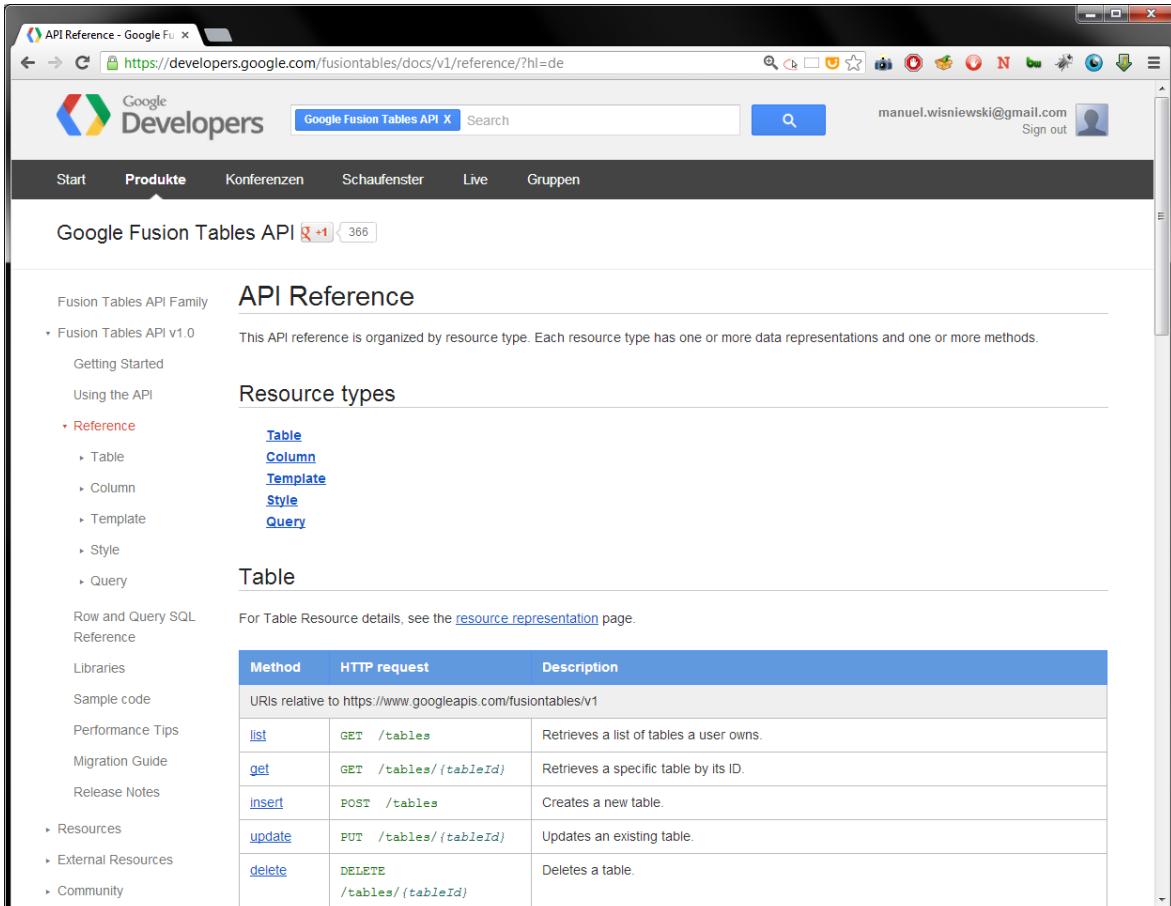
REST ist ein beliebtes Architekturparadigma zur Realisierung von Webservices: Im Zentrum des Modellansatzes steht die Atomarisierung von Anwendungen und Präsentationen in einzelne Ressourcen und die Adressierung einer jeden Ressource mithilfe eines einheitlichen Bezeichners (URI bzw. URL) über das jeweils eingesetzte Netzwerkprotokoll (HTTP, FTP, SMTP, etc.). Im Sinne optimaler Skalierbarkeit erlaubt es die Minimierung von Latenz und Kommunikationsaufwand im Netzwerk und gewährleistet gleichzeitig die Unabhängigkeit aller beteiligten Komponenten. Mittels URLs können so Dokumente in komplexen Netzwerken wie dem WWW eindeutig beschrieben und adressiert werden. Zentrale Funktionsmerkmale von REST sind dabei das Zwischenspeichern (Caching) und Wiederverwenden von Interaktionen sowie das auf schnelle Ersetzbarkeit fokussierte Design der Komponenten.

Mit Hilfe der durch die Google Fusion Tables API zur Verfügung gestellten URLs können über REST

folgende Funktionen abgedeckt werden:

- Erstellen und Löschen von Tabellen
- Lesen und Bearbeiten der Metadaten (Tabellen- und Spaltennamen sowie Spaltentypen)
- Hinzufügen, Aktualisieren und Löschen von Zeilen innerhalb der Tabelle
- Erstellen, Aktualisieren und Löschen von Konfigurationen für die Visualisierung von Daten
- gezielte Abfragen auf die Zeilen innerhalb einer Tabelle

Tabellenstrukturen, Metadaten und Visualisierungseinstellungen können über REST im JSON-Format abgefragt werden. Zeilendaten der Datentabellen werden über SQL-Statements abgefragt, die als HTTP-Requests versandt werden. Die Bereitstellung von Ergebnissen der Datenabfrage erfolgt dann je nach Bedarf im CSV- oder im JSON-Format. Für die Verwendung der API ist grundsätzlich ein Google Account notwendig, da für den REST-Aufruf ein spezieller API-Key zur Autorisierung des API-Zugriffs notwendig ist.



The screenshot shows a web browser window displaying the Google Fusion Tables API Reference. The URL in the address bar is <https://developers.google.com/fusiontables/docs/v1/reference/?hl=de>. The page is part of the Google Developers website, with the navigation bar showing 'Start', 'Produkte', 'Konferenzen', 'Schaufenster', 'Live', and 'Gruppen'. The user is signed in with the email manuel.wisniewski@gmail.com. The main content area is titled 'API Reference' under 'Fusion Tables API Family'. It includes sections for 'Resource types' (Table, Column, Template, Style, Query) and 'Table' (with a table of methods). The 'Table' section lists the following methods:

Method	HTTP request	Description
list	<code>GET /tables</code>	Retrieves a list of tables a user owns.
get	<code>GET /tables/{tableId}</code>	Retrieves a specific table by its ID.
insert	<code>POST /tables</code>	Creates a new table.
update	<code>PUT /tables/{tableId}</code>	Updates an existing table.
delete	<code>DELETE /tables/{tableId}</code>	Deletes a table.

Abbildung 3: Google Fusion Tables API Reference

Zu den zentralen Ressourcen des Fusion Tables-Frameworks gehören neben Tabellen samt Spalten und Zeilen auch Layout-Templates und Style-Anweisungen für die Darstellung in Google Maps. Die Fusion Tables API bietet umfassende Funktionalität für die Arbeit mit diesen Ressourcen.

1.) Arbeiten mit Tabellen:

Zentrales Speicherschema für Daten in Fusion Tables sind Tabellen (oder *tables*), die sich wie gewohnt aus Zeilen und Spalten zusammensetzen. Die Tabellen-Ressource spezifiziert dabei die Tabellenspalten sowie die mit der Tabelle assoziierten Attribute. Über die API können die Spalten verwaltet sowie die Anzeige der Daten bei der Kartenvizualisierung angepasst werden. Drei Typen von Tabellen werden dabei unterschieden:

- **base:** Eine *base table* oder Standard-Tabelle wird entweder durch Anlegen einer neuen Struktur und das Befüllen dieser Struktur mittels SQL-Befehlen bzw. durch den Upload eines Flatfiles eingerichtet.
- **view:** Ein View oder eine Sicht ist eine virtuelle Tabelle, die durch die Auswahl einzelner Tabellenspalten bzw. durch die Anwendung von Filtern erstellt wird. Auch wenn für Views eigene Metadaten hinterlegt werden, verweisen sie lediglich auf die verlinkten Datenstrukturen.
- **merged:** Durch die Kombination mindestens zweier Tabellen über gemeinsame Schlüsselwerte entstehen merged tables, also zusammengesetzte Tabellen. Merged tables erben alle Metadaten von ihren base tables. Das Einfügen neuer Zeilen in merged tables ist hingegen nicht möglich, da sie wieder auf die verlinkten Datenstrukturen verweisen.

Jede Tabelle wird systemseitig mit einer verschlüsselten *Table ID* vom Format 1rgZDIBtz-hMZgBqzVwRMOvjs3RbeK2HE74bY9ivY versehen, die beim Bearbeiten von Spalten, Templates und Styles anzugeben ist.

2.) Arbeiten mit Spalten:

Mithilfe der API können die Spalten einer Fusion Table modifiziert werden. Jede Spalten-Ressource besitzt eine ID, einen Namen und einen zugewiesenen Datentyp (*String, Number, Datetime* oder *Location*).

3.) Arbeiten mit Zeilen:

Die Fusion Tables API erlaubt SQL-Statements für die Abfrage von Zeilendaten. SQL-Statements werden mittels HTTP GET-Request (bei Abfragen) bzw. POST-Request (beim Einfügen, Update und Löschen) vom Client an den Google Fusion Tables Server gesendet. Mithilfe der SQL-Statements können komplexe Abfragen konstruiert und so Zeilendaten gezielt ausgelesen werden. Die Response des Servers liefert jeweils auch die eindeutige *ROWID* zurück, über die dann z.B. Updates entsprechend vorgenommen werden können.

4.) Arbeiten mit Templates:

Das Info-Window-Template kann ebenfalls über die API angepasst werden. Ein Info-Fenster wird angezeigt, sobald ein/e AnwenderIn auf einen Markerpunkt in der Kartendarstellung klickt, und ähnelt grundsätzlich einer Sprechblase. Templates geben an, welche Spalten im Info-Fenster angezeigt werden und können zudem um freien HTML-Code ergänzt werden. Änderungen am Default-Template über die API wirken sich auch auf die Web App aus. Eingegebettete Karten können individuelle Templates anlegen.

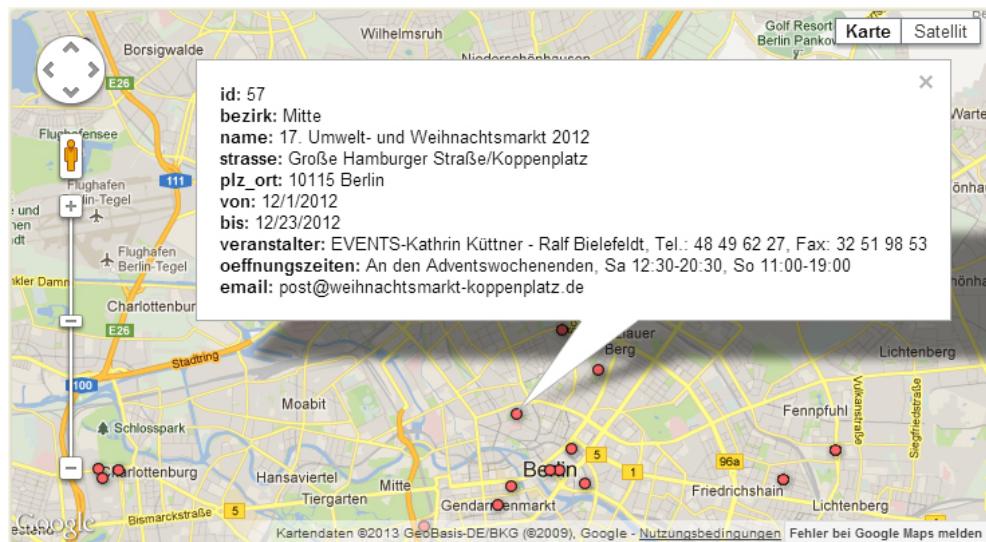


Abbildung 4: Google Maps Info-Window

5.) Arbeiten mit Styles:

Fusion Tables bietet verschiedenen Möglichkeiten für die individuelle Gestaltung von Karten-Features. Zu den Karten-Features gehören in erster Linie Punkte, Linien und Polygone. Die Darstellung dieser Features kann entweder über die Web App oder programmatisch über die API konfiguriert werden. Dazu werden spezielle Styles in Fusion Tables angelegt. Zu diesem Zweck steht in der Google Maps API u.a. auch eigens ein Fusion Tables Layer-Objekt zur Verfügung.

Die Abfrage und Modifikation von Fusion Tables-Ressourcen erfolgt im Rahmen des REST-Ansatzes über HTTP-Requests. Für die Ressourcen Tabelle, Spalte, Template und Style stehen zur Kommunikation mit dem Fusion Tables-Server folgende HTTP-Requests zur Verfügung:

Operation:	Beschreibung:	HTTP Mapping:
list	listet alle Ressourcen eines Typs	GET an die URI einer Ressourcenliste
get	liefert eine konkrete Ressource zurück	GET an die URI einer Ressource
insert	fügt eine neue Ressource ein (legt einen neuen Ressource an)	POST an die URI einer Ressourcenliste, wobei Daten für einen neuen Ressource mitgegeben werden
update	Update einer konkreten Ressource	PUT an die URI einer Ressource, wobei Daten für die upgedatete Ressource mitgegeben werden
delete	löscht eine konkrete Ressource	DELETE an die URI einer Ressource

Tabelle 1: HTTP-Requests für die Ressourcen Tabelle, Spalte, Template und Style

Zum Auslesen der Zeilendaten mittels SQL werden SQL-Statements ebenfalls per HTTP-Request übermittelt. SQL-Statements können als Parameter an die folgende URL adressiert werden: <https://www.googleapis.com/fusiontables/v1/query?sql=>. Folgende SQL-Abfrage sind möglich:

Operation:	Beschreibung:	Query-Format des "sql"-Parameters:
list	listet alle Zeilen einer Tabelle	GET mit konkreter Table ID: SELECT ROWID FROM <table_id>
get	Liefert eine konkrete Zeile	GET mit konkreter Table ID und Suchkriterium: SELECT ROWID FROM <table_id> WHERE <your filter>
insert	fügt eine neue Tabellenzeile ein	POST mit konkreter ROWID, wobei Daten für einen neuen Zeile mitgegeben werden: INSERT INTO <table_id> (<column_name> , <column_name>*) VALUES (<value> , <value>*)
update	Update einer Tabellenzeile	POST mit konkreter ROWID, wobei Daten für die geänderte Zeile mitgegeben werden: UPDATE <table_id> SET <column_name> = <value> , <column_name> = <value> * WHERE ROWID = <row_id>
delete	löscht eine Tabellenzeile	POST mit konkreter ROWID: DELETE FROM <table_id> {WHERE RO- WID = <row_id>}

Tabelle 2: SQL-Queries für HTTP-Requests zum Auslesen von Zeilendaten

Mit Hilfe der zur Verfügung gestellten Methoden und gängigen Formate ist die API universell und programmiersprachenunabhängig einsetzbar. Methoden für das Absetzen eines HTTP-Requests sowie den Datenaustausch mit JSON sind mittlerweile Standard in allen Programmiersprachen im Bereich der Webentwicklung. Nach ersten Recherchen findet Fusion Tables vor allem Anwendung in browsergesteuerten Anwendungen und weniger in komplexen Standalone-Anwendungen.

3 ANWENDUNGSSPEZIFIKATION

Die Anwendungsspezifikation beinhaltet in groben Zügen alle Basisanforderungen an die zu entwickelnde Beispielapplikation auf Grundlage der Google Fusion Tables-Technologie. Als Beispielapplikation wird ein Datencockpit für die Berliner Weihnachtsmärkte im Jahr 2012 umgesetzt. Kernanforderung an das Cockpit ist eine elaborierte Such- bzw. Filterfunktionalität. Zur Konkretisierung des Entwicklungsvorhabens und zur Strukturierung der Benutzeroberfläche für die anvisierten Darstellungs- und Auswahlfunktionen wird ein erster Wireframe erstellt. Abschließend werden die technischen Rahmenbedingungen für die Entwicklung mit Google Fusion Tables abgesteckt. Daraus ergeben sich in weiterer Folge die Arbeitsschritte und -pakete für die technische Umsetzung (Kapitel 4).

3.1 Motivation

Sobald Weihnachten vor der Tür steht, beginnt auch die Weihnachtsmarktsaison, spätestens Ende November schießen sie wie Pilze aus dem Boden. Es gibt Weihnachtsmärkte in verschiedensten Kategorien, Größen und Ausprägungen: für Kinder, für Jugendliche oder für Erwachsene, für Vollblutenthusiasten mit Heißhunger auf Lebkuchen, für Glühweintassensammler oder aber für jene, die nur am Wochenende gemütlich etwas durch den Schnee schlendern wollen. Besucherinformationen zu einzelnen Märkten sind – sofern überhaupt erhoben – oftmals nicht zentral einsehbar, sondern müssen mühselig über die Tagespresse oder unübersichtlich im Internet verstreute Web-sites zusammengesucht werden. Vorrangig interessant für Besucher sind grundlegende Informationen zu Öffnungszeiten, Anfahrt und ggf. Eintrittspreisen. Oft sind dies auch die einzigen Informationen, die sich bei der Recherche überhaupt ermitteln lassen. Das Angebot ist dabei sehr unübersichtlich und intransparent, so dass Weihnachtsmarktbesuche in der Regel eher spontan als geplant stattfinden.

Das zu entwickelnde Datencockpit nimmt sich dieses Missstandes an und soll als zentrale Anlaufstelle für die Suche nach Weihnachtsmärkten dienen. Dabei soll neben der Bereitstellung der o.g. Besucherinformationen insbesondere die Darstellung der geografische Lage der Märkte auf einer Karte die Orientierung für Suchende vereinfachen. Im Rahmen der Lehrveranstaltung beschränkt sich der Prototyp auf die Berliner Weihnachtsmärkte der Saison 2012, grundsätzlich erlaubt die Funktionalität aber auch eine Ausweitung auf alle Weihnachtsmärkte in Deutschland oder sogar darüber hinaus. Allein in Berlin gibt es 2012 ca. 100 Weihnachtsmärkte, dabei werden für die Analyse und Auswertung im Datencockpit auch alle sogenannte Advents-, Winter-, Kiez- und Weihnachtsflohmarkte berücksichtigt.

3.2 Suchfunktionalität

Zur effizienten Suche nach Weihnachtsmärkten sollen den Suchenden verschiedene Segmentierungsmerkmale zur Verfügung stehen. Neben klassischen geografischen Werten gehören dazu auch die Unterscheidung der Märkte anhand eines spezifischen Kriterienkataloges. Die Weihnachtsmarktsuche ist dabei keine herkömmliche Volltextsuche, sondern letztlich ein spezieller Filter, der die Ergebnisse aus den Quelldaten herausfiltert. Nach folgenden Kriterien soll sich die Weihnachtsmarktliste im Prototyp filtern lassen:

- Bezirk
- Öffnungszeit (verfügbare Weihnachtsmärkte an einem bestimmten Datum)
- Eintritt (frei oder nicht)
- Fahrgeschäfte vorhanden
- Marktsortiment bzw. -kategorie

Die Datenbasis ist hierbei nicht in einer herkömmlichen Datenbank persistiert, sondern in Google Fusion Tables. Eine Stärke von Fusion Tables ist die Kombination verschiedener Tabellen durch die Funktion „merge“. Auf diese Weise kann die eigentliche Datenbasis aus verschiedenen Quellen bestehen. Die „gemergte“ Tabelle ist hierbei nur eine Zusammenführung und bildet stets den aktuellen Stand der integrierten Fusion Tables ab. Das hat Vorteile für die Pflege der Inhalte, die durch das Kapseln von Informationen in kleineren Tabellen erleichtert wird.

Das Datencockpit, das als Micro-Website realisiert werden soll, hat die Anwendung der genannten Filtermöglichkeiten zu organisieren. Nach der Auswahl eines oder mehrerer Kriterien soll die Ausgabe der gefilterten Ergebnisse erfolgen. Die einzelnen Kriterien werden dabei mit einem logischen UND verknüpft. Umso mehr Kriterien gleichzeitig ausgewählt werden, desto weniger Ergebnisse stehen letztlich zur Verfügung. Für optimale Benutzerfreundlichkeit werden bei der Selektion einzelner Kriterien die Filterregeln direkt zur Laufzeit angewandt und die Auswahlmöglichkeiten weiterer Kriterien in Abhängigkeit angepasst, so dass nur mögliche Filterkombinationen für die Filterung verwendet werden können. Beispielsweise zeigt die Filtermöglichkeit nach Sortiment nur die für einen vorab gewählten Bezirk zur Verfügung stehenden Sortimente an.

Die Ausgabe der gefilterten Ergebnisse erfolgt statt klassischer Listendarstellung auf Basis einer interaktiven Karte mit Hilfe von Google Maps. Dabei werden die Weihnachtsmärkte auf der Landkarte durch Markerpunkte angezeigt. Für die Angabe der hierfür benötigten „Location“-Daten ist die Verwendung der Klartextadresse gegenüber der Referenzierung mittels der Geokoordinaten Longitude und Latitude zu favorisieren. Google Fusion Tables bietet eine automatische Geokodierung für Textadressen, wobei spezielle Formatvorgaben eingehalten werden müssen. Voraussetzungen für die Umwandlung von Textadressen in Geodaten werden im Rahmen der technischen Umsetzung (Kapitel 4) noch genauer beschrieben.

Neben der Positionierung gefilterter Weihnachtsmärkte auf der Karte sollen zusätzlich *Heatmaps* in die Karte eingebunden werden. Durch das Hinterlegen von komplexen Polygonennetzen auf Basis von Gekoordinaten im KML-Format können Bezirksgrenzen visualisiert und die Bezirksflächen als separate Layer über der Google Maps-Karte angezeigt werden. Mit Hilfe der visualisierten Bezirke können verschiedene Statistiken durch entsprechend deutliche Farbverläufe visualisiert werden, beispielsweise die Anzahl der Weihnachtsmärkte pro Bezirk.

3.3 Schematischer Entwurf

In einem ersten Entwurf für die Umsetzung des Datencockpits wurde folgender Wireframe erstellt, der die genannten Funktionsmerkmale berücksichtigt:

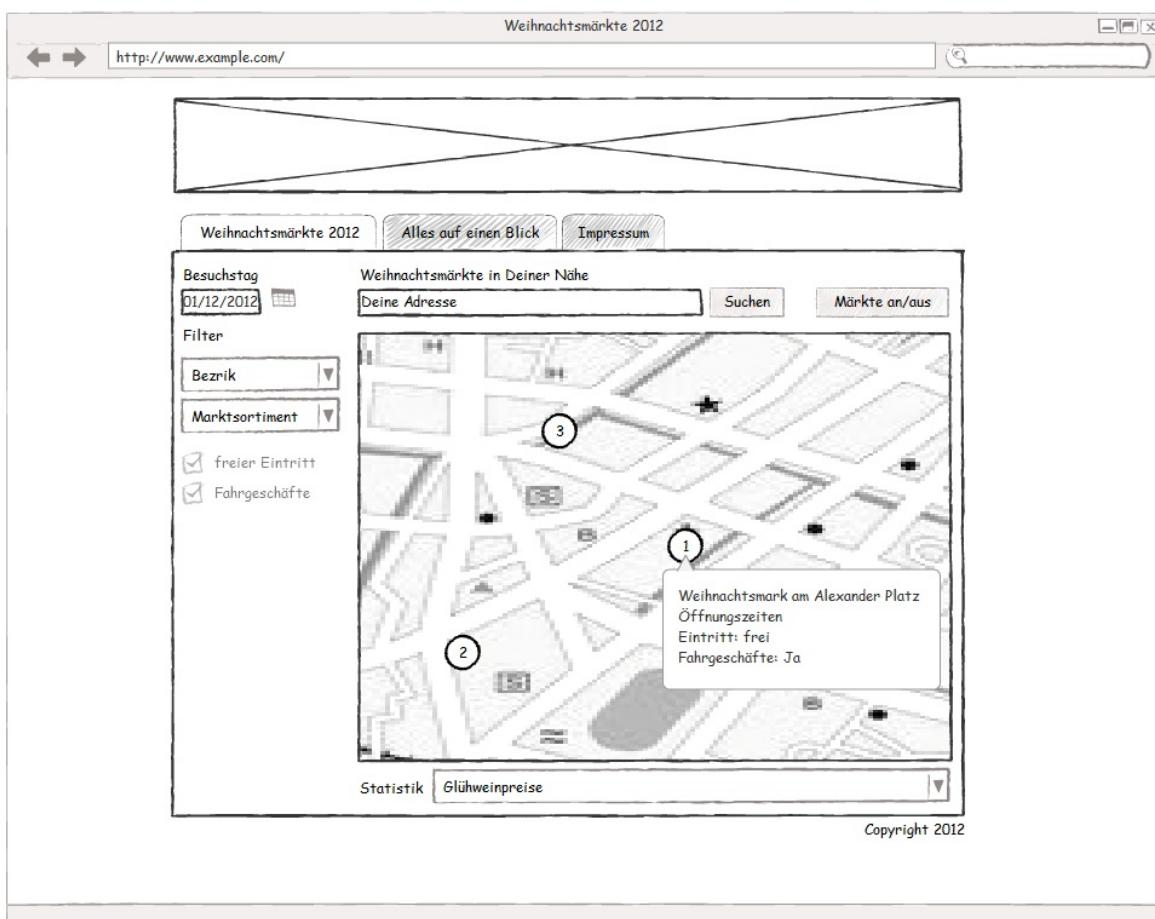


Abbildung 5: Wireframe des Weihnachtsmarkt-Datencockpits

Gemäß der angeführten Filterspezifikation soll das Weihnachtsmarkt-Datencockpit bequem und benutzerfreundlich Informationen zu Berliner Weihnachtsmärkten auf Grundlage des Kartenmaterials von Google Maps aufbereiten und zur Verfügung stellen. Dabei soll AnwenderInnen die Möglichkeit gegeben werden, das Weihnachtsmarktangebot gezielt nach relevanten Kriterien zu filtern und zu durchsuchen. Zentrale Suchkriterien sind dabei die Öffnungszeiten der Märkte, die Lage

(unterschieden nach Bezirk), das Marktsortiment (unterschieden nach Nostalgisch, Handwerk, Kunst, etc.) sowie die Frage nach eventuellen Eintrittskosten und nach der Verfügbarkeit von Fahrgeschäften. Zudem soll die Anwendung UserInnen erlauben, über die Eingabe der eigenen Position alle verfügbaren Weihnachtsmärkte in der Nähe aufgezeigt zu bekommen. Als besonderes Feature können zusätzlich Weihnachtsmarktstatistiken auf dem Kartenmaterial (und ggf. in beigefügten Diagrammen) abgefragt werden, etwa über die anteilmäßige Verteilung der Märkte auf die Berliner Stadtbezirke oder etwa durchschnittliche Glühweinpreise.

3.4 Entwicklungsinfrastruktur

Voraussetzung für die Entwicklung mit Fusion Tables ist in erster Linie ein Google Benutzer-Account sowie ein gültiger API-Schlüssel für Fusion Tables, der über die Google APIs Console⁴ bezogen werden kann. Das Anlegen neuer Fusion Tables erfolgt dann direkt aus Google Drive⁵ heraus. Zugang zu Google Service ist mit allen gängigen Browsern möglich, wobei jedoch für Google Chrome eine spezielle Fusion Tables App als AddOn angeboten wird. Für die Aufbereitung der Daten ist gegebenenfalls ein lokales Tabellenkalkulationsprogramm empfehlenswert. Die Einbindung der Karten/Visualisierungen erfordert Zugang zu einem Webserver und für den Entwurf des Prototyps empfiehlt sich ggf. ein HTML/JavaScript-Editor.

Für die Umsetzung des spezifizierten Prototyps wurde konkret auf folgende Entwicklungsumgebung zurückgegriffen:

- **Browser:** Google Chrome, Firefox, Opera
- **Tabellenkalkulation:** Microsoft Excel 2010, OpenOffice Calc
- **HTML-Editor:** Dreamweaver CS5, Notepad++
- **Webserver:** Apache/XAMPP

Für die Versionsverwaltung und Dokumentation wurden zudem Git eingesetzt und ein Repository auf github⁶ eingerichtet.

4 <http://code.google.com/apis/console/> [05.02.2013]

5 <http://drive.google.com/> [05.02.2013]

6 http://github.com/borste/WiBo_GoogleFusionTables/ [05.02.2013]

4 TECHNISCHE UMSETZUNG

Im Anschluss an die thematische wie technische Einarbeitung und die Spezifikation des Entwicklungsvorhabens gliedert sich die Umsetzung des Weihnachtsmarktcockpits in die folgenden Arbeitsschritte:

1. Datenerhebung bzw. -recherche,
2. Importieren und Aufbereiten der Daten,
3. Konfigurieren der Datenvisualisierungen,
4. Erstellung der Cockpit-Website und
5. Implementierung der interaktiven Anzeigesteuerung.

4.1 Datenerhebung

Die Daten für die Berliner Weihnachtsmärkte 2012 sind abrufbar über das *Berlin Open Data*-Portal⁷ der Senatsverwaltung für Wirtschaft Technologie und Forschung des Landes Berlin:

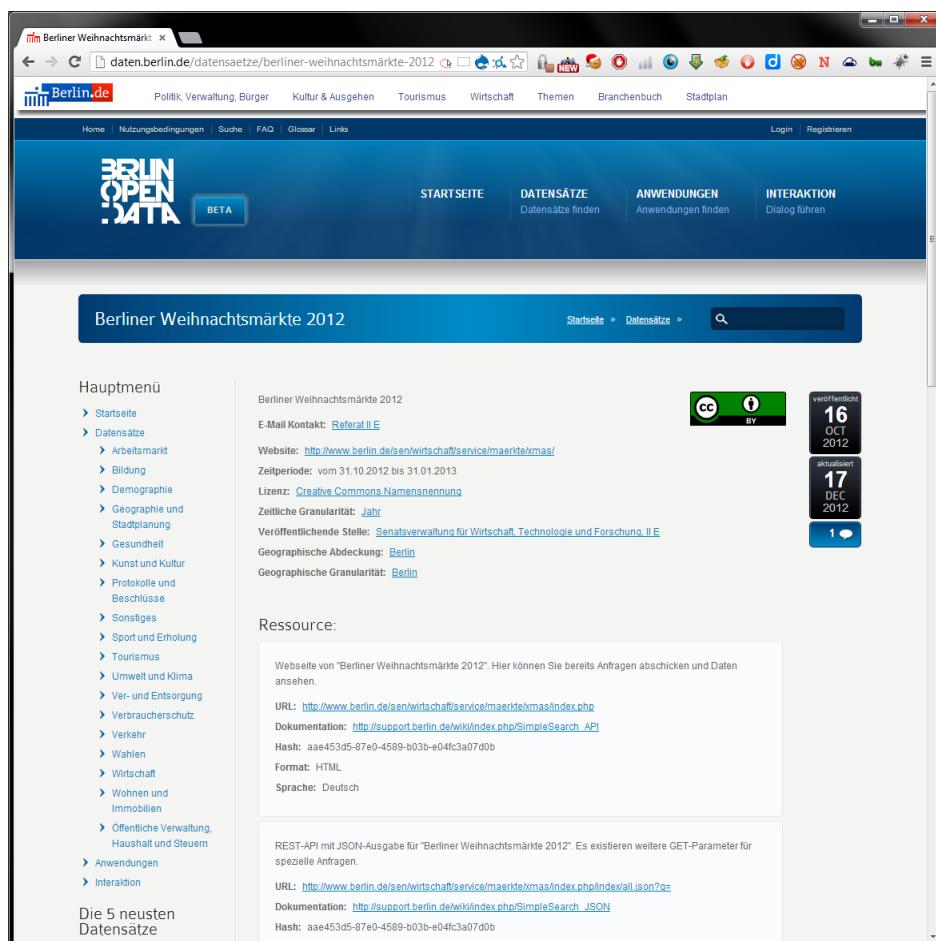


Abbildung 6: Berlin Open Data-Portal der Senatsverwaltung Berlin

⁷ <http://daten.berlin.de/datensaetze/berliner-weihnachtsmaerkte-2012> [13.12.2012]

Das Portal stellt mehrere Möglichkeiten zur Abfrage der Daten in den Formaten HTML, JSON, XML, XLS und CSV bereit. Die bereitgestellten Daten umfassen je Weihnachtsmarkt:

id:	saisonübergreifend eindeutige ID
bezirk:	Veranstaltungsbezirk
name:	Name/Bezeichnung
strasse:	Straße
plz_ort:	PLZ und Ort
von:	Startdatum
bis:	Enddatum
veranstalter:	Name des Veranstalters, ggf. Telefonnummer und/oder Fax
oeffnungszeiten:	Öffnungszeiten (Wochentage und Uhrzeiten)
email:	E-Mail-Adresse des Veranstalters
w3:	Internetadresse des Marktes bzw. des Veranstaltungsortes
bemerkungen:	Bemerkungen (z.B. Heiligabend geschlossen, Fahrgeschäfte, Eintritt)

Tabelle 3: Weihnachtsmarktdetails des Berlin Open Data-Portals

Weitere für die Auswertung benötigten Informationen wie Eintrittskosten, Fahrgeschäfte oder Glühweinpreise mussten individuell erhoben werden. Teilweise sind sie aus der „Bemerkungen“-Spalte ableitbar, ansonsten wurden fehlende Daten – vor allem zu Eintrittsregelungen – über die Website "Weihnachten in Berlin"⁸ beschafft. Für die Glühweinpreise wurden im ersten Entwicklungsdurchlauf lediglich Testdaten verwendet. Dazu wurde mithilfe eines Testdatengenerators⁹ eine Wertetabelle mit Glühweinständen und Preisen zwischen 2 und 5 Euro erzeugt. Die einzelnen Stände wurden dann den jeweiligen Märkten randomisiert zugeordnet. Die Auswertung der Preisdaten erfolgte dann über die Summarize- und Merge-Funktionen in Fusion Tables.

Komplizierter gestaltete sich die Beschaffung der für die Polygondarstellung nötigen Koordinaten für die Berliner Bezirksgrenzen, nachdem die Suche im öffentlichen Tabellenkatalog von Fusion Tables¹⁰ keinen Erfolg brachte. Der Grenzverlauf der Bezirke ist notwendig, um einzelne Bereiche der Karte später als Heatmaps dynamisch anhand der Tabellenwerte einfärben zu können. Die Grenzen mussten also mithilfe des *Google Maps API v3 Tool*¹¹ von Hand eingezeichnet und die Werte in ein KML-File übertragen werden.

8 <http://www.weihnachteninberlin.de/weihnachtsmaerkte/> [13.12.2012]

9 <http://www.generatedata.com/> [13.12.2012]

10 <http://research.google.com/tables> [13.12.2012]

11 <http://www.birdtheme.org/useful/v3tool.html> [13.12.2012]

4.2 Datenimport und -aufbereitung

Nach dem Herunterladen der Daten müssen diese für den Einsatz mit Google Fusion Tables optimiert werden. Dazu gehört u.a. die Verkettung der Adressdaten in ein von Google geocodierbares bzw. von Google Maps als Geodaten interpretierbares Format. Dieses Format ist landesspezifisch und für die Berliner Anschriften gilt folgendes Schema:

[spez. Ortsbezeichnung]¹², [Straßenname] [Hausnummer], [Postleitzahl] Berlin,
Deutschland

Mithilfe eines Tabellenkalkulationsprogramms wurden die Felder `strasse`, `plz_ort` in ein geocodierbares Feld `geo_adresse` zusammengefügt.

Nach Aufbereitung der Daten wurde folgende *base tables* in Fusion Tables importiert:

- Berliner Bezirke Boundaries (Polygons)
- Berliner Weihnachtsmärkte 2012: SimpleSearch CSV-Export (Berlin Open Data)
- Berliner Weihnachtsmärkte 2012: Geo-Adressen
- Berliner Weihnachtsmärkte 2012: Eintrittspreise
- Berliner Weihnachtsmärkte 2012: Glühweinpreise pro Stand

Mithilfe der Summarize-Funktion wurden die Tabellen verdichtet, um spezielle statistische Kennzahlen, wie z.B. Anzahl der Weihnachtsmärkte pro Bezirk, zu ermitteln. Fusion Tables erlaubt leider nicht das Mergen von Summaries, da diese nur als Views innerhalb der einzelnen Datentabellen abgelegt werden. Um durch Summaries erzielte Kennzahlen weiterzuverwenden, muss eine summarized Tabelle über den Befehl „Download...“ aus Fusion Tables exportiert und als neue Fusion Table wieder importiert werden. Dieser Workaround führt zu erheblicher Redundanz im System und macht das Arbeiten mit vielen Tabellen leicht unübersichtlich. Eine genaue Namenskonvention für Tabellen ist daher gerade bei der Arbeit im Team unerlässlich.

Nach mehreren Summaries, Exports, Re-Imports und Merges liefern also letztlich folgende zwei Tabellen die Datenbasis für das Weihnachtsmarktcockpit:

1. Wibo - Weihnachtsmarkt Finder 2012 – Weihnachtsmärkte

[docid=1rgZDIBtzhMZgBqzVwRMOvjs3RbeK2HE74bY9ivY](#)

2. Wibo - Weihnachtsmarkt Finder 2012 - Bezirke Auswertung

[docid=1mYQYgSuc3D8RZeklmSLOB2XcOhIETwXqcXXv1DA](#)

Die „Weihnachtsmärkte“-Tabelle (1.) besteht aus einem großen Merge aller erhobener Weihnachtsmarktinformationen auf Weihnachtsmarkt-Detaillevel:

¹² optional

id:	saisonübergreifend eindeutige ID
bezirk:	Veranstaltungsbezirk
name:	Name/Bezeichnung
strasse:	Strasse
plz_ort:	PLZ und Ort
von:	Startdatum
bis:	Enddatum
veranstalter:	Name des Veranstalters, ggf. Telefonnummer und/oder Fax
oeffnungszeiten:	Öffnungszeiten (Wochentage und Uhrzeiten)
email:	E-Mail-Adresse des Veranstalters
w3:	Internetadresse des Marktes bzw. des Veranstaltungsortes
bemerkungen:	Bemerkungen
geo_adresse:	geokodierbare Notation der Adresse
eintrittspreis:	Eintrittspreis (Standard)
eintrittspreis_ermäßigt:	Eintrittspreis (Ermäßigt)
sortiment:	Marktkategorie
fahrgeschäfte:	Vorhanden=1, nicht vorhanden=0

Tabelle 4: Spalten der Tabelle „Wibo - Weihnachtsmarkt Finder 2012 – Weihnachtsmärkte“

Die „Bezirke Auswertung“-Tabelle liefert hingegen Daten auf Detailebene der Bezirke und liefert die Grundlage für die Darstellung der Heatmaps (siehe Kapitel 4.3.2):

bezirk:	Bezirksname
count:	Anzahl Weihnachtmärkte pro Bezirk
MAXIMUM(eintrittspreis):	höchster Eintrittspreis im Bezirk
AVERAGE(preis):	durchschnittlicher Glühweinpreis
geometry:	Polygondaten für Bezirksgrenze (kml)

Tabelle 5: Spalten der Tabelle: „Wibo - Weihnachtsmarkt Finder 2012 - Bezirke Auswertung“

Die beiden Tabellen liefern alle nötigen Daten für die Auswertung und Visualisierung der Weihnachtsmärkte im Datencockpit auf den Detailebenen Weihnachtsmarkt und Bezirk. Weitere Daten können z.B. durch Mergen mit zusätzlichen Tabellen als neuen Spalten hinzugefügt und weitere Karten und Diagramme mühelos angelegt werden.

4.2.1 Exkurs: Probleme beim Importieren der Daten

Sowohl in Microsoft Excel 2010 als auch in OpenOffice Calc traten beim Einlesen der Berlin Open Data-Daten zur weiteren Nachbearbeitung Fehler auf. So wurde beispielsweise beim Standardmäßigen Öffnen der csv-Dateien in Excel die UTF-8-Kodierung nicht erkannt und somit alle Umlaute verworfen. Der Versuch, die csv-Datei über die Datenimportfunktion mit richtiger Kodierung zu importieren scheiterte daran, dass einige Zelleninhalte Zeilenumbrüche enthalten. Dieser wird standardmäßig von Excel als Zeilenende und damit Ende des Datensatzes interpretiert. Der Einfachheit halber wurde im vorliegenden Fall die csv-Datei klassisch geöffnet und die Umlaute kurzerhand mittels „Suchen und Ersetzen“ ausgebessert.

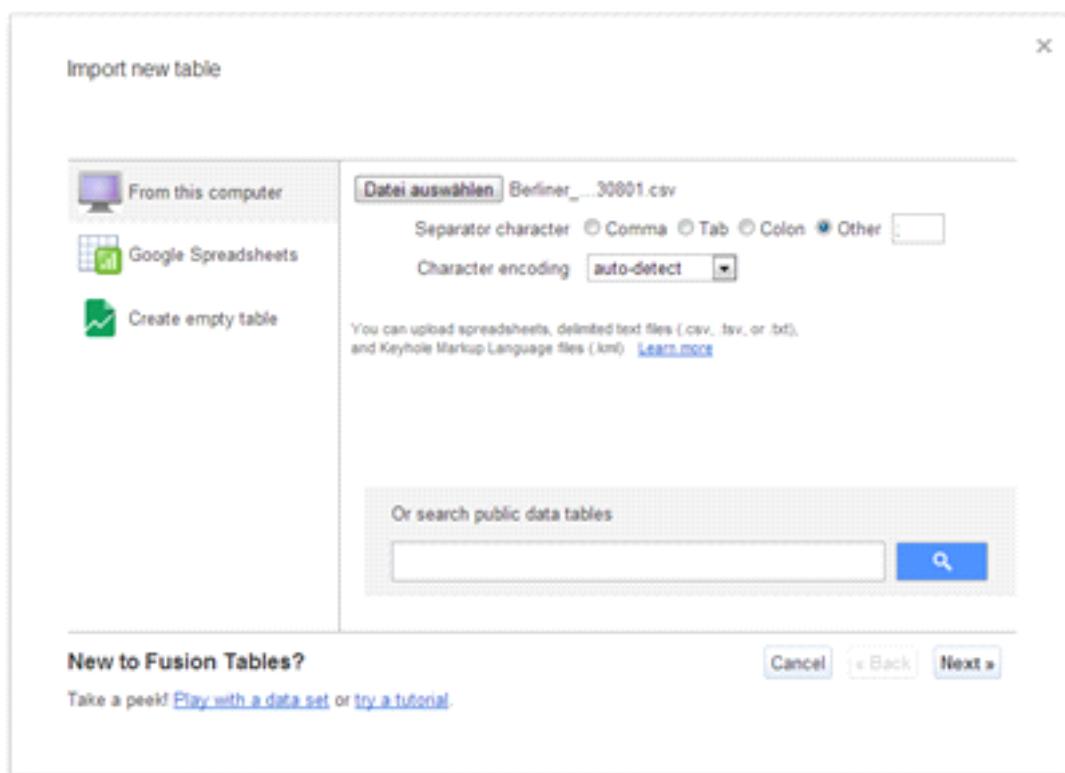


Abbildung 7: Neue Tabelle importieren (1): Importeinstellungen

Beim Import in Fusion Tables ist es dann wichtig, als Trennsymbol das von Excel für csv-Dateien verwendete Semikolon („;“) unter „Other“ einzustellen und das Character Encoding auf „auto-detect“ zu setzen. Excel speichert die Dateien standardmäßig in ANSI-Kodierung ab. Die von der API generierten csv-Files haben eine „ANSI as UTF-8“-Kodierung und werden von Google nicht unterstützt.

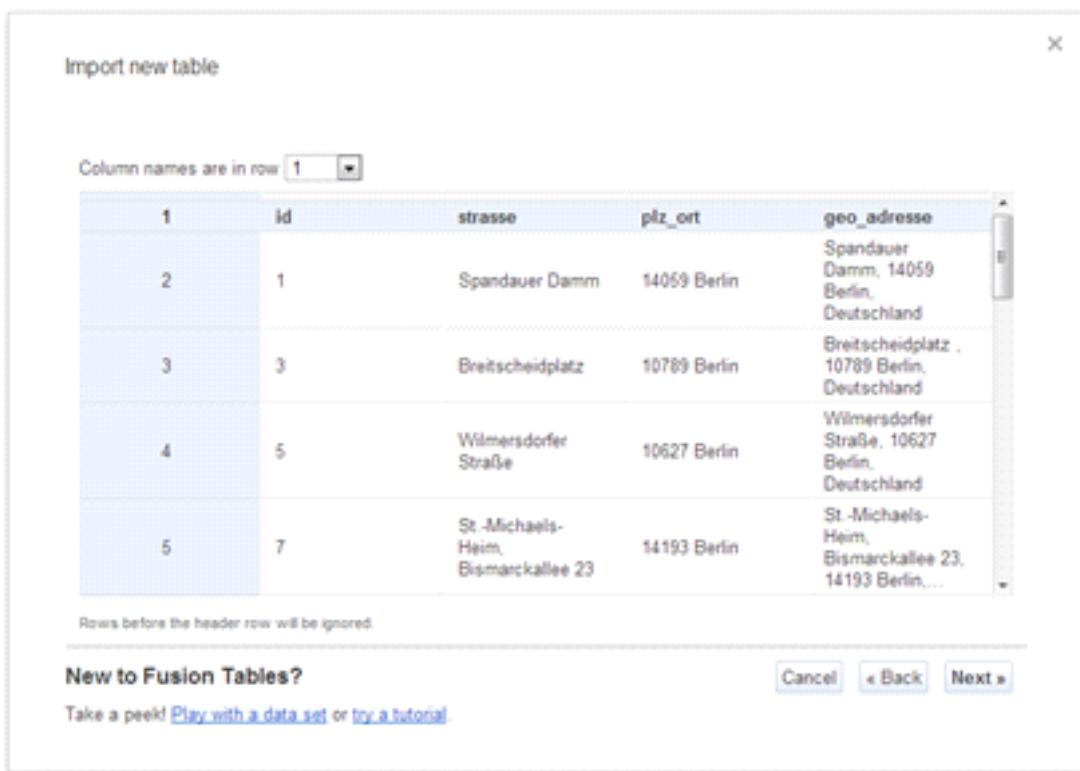


Abbildung 8: Neue Tabelle importieren (2): Erfolgreicher Import

Beim Import- und Exportieren von Daten zwischen Google Fusion Tables und lokalen Tabellenkalkulationsprogrammen ist also gegenwärtig noch Vorsicht geboten und erhöhte Aufmerksamkeit vonnöten. Auch bei der Anzahl der unterstützten Datentypen und Zellformatierungen ist Fusion Tables noch tief in der Beta-Phase verankert und lässt viele gängige Formate vermissen. So wird beispielsweise bislang als einzige Währungsnotation für Zahlenwerte die Dollarformierung samt Dezimalpunkt angeboten. Hier ist für folgende Releases Nachbesserung geboten.

4.3 Konfiguration der Visualisierungen

Für die Visualisierung der Daten stehen wie bereits erwähnt zwei zentrale Konzepte zur Verfügung: Diagramme und geografische Abbildungen in Google Maps. Die Fusion Tables Web App erlaubt das handliche Konfigurieren beider Darstellungsformen. Über die API können die Grafiken bzw. Karten dann später in das Datencockpit eingebunden werden.

4.3.1 Diagramme mit Fusion Tables

Zum Anlegen einer neuen Visualisierung in Form eines Diagramms kann in der Fusion Tables Web App rechts neben dem „Rows“-Reiter aus der Dropdown-Liste die Option „Add chart“ ausgewählt werden. Dadurch wird eine neue Karteikarte mit einem Diagramm angelegt. Zur Auswahl stehen nun links verschiedene Diagrammformen: Punktdiagramm (*Scatter chart*), zoombares Liniendiagramm

gramm (*Zoomable line chart*), Flächendiagramm (*area chart*), Säulendiagramm (*Column chart*), Balkendiagramm (*Bar chart*), Liniendiagramm (*Line chart*), Kreisdiagramm (*Pie chart*) und Netzwerkgraph (*Network graph*). Daneben sind die für die aktuell gewählte Diagrammkategorie als relevant identifizierten Datenspalten aufgelistet und ggf. weitere Konfigurationsparameter.

Zur Visualisierung der Häufung von Weihnachtsmärkten pro Bezirk wählen wir beispielsweise ein Balkendiagramm (*Bar chart*), als *Category* den Bezirk und als *Value* count, also die durch ein Summary der Weihnachtsmarkt-base table berechnete Anzahl der Weihnachtsmärkte pro Bezirk. Dann setzen wir die Anzahl angezeigter Kategorien (*Maximum categories*) von 10 auf 12, da wir alle zwölf Berliner Bezirke anzeigen wollen, und sortieren die Liste (*Sort by*) nach Anzahl (*count*). Das Ergebnis ist folgendes Balkendiagramm:

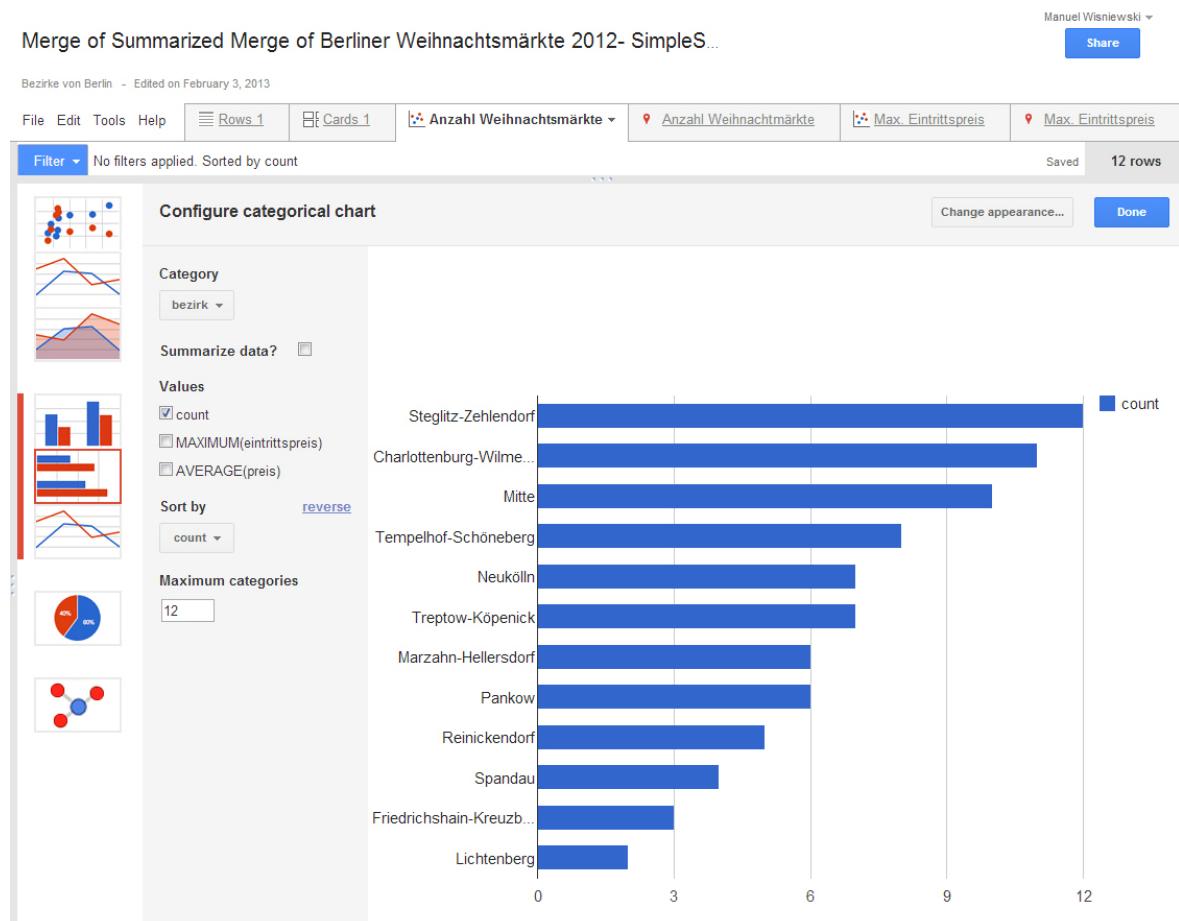


Abbildung 9: Balkendiagramm: Anzahl Weihnachtmärkte pro Bezirk

Auch wenn diese Visualisierungsform im derzeitigen Entwurf keine Anwendung findet, dient sie der Illustration der Möglichkeiten von Fusion Tables. Alle Style-Einstellungen für diese Diagrammkarte werden automatisch gespeichert und lassen sich z.B. via URL oder iFrame einbetten bzw. über die API direkt referenzieren.

4.3.2 Karten und Heatmaps mit Fusion Tables

Um eine Heatmap zu erzeugen, wählen wir hingegen den Karteireiter (*Map*). Alternativ kann über die Option „Add map“ eine neue Karte zur Tabelle hinzugefügt werden. Fusion Tables identifiziert automatisch jene Tabellenwerte, die Geodaten repräsentieren, und stellt sie auf der Karte da. Für die Weihnachtsmarktliste liest Fusion Tables beispielsweise die Adressdaten der einzelnen Märkte und markiert sie automatisch auf der Karte:

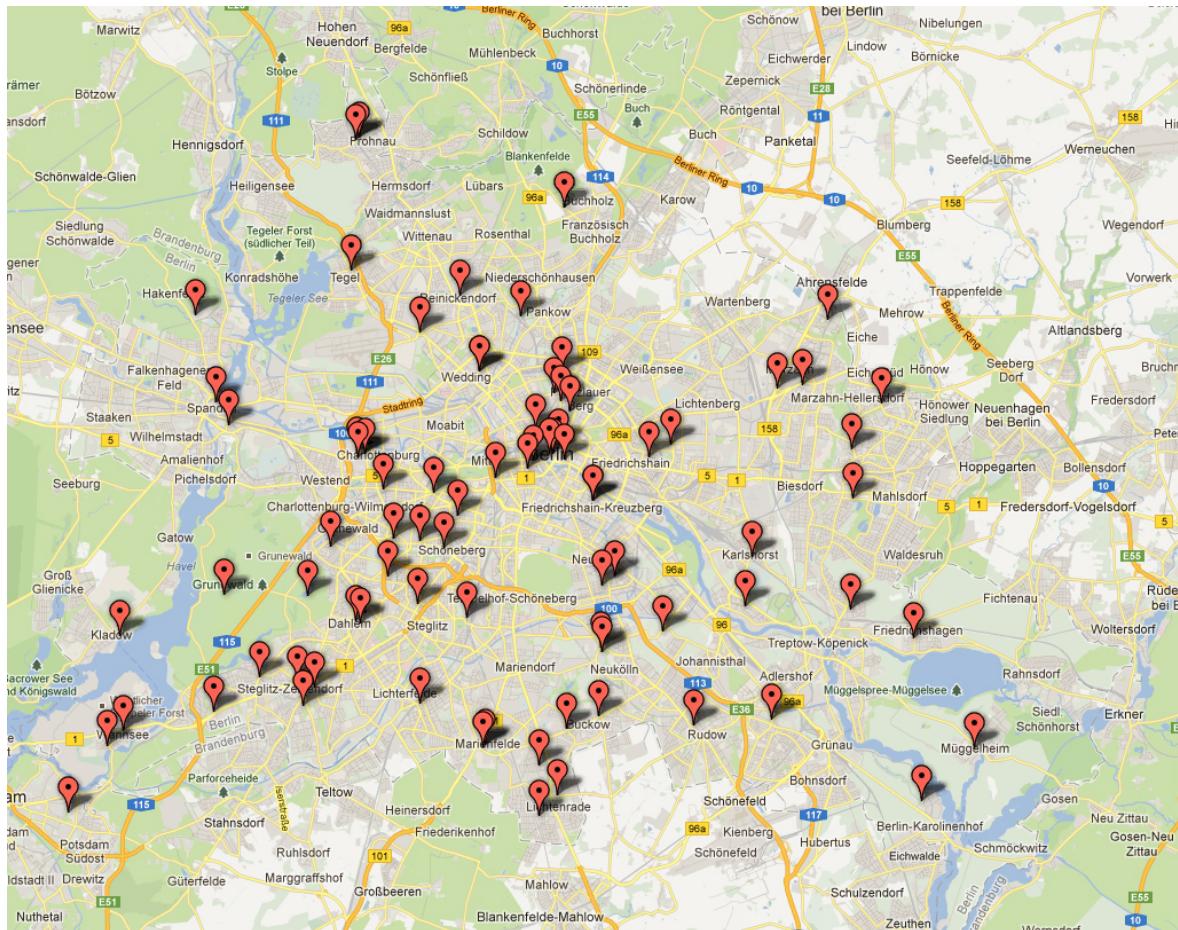


Abbildung 10: Markierung der Adressdaten in Google Maps

Sollten mehr als eine Spalte mit Geodatenwerten verfügbar sein, kann im Dropdown-Menü des Kartenreiters über die Option „Change location“ die entsprechende Tabellenspalte für die Flächedarstellung (hier: geometry) ausgewählt werden. Standardmäßig werden zunächst alle Flächen mit der gleichen vordefinierten Farbfüllung dargestellt.

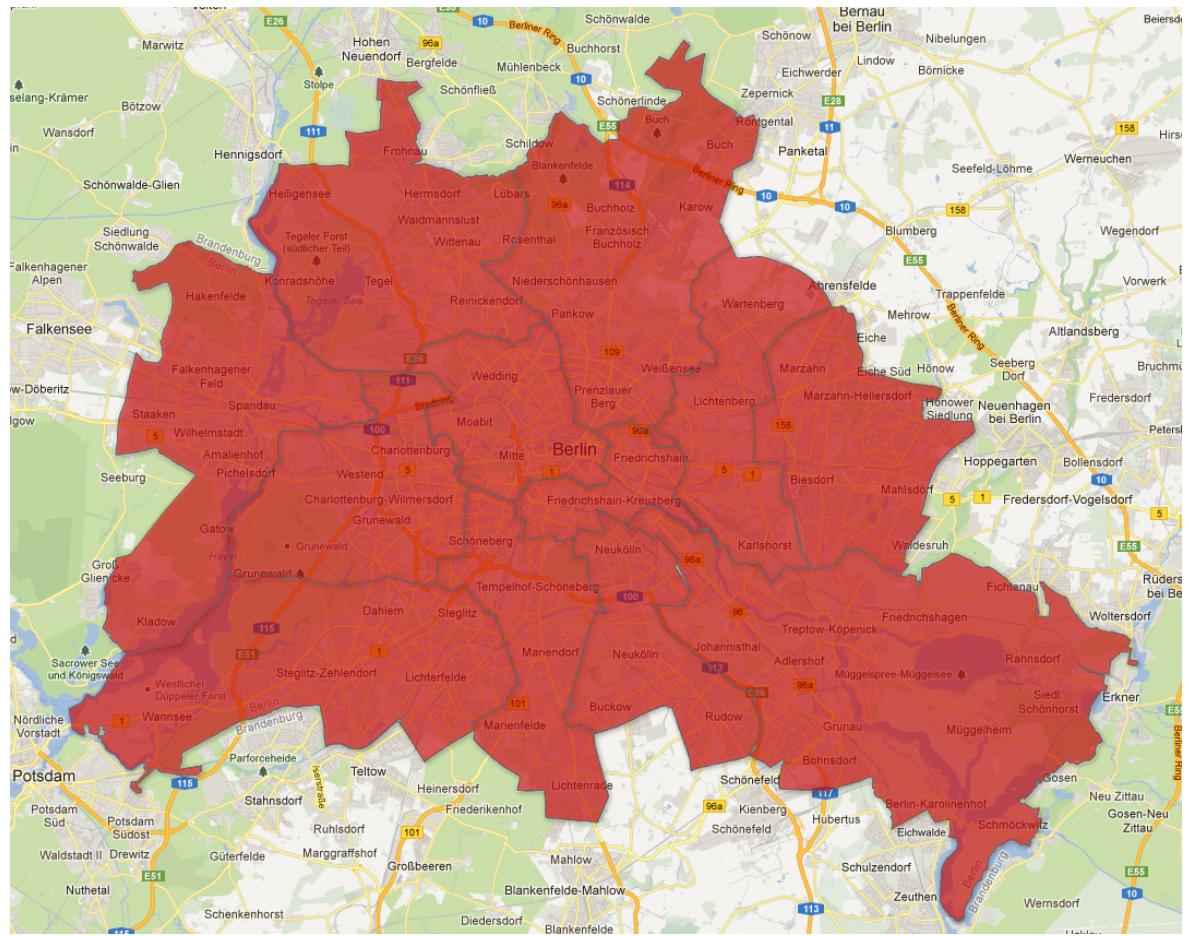


Abbildung 12: Polygon-Flächendarstellung in Google Maps

Um die Färbung der Flächen nun an bestimmten Tabellenwerten auszurichten, wählen wir aus dem Hauptmenü „Tools > Change map styles...“.

Merge of Summarized Merge of Berliner

Bezirke von Berlin – Edited on February 3, 2013

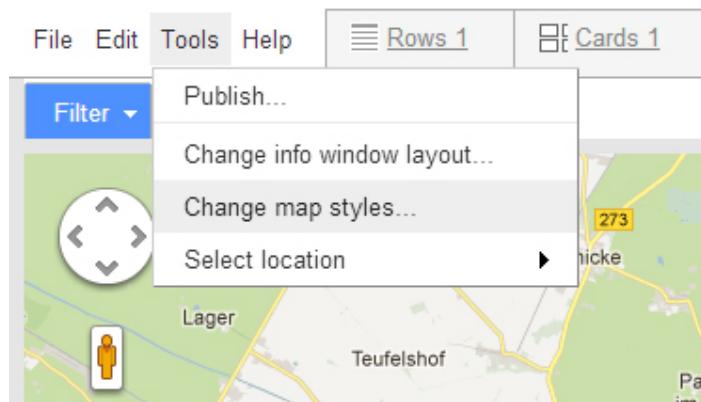


Abbildung 13: Menü: "Tools - Change mal styles..."

Im folgenden Dialogfenster müssen in der Kategorie *Polygons* die Option „Fill color“ und der Reiter „Gradient“ ausgewählt werden:

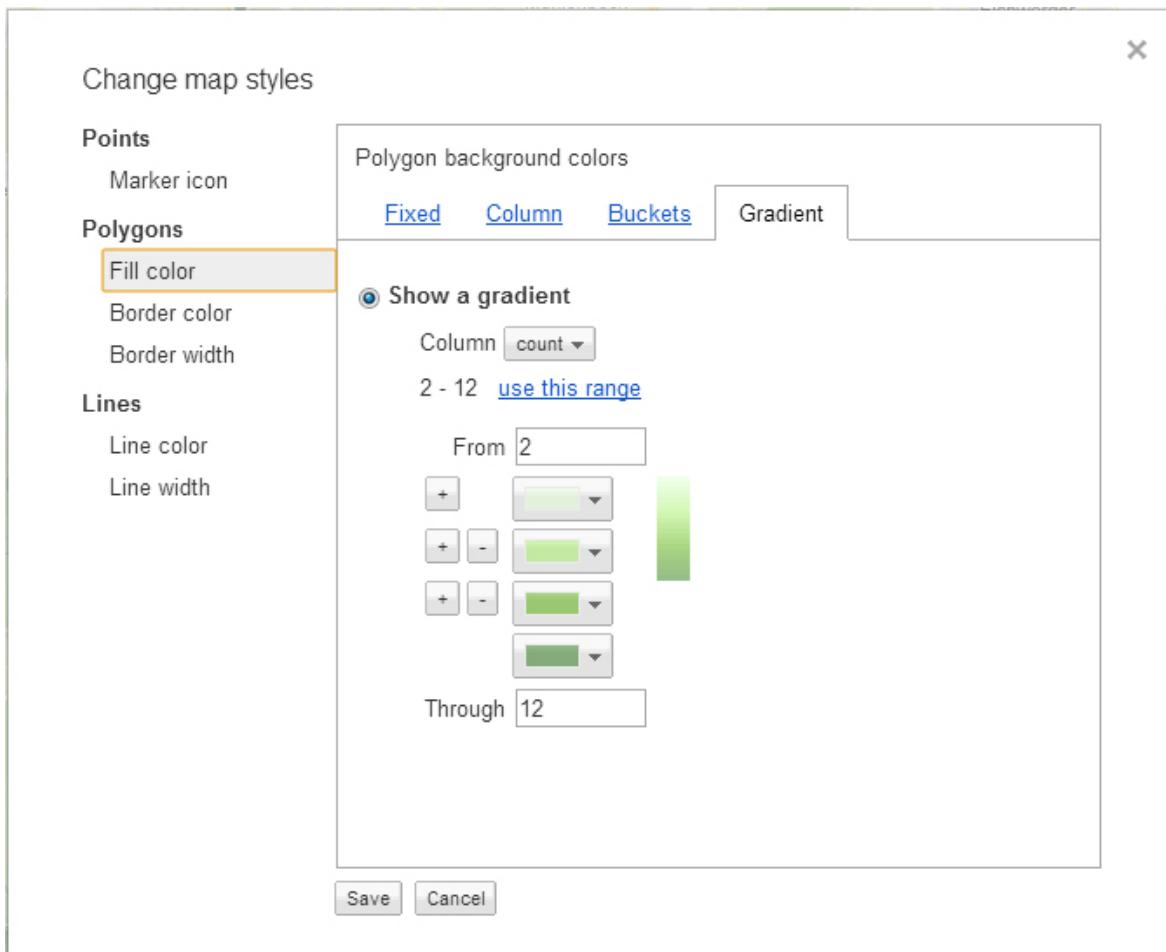


Abbildung 14: Change map styles: Show a gradient

Über die *Column*-Angabe wird nun die Tabellenspalte ausgewählt, die als Basis für die Farbberechnung verwendet werden soll. Fusion Tables analysiert die Daten und unterbreitet automatisch Vorschläge für die Maximal- und Minimalwerte. Über die einzelnen Farbfelder kann nun der Farbverlauf entlang der Messwerte individuell angepasst werden. Mit einem Klick auf „Save“ werden die Einstellungen gespeichert und der „Map style“-Dialog geschlossen. Das Resultat der Style-Konfiguration ist eine Heatmap, auf der ein Bezirk umso dunkler eingefärbt ist, je mehr Weihnachtsmärkte er hat:

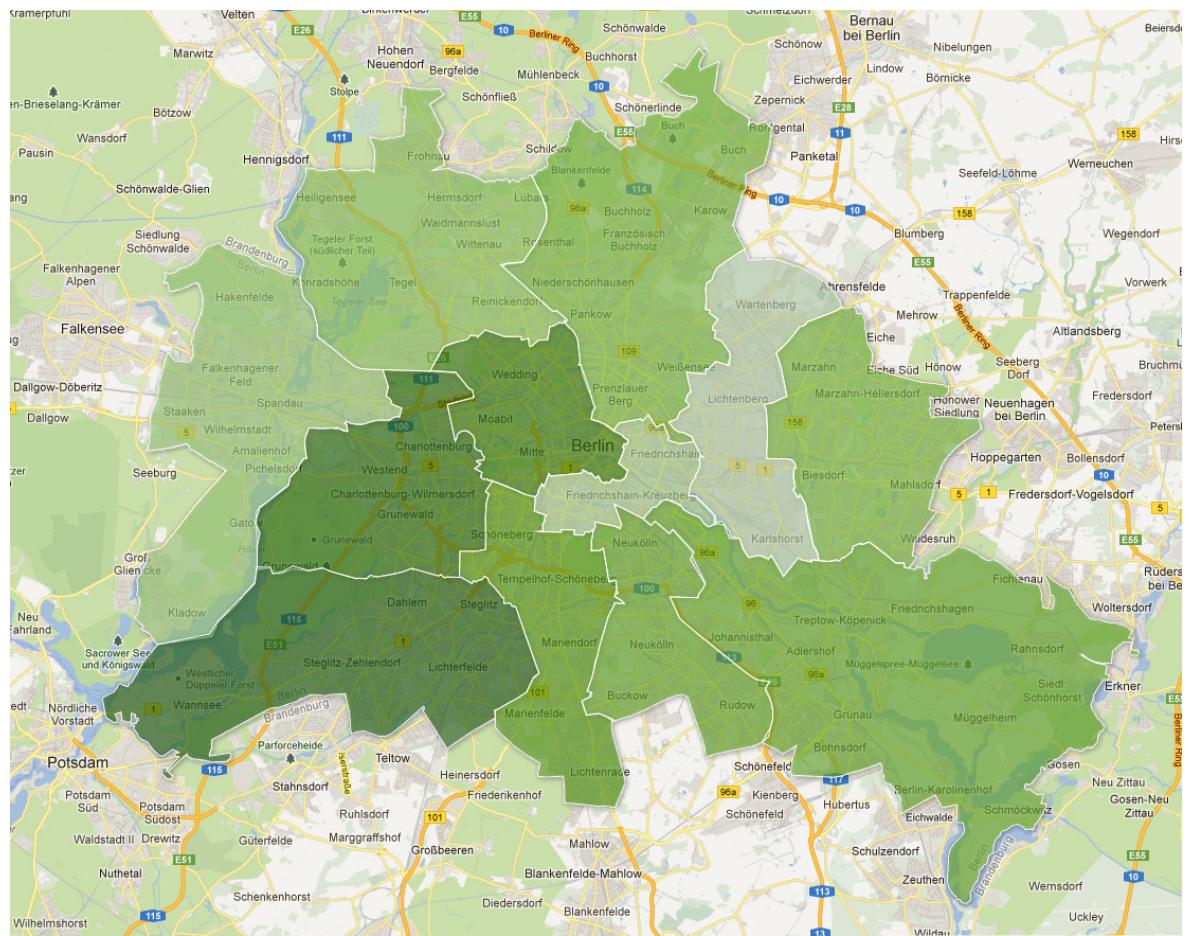


Abbildung 15: Heatmap: Anzahl Weihnachtsmärkte pro Bezirk

Ebenso wie bei den Diagrammen, werden für Karten und Heatmaps alle Style-Einstellungen automatisch gespeichert. Natürlich lassen sich auch diese Datenbilder wieder via URL oder iFrame einbetten. Über den Menüpunkt „Publish“ lässt sich auch der für die Einbettung nötige HTML- und JavaScript-Code anzeigen und kopieren. Die Datenkapselung der Kartensvisualisierung erfolgt dabei in Form eines eigenen Fusion Tables Layer-Objektes, das über die API referenziert werden kann.

4.4 HTML-Prototyp

Kernstück des Wissenschaftlichen Projekts ist die Implementierung eines Prototypen der die in Kapitel 3.2 beschriebenen Anforderungen berücksichtigt. Der Prototyp wurde in HTML und JavaScript umgesetzt. Für die Realisierung der Funktionalitäten wurden zusätzlich zu Google Fusion Tables die folgenden Javascript-Bibliotheken und APIs verwendet:

- jQuery (umfangreiche Javascript-Bibliothek, die komfortable Funktionen zur DOM-Manipulation zur Verfügung stellt und das Absetzen von REST-Anfragen an Google Fusion Tables ermöglicht)
- jQuery UI (bietet Lösungen zur Gestaltung und Funktionalität der Benutzeroberfläche)

- Google Maps API (Integration und Visualisierung auf der Weihnachtsmärkte auf der Karte)

Die Elemente zur Anzeigensteuerung wie auch die Visualisierung des Datencockpits sind auf Basis von HTML und CSS umgesetzt und lösen nach verschiedenen Events wie „Click“ oder „Change“ bestimmte Javascript-Funktionen zur Datenmanipulation aus. Zum Einsatz kommen hierbei Input-Felder, Auswahllisten, Checkboxen und Submit-Buttons.

4.4.1 Dynamische Auswahllisten

Im Rahmen der Filtermöglichkeiten ist vorgesehen, einen Stadtbezirk sowie auch das gewünschte Marktsortiment aus einer Auswahliste auszuwählen. Die Auswahl dient zur Filterung der in der Fusion Table hinterlegten Daten und muss daher vollständig konsistent sein. Aus diesem Grund empfiehlt es sich, die Auswahlisten direkt aus den vorhandenen Daten zu generieren und nicht gesondert und getrennt statisch im HTML anzugeben.

Wie in Kapitel 2.3 dargestellt, bietet Google einen REST-Webservice an, um spezielle Abfragen direkt über den Browser bzw. normale HTTP-Requests abzusetzen. Genau diese Funktionalität wird für das Generieren automatischer Listen aus Fusion Tables-Daten benötigt. Vor allem in diesen Anwendungsfall wird klar, dass Fusion Tables eine Datenbank ist, die mit gewohnten SQL-Statements abgefragt werden kann. Für die SQL-Abfrage via REST wird die finale URL, der SQL-Query mit Table ID und ein API Key benötigt. Um mit den Google APIs zu arbeiten und deren Missbrauch zu verhindern ist es notwendig, sich einen Google Developer Account und einen persönlichen API Key für die gewünschte API zu erstellen, in unserem Fall für die Fusion Tables API. Sind alle Voraussetzungen erfüllt wird noch eine Funktion benötigt, die einen HTTP-Request aus Javascript absetzt und das Ergebnis entgegennimmt (JSON-Format). Die zu Anfang erwähnte Bibliothek jQuery bietet genau diese Funktionalität mit `$.get()`¹³ an.

Die SQL-Query ist relativ simple und gibt eine Liste aller Bezirke gruppiert nach Bezirken zurück. Da die Fusion Tables SQL API keine DISTINCT-Funktion anbietet, um jeden Bezirk nur einmal zu selektieren, wird der Workaround mit GROUP BY verwendet, der in diesem Fall dasselbe Resultat hat:

```
var listDistrictsQuery = "SELECT bezirk FROM " + tableId +
                         " GROUP BY bezirk";

$.get('https://www.googleapis.com/fusiontables/v1/query?
      sql=' + listDistrictsQuery + '
      &key=' + apiKey,

      function(data) {
          listDistricts(data);
      });

```

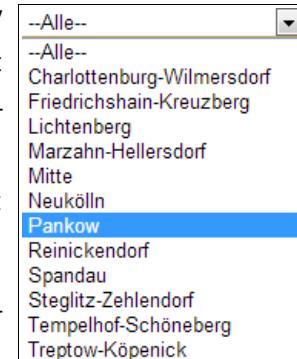
13 URL: <http://api.jquery.com/jQuery.get/> [13.01.2013]

Der zweite Parameter beschreibt das zurückgegebene Objekt (data), welches hierbei in einer anonymen Funktion direkt weiterverarbeitet wird, indem die Darstellungsfunktion „listDistricts(data)“ aufgerufen wird. Der Vorteil einer anonymen Funktion ist, dass diese nur für den einen Zweck der Weitergabe existiert und auch nur im Falle des Requests aufgerufen wird:

```
function listDistricts(data) {
    for (var i in data['rows'])
    {
        $('#district').append('<option>' + data['rows'][i] +
        '</option>');
    }
}
```

Innerhalb der Funktion „listDistricts(data)“ wird die dass Array data['rows'] durchlaufen und listet jeden darin enthaltenden Wert als eine „Option“ im Auswahlmenü. Als Resultat erhalten wir eine Auswahlliste mit allen Bezirken die innerhalb der Fusion Table existieren. Das „-- Alle --“ Element wird bereits fest im HTML-Markup angelegt und kann nicht generisch erstellt werden.

Analog wurde dieses Vorgehen auch für die Auswahlliste des Markt- sortiments verwendet.



```
<select id="district">
    <option value="all">--Alle--</option>
</select>
```

Filter

Bezirk

Markt sortiment

--Alle--
-Alle-
Bio
Handwerk
Kunst
Mittelalter
Nostalgisch
Standard

Abbildung 16: Prototyp: Filterlisten

4.4.2 Google Map und Fusion Tables

Neben den Elementen zur Anzeigensteuerung ist die eingebundene Google Maps das Kernelement des Datencockpits. Die Einbindung der Karte erfolgt in zwei Schritten: Im ersten Schritt ist es notwendig, mit Hilfe der Google Maps API ein Google Maps-Objekt beim initialen Laden des Prototypen zu erstellen. Die Erzeugung eines Objekts in Javascript erfolgt dabei auf ähnliche Weise wie in andern bekannten Programmiersprachen mit dem Unterschied, dass keine Typen-Deklaration notwendig ist. Dem Google Maps Konstruktor können darüber hinaus noch weitere Eigenschaften für die Darstellung der Karte mitgegeben werden, zum Beispiel der Mittelpunkt (center), Zoom-Stufe (zoom) und der Kartenstil (mapTypeId).

```
map = new google.maps.Map(document.getElementById('map-canvas'), {  
    center: new google.maps.LatLng(52.524577,13.393501),  
    zoom: 9,  
    mapTypeId: google.maps.MapTypeId.ROADMAP  
});
```

Im zweiten Schritt muss dann noch ein entsprechend gekennzeichnetes DIV-Konstrukt mit derselben ID „map-canvas“ vorliegen, in dem die Google Map dargestellt wird:

```
<div id="map-canvas"></div>
```

Über gewöhnliche CSS-Eigenschaften wie „height“ und „width“ kann anschließend noch die Größe der Karte gesteuert werden.

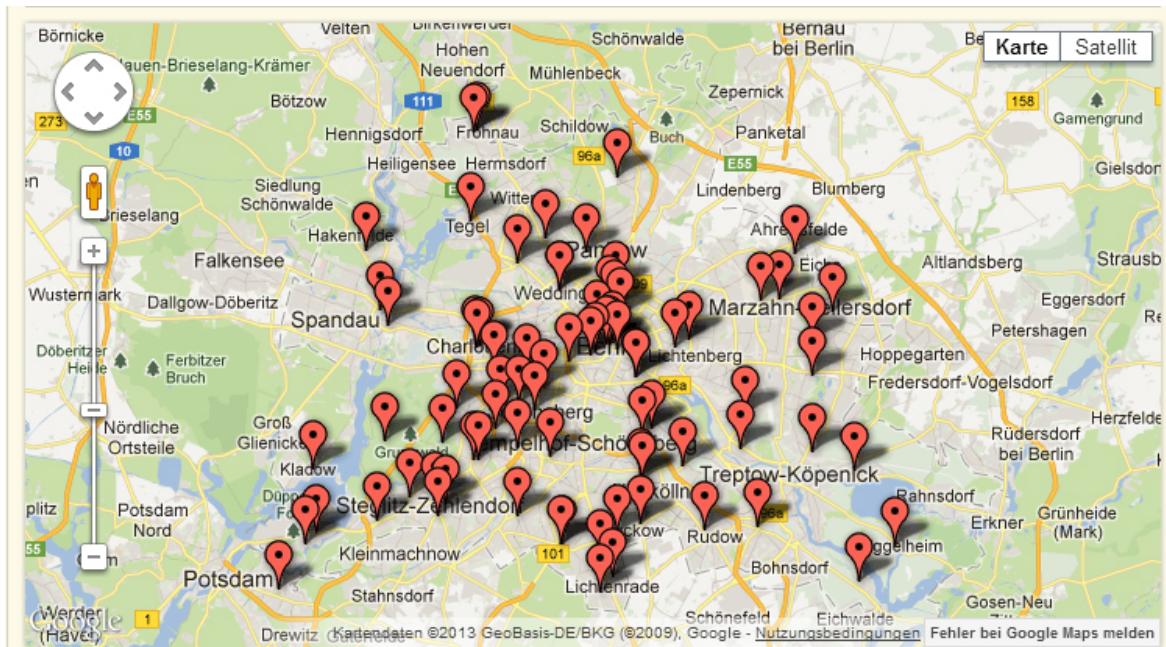


Abbildung 17: Prototyp: Eingebettete Google Map mit Positionsmarkern

Nach dem die Kartenbasis mit Google Maps integriert wurde, muss noch die in Google Fusion Tables hinterlegten Weihnachtsmärkte auf der Karte platziert werden. Google bietet hierzu sogenannte *Layers* an. Ein Layer ist ein Objekt, das auf Karten platziert werden kann und verschiedene Elemente beinhaltet. Bei dem Einsatz mehrerer Layer kann jeder Layer und somit die Darstellung der Elemente einzeln auf der Google Map gesteuert werden. Für die Verknüpfung mit Google Fusion Tables gibt es einen eigenen Typ namens *FusionTablesLayer*¹⁴. Dieser spezielle Typ stellt ein Interface zur Verfügung und sorgt für das automatische Darstellen der Fusion Tables Datensätze auf der Google Map, vorausgesetzt es existiert eine Spalte vom Typ „Location“. Diese Spalte heißt in unserem Datenbestand „geo_adresse“ und beinhaltet einer Verkettung der einzelnen normalisierten Adressdaten.

öffnungszeiten	Mo-Di 14:00-22:00 Fr-Sa 12:00-22:00
email	info@werbeteam-berlin.de
w3	www.werbeteam-berlin.de
bemerkungen	24.12. geschlossen
geo_adresse	Spandauer Damm, 14059 Berlin, Deutschland
eintrittspreis	0
eintrittspreis_ermässigt	0
sortiment	Handwerk
fahrgeschäfte	0

Abbildung 18: Geokodierbare Location-Angaben in der Spalte "geo_adresse"

Die Verbindung zur gewünschten Fusion Table erfolgt mittels der Angabe der eindeutigen Table ID. Die Table ID kann im Menü in der Fusion Table unter *File → About this table* abgelesen werden.

```
var tableId = '1rgZDIBtzhMZgBqzVwRMOvjs3RbeK2HE74bY9ivY';
var locationColumn = 'geo_adresse';

layer = new google.maps.FusionTablesLayer({
    query: {
        select: locationColumn,
        from: tableId
    },
    map: map
});
```

Neben der Spalte des Typs „Location“ und der Table ID in Form einer Query muss nun auch das zuvor erstellte Google Maps Objekt als Basiskarte übergeben werden.

Durch die Generierung unserer Auswahllisten, die Integration der Google Map und dem Erstellen eines speziellen Fusion Tables Layers auf Basis unserer Google Fusion Table sind alle Voraussetzun-

14 URL: <https://developers.google.com/maps/documentation/javascript/layers?hl=de#FusionTables> [23.01.2013]

gen für die Filterung und interaktive Steuerung gegeben.

4.5 Interaktive Anzeigesteuerung

Ziel des Datencockpits ist letztlich nicht die bloße Darstellung aller Weihnachtsmärkte als Adressmarker, sondern die Filterung nach Weihnachtsmärkten angepasst an die eigenen Bedürfnisse. Im vorherigen Kapitel wurden bereits alle dafür notwendigen Elemente erläutert und angelegt. Dieses Kapitel beschreibt daher das Auffangen der ausgelösten JavaScript-Events und der daraus resultierenden SQL-Anfragen an die Google Fusion Tables API.

4.5.1 Filterfunktionalität

Um die Weihnachtsmärkte auf der eingebettet Karten direkt und ohne das zusätzliche Betätigen eines „Absende“-Buttons je nach Filterwahl zu aktualisieren, werden JavaScript-Events eingesetzt. Jedes einzelne Filterelement hat hierzu eine eindeutige ID, um über einen Actionlistener auf bestimmte Aktionen dieses Elements zu reagieren. Für die Reaktion auf Nutzereingaben bietet die Google Maps API die browserübergreifende Ereignisverarbeitungsmethode `addDomListener()`¹⁵, mit der die meisten unterstützten DOM-Ereignisse des Browsers verarbeitet werden können.

Der Actionlistener für die Auswahl eines Bezirks wird auf folgende Weise registriert: Nach dem Auslösen des Events *Change*, wird die parametrisierte Funktion *updateMap()* aufgerufen, um die eingebettete Karte direkt zu aktualisieren. *Change* bezieht sich hierbei auf die Veränderung des gesetzten Wertes, also die Auswahl eines anderen Wertes im Auswahlmenü (siehe Kapitel 4.4.1).

```
//Bezirk

google.maps.event.addDomListener(document.getElementById('destrict'),
    'change', function() {

    updateMap(layer, tableId, locationColumn,"destrict");

});
```

Bis auf die Auswahl des Besuchsdatums wird jedes Filterelement mit dem Event *Change* abgefragt. Bei der Auswahl des Besuchsdatums kommt das Event *Click* zum Einsatz, da der Wert des Eingabefeldes ebenfalls mittels JavaScript gesetzt wird und somit das Event *Change* vom Browser nicht ausgelöst wird. Das *Click*-Event bezieht sich hierbei auf das nebenstehende Aktualisierungssymbol.

Die Funktion *updateMap()* bildet das Herz unserer Beispielapplikation und generiert verschiedene SQL-Abfragen auf Basis kombinierter Filtermöglichkeiten. Würden die verschieden gewählten Fil-

15 URL: <https://developers.google.com/maps/documentation/javascript/controls?hl=de> [06.02.2013]

ter nicht kombiniert werden, so würde die Karte immer nur die Weihnachtsmarktauswahl gemäß des zuletzt gewählten Filters anzeigen. Eine Kombination aus „Alle Weihnachtsmärkte im Bezirk Mitte“ und dem Marktsortiment „Kunst“ wäre demnach nicht möglich. Selbiges gilt für die anderen Filterfunktionen und das Besuchsdatum.

Die Funktion empfängt dabei die folgenden Parameter.

```
function updateMap(layer, tableId, locationColumn, element)
```

Layer ist unser Fusion Tables Layer, den wir verändern möchten. *Tableid* ist unserer Google Fusion Table und *locationColumn* die Spalte in der Fusion Table mit den Geokoordinaten. *Element* ist das Filterelement, welches letztlich das Event ausgelöst hat, um die Karte zu aktualisieren.

Mittels Switch-Case-Anweisung wird dann pro Filterelement eine andere Abfrage erzeugt. Pro Element wird als erstes das Where-Statement zusammengesetzt, da alle Filter letztlich über das Where-Statement beschrieben werden können. Hierbei wird lediglich unterschieden, ob alle oder nur ein bestimmtes Element angezeigt werden soll. Anschließend wird je Filterelement das zuletzt gebildete Where-Statement in einem assoziativen Array namens *query[]* lokal abgelegt. Nur auf diese Weise ist es möglich, über eine weitere spezielle Funktion *getWhereStatements()* die SQL-Abfrage zu formulieren und dabei alle gewählten Filter, wie z.B. Bezirk, Marktsortiment und ein Datum in einer SQL-Abfrage zu formulieren.

```
case "destrict":  
    var whereStatement = (value=="all" ? "" : "bezirk = '" + value + "'");  
    queries["destrict"] = whereStatement;  
    layer.setOptions({  
        query: {  
            select: locationColumn,  
            from: tableId,  
            where: getWhereStatements()  
        }  
    });  
break;
```

Beispiel der final gebildeten SQL-Abfrage für alle Weihnachtsmärkte im Bezirk Mitte mit dem Sortiment Kunst, kostenlosem Eintritt und Fahrgeschäften am 12.12.2012:

```
SELECT geo_adresse FROM tableid WHERE bezirk = 'Mitte' AND sortiment = 'Kunst' AND  
eintrittspreis = '0' AND fahrgeschaefte = '1' von <= '12/12/2012' AND bis >=  
'12/12/2012'
```

Die finale SQL-Abfrage wird dann mit Hilfe der Funktion `setOptions()` der Google Maps API direkt über den Parameter `query` an den gewünschten *Layer* übergeben. Die SQL-Abfrage selektiert hierbei die Geokoordinaten aus der Spalte „*geo_adresse*“ für alle Datensätze, welche die Bedingungen des Where-Statements erfüllen. Zusätzlich wird der Layer auf Basis des Abfrageergebnisses aktualisiert.

4.5.2 Einblenden der statistischen Werte

Als ein weiteres Feature beinhaltet das Datencockpit die visuelle Ausgabe von Statistiken, die exemplarisch die Anzahl der Weihnachtsmärkte und die durchschnittlichen Glühweinpreise pro Bezirk ausgibt.

Im ersten Schritt ist es notwendig, einen separaten Google Maps Layer auf Basis der zu visualisierenden Daten anzulegen. Die vollständige Visualisierung und Auswahl der Daten erfolgt in der Web App und nicht auf Code-Ebene (siehe Kapitel 4.3).

```
//Anzeige der Statistik Anzahl Weihnachtsmaerkte
layerEvaluationCountDistricts = new google.maps.FusionTablesLayer({
    query: {
        select: 'geometry',
        from: tableIdHeatmap
    },
    styleId: 2,
    templateId: 2
});
```

Für die Visualisierung der Statistik pro Bezirk gibt es die Möglichkeit, jeden Bezirk individuell zu färben und somit beispielsweise direkt auf die Verteilung der Weihnachtsmärkte pro Bezirk zu schließen. Die Berliner Bezirke liegen hierfür als Polygone im KML-Format in der Spalte *geometry* in einer gesonderten Fusion Table vor. Wie bereits im Kapitel 4.3 erläutert, können mit Hilfe der Web App bereits Charts, Maps und andere Auswertung per Klick erstellt werden. Je Visualisierung, die in einer Fusion Table in einem separaten Reiter dargestellt wird, kann über den Parameter `templateId`¹⁶ darauf zugegriffen werden. Jeder Tab, ganz gleich ob Chart, Map, etc. entspricht einem Template mit einer eindeutigen ID, beginnend bei der Nummer 0. Die Funktion „Cards“ ist allerdings hiervon ausgeschlossen und wirkt sich nicht auf die Nummerierung der Templates aus.

Selbiges gilt auch für die Visualisierung eines Reiters bzw. einer Map. Die eigenen Farbdefinitionen werden in der Fusion Table abgelegt und können mittels `styleId`¹⁷ über die API ausgelesen werden. Leider gibt es bisher keine Möglichkeit die verschiedenen IDs für Templates und Styles in einer Lis-

16 URL: <https://developers.google.com/fusiontables/docs/v1/reference/template> [02.02.2013]

17 URL: <https://developers.google.com/fusiontables/docs/v1/reference/style> [02.02.2012]

te einzusehen. Die Parameter erhält man nur über die Funktion „Publish“ und die angehängten Parameter im finalen Embed-Code. Letztlich bietet die API auch die Möglichkeit, jegliche Style-Definition direkt auf Code-Ebene zu erstellen, jedoch ist es für die Anwender deutlich einfacher das Styling via Web App durchzuführen.

Nachdem also der neue Fusion Tables Layer in der Anwendung initialisiert wurde, kann er durch einfaches Zuweisen zum Map-Objekt eingeblendet werden.

```
layerEvaluationCountDistricts.setMap(map);
```

Um Überlagerungen mehrerer Layer zu vermeiden, kann ein Layer mittels obj.setMap(null) deaktiviert werden, was zu empfehlen ist:



Abbildung 19: Überlagerte Darstellung mehrerer Layer in Google Maps

4.6 Einschränkungen/Ausblick

Grundsätzlich bietet Google Fusion Tables alle Möglichkeiten, um Filterungen und Visualisierungen der Datenbestände zu realisieren. Voraussetzung ist jedoch immer ein aufbereiteter Datenbestand, der die späteren Filtermöglichkeiten durch explizite Werte und Spalten beschreibt. Während der Implementierung einer Anwendung sollte außerdem eine klare Trennlinie zwischen Applikationslogik und Darstellung erfolgen. Die Gestaltung via Web App, aber auch auf Code-Ebene führen letztlich zu einer kaum noch nachvollziehbaren Implementierung. Es ist unklar an welcher Stelle dann welche Definitionen spezifiziert wurden. Durch die einfache Handhabung der Web App empfiehlt es sich, die Visualisierung solange es geht auch in der Web App vorzunehmen.

Die Implementierung von Events, der SQL-Logik und der HTML-Elemente sollten strikt auf Code-Ebene passieren und nicht mittels verfügbarer Konfiguratoren vorkonfiguriert und via iFrame eingebunden werden. Wie wir bereits gezeigt haben, lassen sich nur so auch beliebige Szenarien realisieren.

Auch die Beispielanwendung unseres Datencockpits bietet Potential für mehr: Beispielsweise könnten die dargestellten Filtermöglichkeiten ebenfalls in deren Abhängigkeit reduziert werden (wie ursprünglich auch in Kapitel 3.2 spezifiziert), so dass keine Falscheingaben mehr möglich sind. Weiterhin wäre die Erweiterung um beliebig viele Filter oder weiterführende Informationen denkbar, beispielsweise welche Sehenswürdigkeiten sich in der Nähe zu einem Weihnachtsmarkt befinden, Gastronomietipps, Verkehrsanbindung, etc.

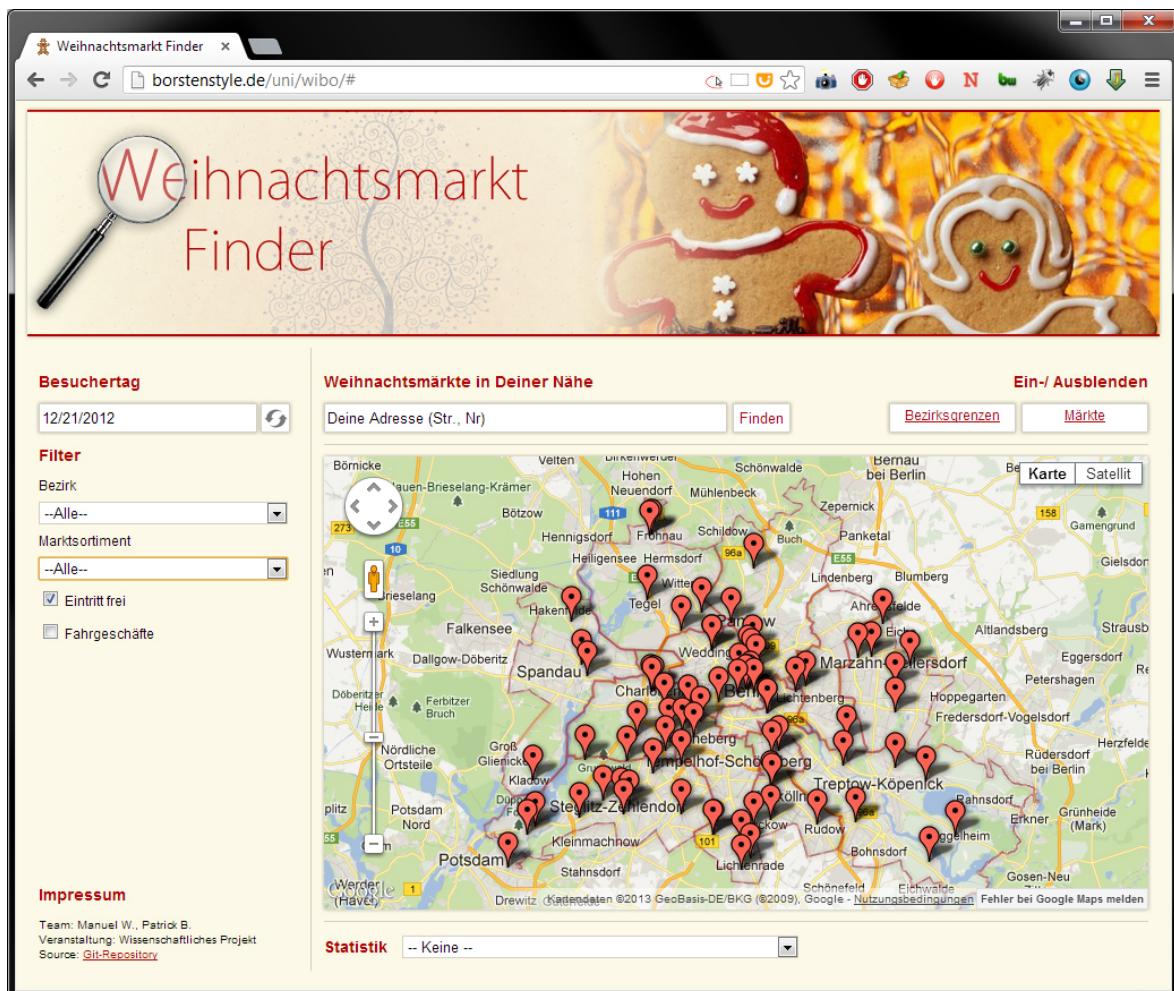


Abbildung 20: Finaler Prototyp

5 ZUSAMMENFASSUNG

Google Fusion Tables bietet in erster Linie ein mächtiges Werkzeug für die Verbindung von statistischen Daten und Google Maps. Die dynamische Einbindung von Werten in das Kartenmaterial von Google ermöglicht insbesondere durch die automatisierte Geokodierung und die zahlreichen Visualisierungstile (z.B. Heatmaps) interessante Anwendungsbereiche und erhellende Einblicke in die Datenbestände. Der Weihnachtsmarkt Finder-Prototyp hat gezeigt, welche Vorteile die Strukturierung und gezielte Filterung großer Datenbestände für den Zugang zum gespeicherten Wissen haben kann.

Die für die Umsetzung von Google zur Verfügung gestellten Werkzeuge Web App und Fusion Tables API bieten trotz oder gerade wegen ihres reduzierten Architekturansatzes interessante Tools für den Einsatz der Technologie. Mit Hilfe der zur Verfügung gestellten Methoden und gängigen Formate ist die API universell und programmiersprachenunabhängig einsetzbar. Methoden für das Absetzen eines HTTP-Requests sowie der Datenaustausch über JSON sind mittlerweile Standard in allen Programmiersprachen im Bereich der Webentwicklung. Obgleich sich das Projekt noch im Entwicklungsstadium befindet, ist es zudem durch die große Google-Community umfangreich dokumentiert (wenn auch durchgehend nur auf Englisch), was die Einarbeitung in die Thematik maßgeblich erleichtert. Für die schnelle und lebendige Visualisierung statistischer Daten mit geografischem Bezug im Netz ist wird sich daher wohl auf lange Zeit kaum eine bessere Infrastruktur finden lassen.

Die Auslagerung von Anwendungsdaten in die Cloud, also auf Google-Server, bringt natürlich auch Nachteile mit sich: Daten sind nur online verfügbar und können mitunter nicht entkoppelt lokal in eine Anwendung integriert werden. Ersten Recherche-Stichproben zufolge findet Fusion Tables daher bislang vor allem in browsergesteuerten Webanwendungen kleinerer Größe Anwendung und weniger in komplexen Software-Anwendungen. Datenschützer können zudem den Umgang mit Datentabellen monieren, da alle für die Darstellung verwendeten Tabellen mindestens indirekt über die Table ID im Quelltext öffentlich zugänglich sind. Der Umstand, dass AnwenderInnen bereitwillig beliebige Daten mit Geokoordinaten koppeln und diese Daten auch noch direkt auf Google Servern speichern, legt die Vermutung nahe, dass Google mit der Auswertung dieser Daten den Funktionsumfang seiner Services auch an anderer Stelle noch voranzutreiben beabsichtigt. Diese Überlegung stützt auch folgendes Zitat:¹⁸:

„The overarching goal of structured-data research at Google is to build tools that enable a rich ecosystem of structured data on the Web. To fuel such an ecosystem we need to build tools for discovery, extraction, annotation, sharing, querying, integration, visualization and publishing of structured data sets.“

18 URL: <http://research.google.com/pubs/DataManagement.html> [11.01.2013]

Ob sich Fusion Tables hingegen abseits seiner Google Maps-Features als Allround-Online-Datenbanklösung etablieren können wird, bleibt indes abzuwarten. Die Konkurrenz in diesem Marktsegment ist groß und der bislang erreichte Funktionsumfang des Services ist dafür noch zu rudimentär. Insbesondere die Benutzerfreundlichkeit der Web App lässt bisweilen arg zu wünschen übrig. Dabei ist besonders bedauernswert, dass die Fusion Tables App bislang so wenig Funktionalitäten der Google Spreadsheet-Lösung übernommen hat, obgleich beide Anwendungen sich mit ähnlichem UI präsentieren. Auch das Datenhandling bei Tabellen und Views insbesondere bei Summaries bietet noch viel Spielraum für Verbesserungen. Eine Auslagerung von Datenbeständen für eigene Anwendungen in Google Fusion Tables wird daher voraussichtlich in naher Zukunft keine Gefahr für existierende Datenbankmanagementsysteme (DBMS) darstellen.

LITERATUR/QUELLEN

- [1] Jason Baker, Chris Bond, James C. Corbett, JJ Furman, Andrey Khorlin, James S. Larson, Jean-Michel Leon, Yawei Li, Alexander Lloyd, Vadim Yushprakh. Megastore: Providing Scalable, Highly Available Storage for Interactive Services. In: Proceedings of the Conference on Innovative Data system Research (CIDR): 223-234, 2011.
URL: <http://research.google.com/pubs/archive/36971.pdf> [13.12.2012]
- [2] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data. In: OSDI'06: Seventh Symposium on Operating System Design and Implementation, Seattle, WA, November, 2006.
URL: <http://research.google.com/archive/bigtable-osdi06.pdf> [13.12.2012]
- [3] Hector Gonzalez, Alon Y. Halevy, Christian S. Jensen, Anno Langen, Jayant Madhavan, Rebecca Shapley, Warren Shen. Google Fusion Tables: Data Management, Integration and Collaboration in the Cloud. In: Proceedings of the Symposium on Cloud Computing (Industrial Track), Indianapolis, 2010.
URL: <http://www.cs.washington.edu/homes/alon/files/socc10.pdf> [13.12.2012]
- [4] Hector Gonzalez, Alon Y. Halevy, Christian Jensen, Anno Langen, Jayant Madhavan, Rebecca Shapley, Warren Shen, Jonathan Goldberg-Kidon. Google Fusion Tables: Web-Centered Data Management and Collaboration. In: Proceedings of the ACM SIGMOD conference, ACM, 2010.
URL: <http://homes.cs.washington.edu/~alon/files/sigmod10.pdf> [13.12.2012]

Google Fusions Tables API:

- Aktuelle Fusion Tables API 1.0:
https://developers.google.com/fusiontables/docs/v1/getting_started [02.12.2012]
- deprecated Fusion Tables API:
https://developers.google.com/fusiontables/docs/developers_guide [02.12.2012]

Open Data Portal:

- <http://daten.berlin.de/datensaetze/berliner-weihnachtsmaerkte-2012> [13.12.2012]

Weihnachten in Berlin:

- <http://www.weihnachteninberlin.de/weihnachtsmaerkte/> [13.12.2012]