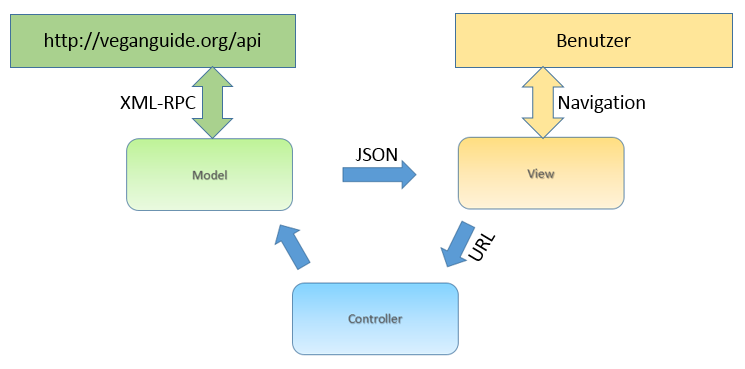
# Backend

## Struktur

Die serverseitige Struktur des Quellcodes folgt den Vorgaben des MVC-Prinzips. Durch die Einteilung in die drei Grundeinheiten „Datenmodell“ (engl.: „Model“), „Präsentation“ (engl.: „View“) und „Programmsteuerung“ (engl.: „Controller“) werden einige Präferenzen erzielt. Die für die Software Entwicklung wesentlichen Vorteile sind nachfolgend genannt:

1. Die Programmlogik ist von der Datenpräsentation getrennt
2. Der Quellcode ist übersichtlich
3. Die Software ist leicht zu warten

Das nachfolgende Schaubild stellt vereinfacht die Struktur der Grundeinheiten der Veganguide Anwendung und deren Kommunikation untereinander dar.



**View:**

Die Präsentation der Daten ist Aufgabe des Views. Angewendet werden statisches HTML und CSS. Somit ist sichergestellt, dass die Web Anwendung nicht lediglich als solche genutzt werden kann, sondern auch mit jedem Internet Browser verwendbar ist, der den Minimalanforderungen entspricht. Die für die Interaktion notwendige Dynamik entsteht durch die Skriptsprachen Javascript und JQuery. Der Endbenutzer navigiert durch die bereitgestellte Benutzeroberfläche und löst damit Funktionen aus. Jegliche Kommunikation vom View in Richtung Controller findet mittels URL-Adressierung statt.

**Controller:**

Der View kann mittels einer URL-Adressierung Daten vom Server anfordern. Dazu muss die URL einer vordefinierten Syntax entsprechen und die zu übermittelnden Parameter enthalten. So wird zum Beispiel die Anfrage mit dem Parameter „country“ und dem zu übermittelnden Wert „germany“ in der URL als „…/country/germany/…“ an den Webservice übergeben. Wenn der View eine Anfrage an den Server sendet, wird im Controller geprüft, ob die Syntax der URL stimmt und vordefiniert ist. Des Weiteren wird anhand der Parameter entschieden welche Methode der Weiterverarbeitung im Model passieren soll.

**Model:**

Das Model stellt mittels der übergebenen Parameter eine Antwort des Servers an den View zur Verfügung. Enthalten ist die standardisierte Logik der Datengewinnung, der Datenzwischenspeicherung („Backend Cache“) und der Bereitstellung der Daten in einer bestimmten Formatierung. Die Antwort an den View geschieht im JSON – Format und muss dementsprechend vom Model formatiert werden. Grundsätzlich kann bei der Veganguide Anwendung in drei Arten von Antworten des Servers an den View unterschieden werden:

1. Bei fehlerhafter Syntax der URL gibt das Model direkt eine Fehlermeldung an den View zurück.
2. Wenn die URL syntaxkonform ist, bei der Datengewinnung jedoch ein Fehler auftritt, wird der aufgetretene Fehler von der Web Schnittstelle als JSON formatiert und an den View weitergeleitet.
3. Wenn keine Fehler bei der URL oder der Datengewinnung auftreten, werden die gefragten Daten von der Webschnittstelle als JSON formatiert und an den View weitergeleitet.

## Framework

Die Implementierung der beschriebenen Struktur ist mit dem etablierten PHP - Framework „Silex“ des Herausgebers „Sensiolabs“ realisiert. Dieses stellt zahlreiche Funktionen zur Verfügung, die den Entwicklungsprozess deutlich vereinfachen. Die für die Entwicklung bedeutendsten und meistgenutzten Vorteile sind nachfolgend genannt:

1. Es werden aufschlussreiche Fehlermeldungen zurückgegeben.
2. Es steht eine Testumgebung zum Testen der Backend Funktionalität zur Verfügung.
3. Das Zuordnen der Anfragen („Requestmapping“) und dessen Verarbeitung läuft standardisiert.

## Datengewinnung

Die Veganguide Anwendung nutzt keine eigene Datenquelle sondern eine offene und nichtkommerzielle Datenquelle aus dem Internet. Diese bietet eine definierte Schnittstelle und ist erreichbar unter der Domain „http://veganguide.org“. Eine Schnittstellendokumentation ist unter „http://veganguide.org/api“ veröffentlicht. Die vorgegebene Kommunikation mittels XML-RPC bietet die Möglichkeit Anfrage-Parameter und Antwort-Parameter strukturiert im XML Format zu definieren.

Daten, die auf diesem Weg nicht abrufbar sind, werden durch die Nutzung eines RSS – Feets (z.B. die Funktion: „Neue Lokale“) oder durch Webscraping (z.B. die Funktionen: „Blog“ und „Blog-Kommentare“) erlangt.

Alle gewonnenen Daten liegen zunächst im XML – Format vor und müssen vom Backend an die definierte Schnittstellenstruktur angepasst und als JSON formatiert werden.

## Backend Cache

Das Backend Cache wird ebenfalls durch das Model realisiert. Um die Übertragungslast zwischen dem serverseitigen Prozess der Anwendung und der offenen Datenquelle im Internet möglichst gering zu halten, wird die bereitgestellte Antwort 24 Stunden als temporäre Datei zwischengespeichert. Falls dieselbe Anfrage mit identischen Parametern erneut den Controller erreicht, wird im Model lediglich die entsprechend identifizierbare temporäre Datei eingelesen und als Antwort an den View zurückgegeben. Somit wird die Performance bei wiederholten Aufrufen deutlich gesteigert. Als Identifikation dient der Name der Cache-Datei. Dieser wird gebildet aus der MD5 – Verschlüsselung der angefragten URL und ist damit eindeutig.