2018

Von

Pascal Röcker  
Oliver Bauer  
Raphael Lawo

Dokumentation   
Web Service

Party Planer API

Inhaltsverzeichnis

[1. Projektidee und -ablauf 1](#_Toc4927321)

[1.1. Grundidee des Webservices Party planer 1](#_Toc4927322)

[1.2. Projektplan 1](#_Toc4927323)

[2. BPMN-Modell 2](#_Toc4927324)

[2.1. Party planer 2](#_Toc4927325)

[3. Implementierung 6](#_Toc4927326)

[3.1. Anfragen an Gerichte und Cocktail API 6](#_Toc4927327)

[3.2. Aufbau eigener Partyplan API 6](#_Toc4927328)

[3.3. Web-Oberfläche 6](#_Toc4927329)

[3.4. Backend 7](#_Toc4927330)

Abbildungsverzeichnis

[Abbildung 1 BMPN-Modell Partyplaner 3](#_Toc3877867)

[Abbildung 2 Gerichte und Cocktail aufteilen 4](#_Toc3877868)

[Abbildung 3 Daten zusammenbauen 5](#_Toc3877869)

# Projektidee und -ablauf

## Grundidee des Webservices Party planer

Unsere Grundidee des Webservices ist es, dass man mithilfe diesem Webservice eine Party planen kann. Dort gibt man sein Lieblings Essen und Cocktail an und die dazugehörigen Rezepte werden einem angezeigt. Dafür ist eine API für Rezepte für Essen und eine API für Rezepte für Cocktails angebunden.

Diesen ganzen Webservice könnte man noch unter anderem mit der Google Maps API erweitern, um direkt die Location auf Google Maps anzeigen zu lassen.

## Projektplan

Direkt am Abend nach der ersten Vorlesung von Webservices (am 26 Februar), in der wir uns bereits als Gruppe zusammengefunden hatten setzten wir uns für einen Brainstorming-Termin zur Themenfindung zusammen. Hierbei wurde sehr schnell klar, dass die einzige wirklich für uns in Frage kommende Projektidee ein Party-Planer wäre. Da wir zu diesem Zeitpunkt noch keine Vorlesung zum Thema BPMN hatten, uns aber schon klar war welche Komponenten unser Party-Planer benötigen würde machten wir uns in der folgenden Woche direkt daran die wichtigsten Einzelkomponenten wie die Abfrage und Verarbeitung der Daten unserer Essens- und Cocktaildatenbanken oder den Docker zum Laufen zu bringen. Direkt nach dem Erlernen der BPMN-Grundlagen am 14 März setzten wir die erste Version unseres BPMN Modells zu unserem Webservice auf. Am Montag den 18. März war außerdem das letzte Treffen vor der Präsentation, an dem das BPMN-Modell endgültig fertiggestellt und eine Lauffähige Tech-Demo vorbereitet wurde. Zudem wurde an diesem Treffen das weitere Vorgehen bezüglich der Finalen Implementierung von API und Weboberfläche besprochen um eine problemlose Fertigstellung des Projekts garantieren zu können.

Während dem Projekt waren die Aufgaben wie folgt verteilt:

|  |  |
| --- | --- |
| Name | Aufgabenbereich |
| Oliver Bauer | Abfragen und Verarbeiten der Daten  Erstellung der Tech-Demo |
| Raphael Lawo | Frontend  Docker |
| Pascal Röcker | REST-API |

# BPMN-Modell

## Party planer

Abbildung 1 zeigt das BPMN-Modell, welches das Backend unsere Partyplan API aufzeigt. Der Kunde stellt eine Anfrage für ein Gericht oder Cocktail. Die API trennt die jeweiligen Strings auf, wie in Abbildung 2 zu sehen ist, bei einem fehlerhaften Inhalt wird direkt eine Benachrichtig an den Kunden gesendet und der Vorgang wird direkt abgebrochen. Ist die Eingabe korrekt, so wird jeweils die Gerichte-API und die Cocktail-API angesprochen. Die beiden API’s geben dann die Information in vorm einer JSON zurück. Diese beiden JSON Dateien bauen wir dann zu einer JSON Datei zusammen, abgebildet in Abbildung 3. Als Rückgabewert hat unsere API ein JSON-Objekt, in dem allem Information enthalten sind. Zudem werden die Gerichte und Cocktails in einer Datenbank abgespeichert für spätere Wiederverwendung. Nachdem die Ausgabe an den Kunden und die Abspeicherung in die Datenbank abgeschlossen ist, ist der Prozess der API beendet.

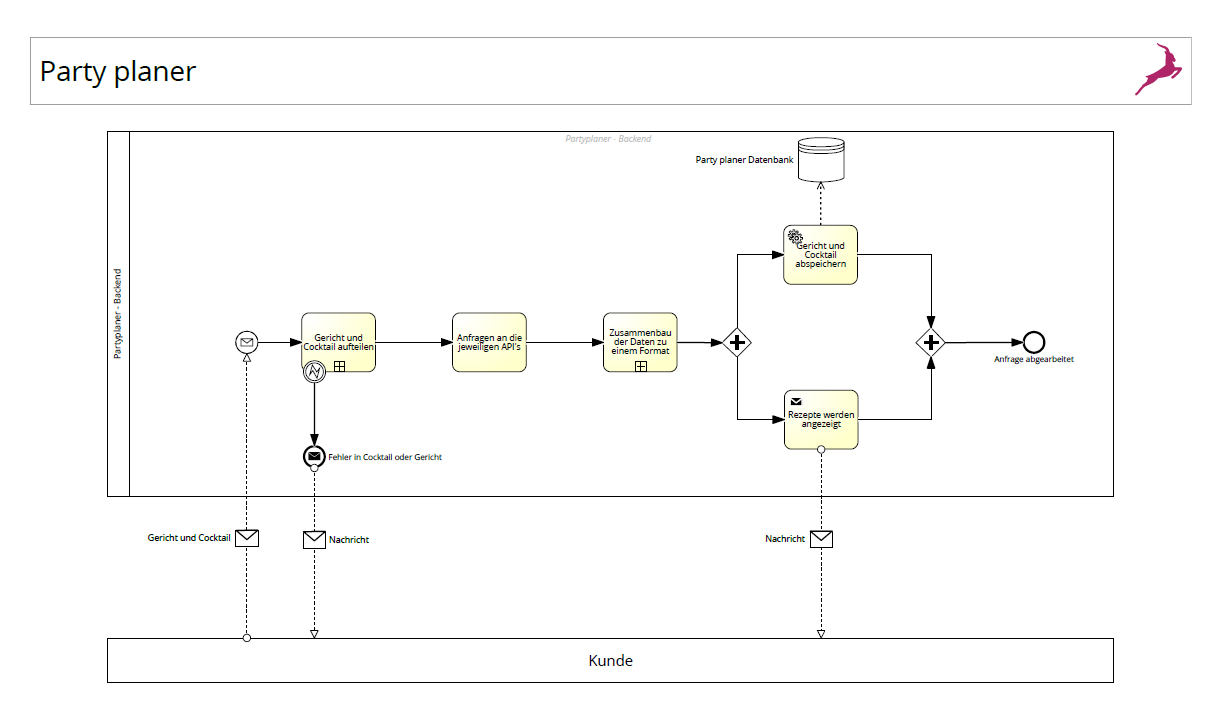


Abbildung 1 BMPN-Modell Partyplaner

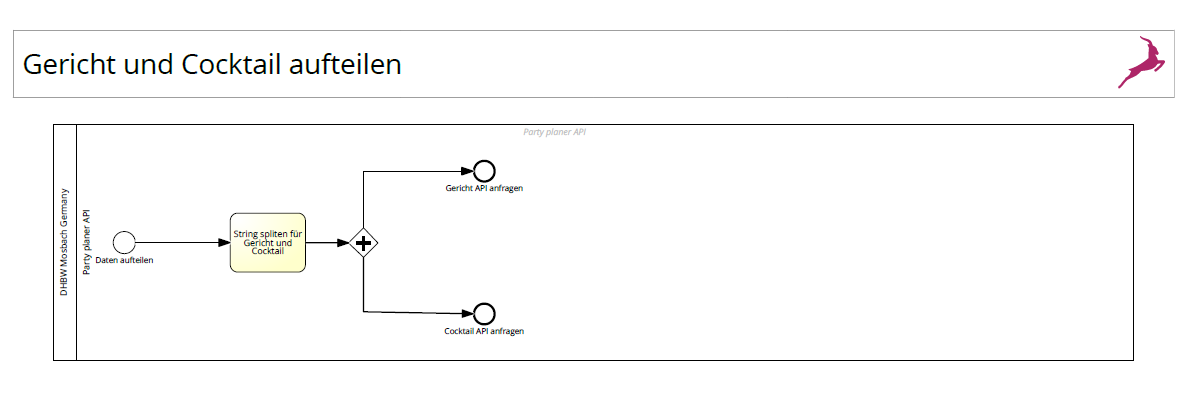


Abbildung 2 Gerichte und Cocktail aufteilen

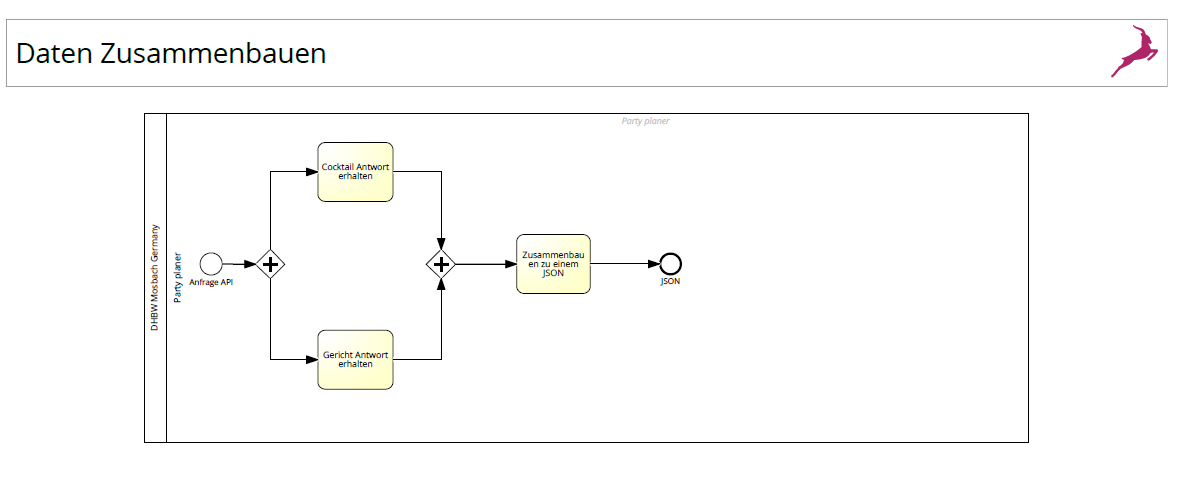


Abbildung 3 Daten zusammenbauen

# Implementierung

## Anfragen an Gerichte und Cocktail API

Die Abfolge zum Erhalten der Rezeptdaten von TheMealDB und TheCocktailDB erhält von der jeweiligen Datenbank-API ein JSON-Objekt mit den Zutaten und den Zubereitungsanweisungen, sortiert die Informationen aller Rezepte zu einer Liste zusammen und gibt die fertige Liste zur weiteren Verarbeitung weiter. Am Anfang dieser Abfolge wird jeweils ein String für die Gerichte und ein String für die Getränke der zuständigen Methode übergeben. Diese Strings werden dann zu einzelnen Strings mit den angefragten Zutaten-, Gerichts- oder Getränkenamen aufgeteilt, die Namen jeweils mit Hilfe der Grund-URI zu einer abfragbaren URI zusammengebaut und so für jede Kost mit Hilfe eines InputStreams die Informationen von der API abgefragt. Mit Hilfe eines BufferedReaders und einer einfachen StringBuilder-Schleife werden darauffolgend die von der API erhaltenen Informationen zu einem String zusammengesetzt, dieser zu einem JSON-Objekt umgewandelt und so ein JSON-Objekt für die jeweilige Kost erhalten.

Die für uns relevanten Daten aus diesen JSON-Objekten wie die Zutaten, die Menge der Zutaten und die Zubereitungsanweisungen werden dann zur einfacheren Verarbeitung in ArrayListen umgeschrieben und diese ArrayListen dann weitergegeben.

## Aufbau eigener Partyplan API

## Web-Oberfläche

Unsere Web-Oberfläche beinhaltet für den Anschauungszweck zwei Eingabefelder, in denen jeweils Gerichte und Cocktails eingetragen werden können. Die HTML-Ausgaben werden dabei dynamisch mit Hilfe von JavaServer Pages[[1]](#footnote-1) (JSP) erzeugt und die jeweiligen benötigten Informationen zur Laufzeit aus dem Backend bereitstellen zu können. Die jeweiligen Rezepte werden auf der Web Oberfläche angezeigt.

|  |
| --- |
| <% |
|  | String essen = request.getParameter("food"); |
|  | String drink = request.getParameter("drink"); |
|  |  |
|  | JSONObject jsonA = null; |
|  | JSONObject jsonB = null; |
|  |  |
|  | RecipeGetter recipe = new RecipeGetter(); |
|  | ArraylistToXML xml = new ArraylistToXML(); |
|  | //out.println("------------out.print--------"); |
|  | //out.println(request.getParameter("food")); |
|  | //out.println(drink); |
|  | ArrayList<String> stringsOfJSON = new ArrayList<>(); |
|  | ArrayList<String> reciepeList = new ArrayList<>(); |
|  | if (essen != null) { |
|  | jsonA = xml.arraylistToJSON(recipe.getDataFromAPIFood(essen)); |
|  | jsonB = xml.arraylistToJSON(recipe.getDataFromAPIDrinks(drink)); |

Der oben angezeigte gekürzte Code-Ausschnitt zeigt, wie die Informationen an das Backend übermitelt werden und das Backend die Daten für das Frontend wieder bereit stellt. Dabei werden jeweils aus dem Eingabefeld „food“ und „drink“ die Strings rausgenommen und an das Backend mit Hilfe der Methoden „getDataFromAPIFood“ und „getDataFromAPIDrinks“ übermittelt. Diese beiden Methoden haben als Rückgabewert ein JSON-Objekt und diese JSON-Objekte bauen wir zu einem zusammen und geben das dem Kunden zurück, wie in Kapitel 2 BPMN-Modell zu erwähnt.

JSP wird in HTML mithilfe von „<%“ eingeleitet und mit „%>“ wieder geschlossen. Innerhalb dieses Bereichs ist es erlaubt JavaCode zu schreiben, der auch ausgeführt wird. Dadurch kann man dynamisch Inhalte übermitteln und auch anzeigen lassen.

Zudem wurde das Framework Bootstrap[[2]](#footnote-2) mit dem Framework Material Design[[3]](#footnote-3) verwendet. Die Vorteile von Bootstrap sind, dass die Weboberfläche direkt Responsive ist, wenn man es richtig verwendet. Das geschieht dadurch, dass Bootstrap auf ein Grid-Layout basiert und dort in den jeweiligen CSS-Klasse Größen gesetzt sind, wie es aussehen soll. Durch dieses Grid-Layout lässt sich auch eine Website sehr schnell aufbauen und sieht auch sehr professionell aus. Durch dieses ganze Framework wird einem sehr viel Arbeit abgenommen.

Das Material Design Framework für Bootstrap bringt noch weiter Funktionen mit sich, unteranderem neue Styles.

## Backend

Als Backend wurde ein Thorntail[[4]](#footnote-4) Webserver verwendet der auch unter Linux läuft. Dass der Webserver unter Linux lauffähig ist, ist deshalb wichtig, da der ganze Webservice in einem Docker-Container läuft. Docker ist eine Linux Applikation die Container bereitet stellt für Anwendungen. Dadurch ist es nicht erforderlich, immer einen neunen Server bereitzustellen, wenn man eine neue Applikation hat, diese Applikationen können einfach in einem Container innerhalb von Docker gestartet werden.

1. JSP: https://de.wikipedia.org/wiki/JavaServer\_Pages [↑](#footnote-ref-1)
2. Bootstrap: https://getbootstrap.com [↑](#footnote-ref-2)
3. Material Design Bootstrap: https://mdbootstrap.com [↑](#footnote-ref-3)
4. Thorntail: https://thorntail.io [↑](#footnote-ref-4)