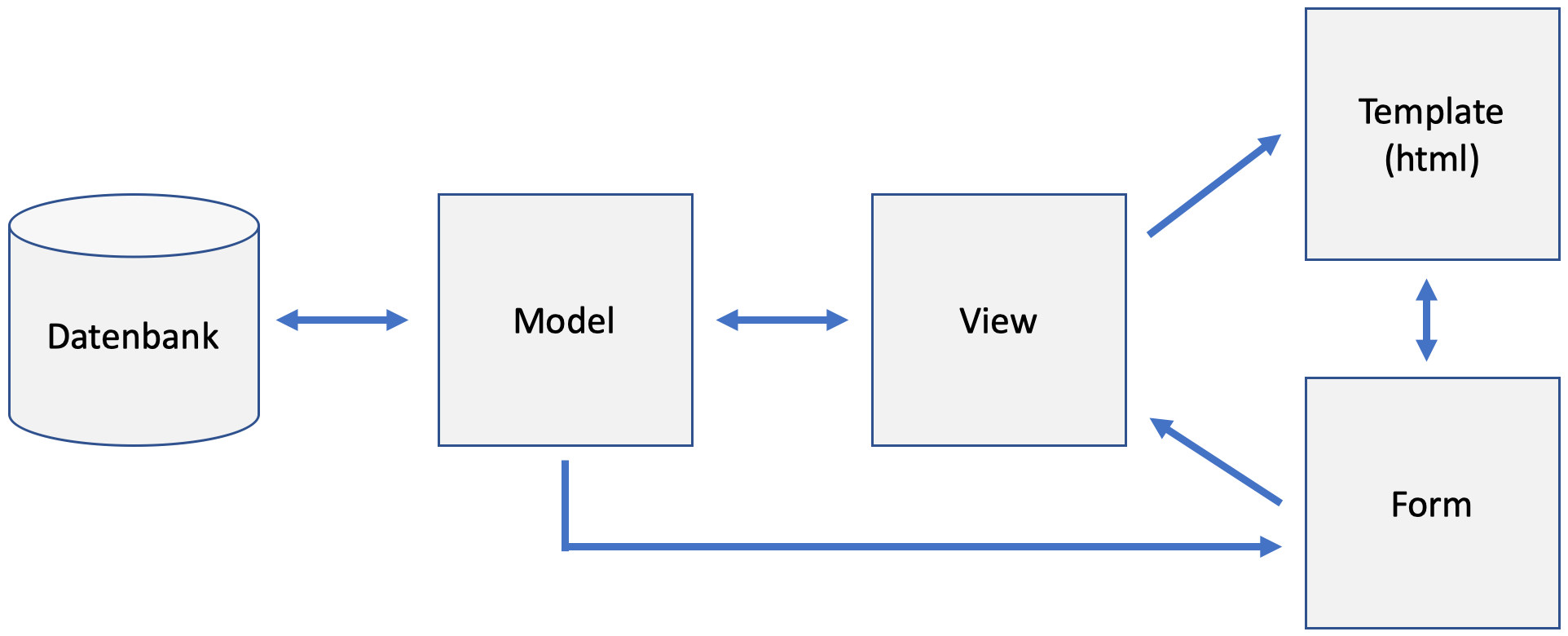
Django in aller Kürze

Ein herausragendes Feature von Django ist das Objekt-relationale Mapping (ORM): Der Entwickler beschreibt seine Objekte (Model) lediglich einmal in Python, um die Ablage in einer Datenbank kümmert sich anschließend Django vollautomatisch. Welche Datenbank genutzt wird, muss den Entwickler nicht kümmern. Django bietet Plugins unter anderem für MySQL und PostgreSQL, neu angelegte Projekte nutzen voreingestellt eine SQLite Datenbank.

Abfragen und Veränderungen an Django-Objekten werden durch Objekt-Methoden wie *save* oder *get* im Hintergrund zu SQL-Statements umgesetzt. Auch Fremdschlüssel-Beziehungen und Tabellen-Joins lassen sich durch entsprechende Deklaration im Django-Model einfach umsetzen.



Der schematische Zusammenhang zwischen den einzelnen Komponenten einer Django-Anwendung ist in Abbildung „model\_view\_grafik“ dargestellt: Das Model legt die Klassenstruktur und Beziehungen zwischen den Objekten fest. Im Hintergrund erledigt das Django-Framework den Abgleich mit einer Datenbank. Das einmal deklarierte Model kann anschließend als Vorlage für Views und Forms genutzt werden, getreu dem DRY-Prinzip: „don’t repeat yourself“. Bei den Views sind typische Anwendungssituationen wie Bearbeitung eines Objekts oder Listenansichten bereits im Framework vorhanden und erfordern im Bedarfsfall nur kleine Anpassungen.

Die Darstellung im Browser übernehmen Templates, die aus HTML, CSS und JavaScript bestehen können. Angereichert werden Templates mit der Django Template Language. Diese unterstützt die Ausgabe von Daten in den HTML-Code. Durch Schleifen- und If-Befehle können Inhalte auch dynamisch im Template erzeugt werden. Für ein einheitliches und modernes Erscheinungsbild im Browser empfiehlt sich der Einsatz eines Frontend-Frameworks wie z. B. Bootstrap.

Formulare (Forms) können ebenfalls sehr einfach mit Bezug zum Model erstellt werden. Sie stellen das Bindeglied zwischen Eingaben im Browser und der Verarbeitung im View dar. Auch hier wird dem Entwickler viel Arbeit abgenommen. Durch den Bezug zum Model und den dort definierten Datentypen bietet Django eine vorgefertigte Formular-Validierung, die Fehleingaben seitens der Anwender verhindert. In ein Datumsfeld kann beispielsweise auch nur ein Datum eingegeben werden. Darüber hinaus werden für jeden Datentyp bereits passende UI-Elemente (Widgets) wie Drop-Down Auswahlmenüs oder Textfelder ausgewählt.

Für Unternehmensanwendungen außerdem interessant sind die vorgedachten Features zur Mehrsprachigkeit und zur Berücksichtigung von unterschiedlichen Zeitzonen. Wenn zu guter Letzt sonst nichts hilft: Spezielle Anforderungen lassen sich durch Vererbung und Anpassung individuell ändern oder ergänzen. Dabei unterstützt eine aktive Community, z. B. bei Stack Overflow, die sich in den vergangenen 15 Jahren seit dem ersten Erscheinen des Frameworks herausgebildet hat. Viele nützliche Dinge, die das Django-Framework nicht von Haus aus schon erledigen kann, finden sich zudem in Form von installierbaren Apps. Einige spannende Erweiterungen speziell für Unternehmensanwendungen stellt der Artikel vor.

Beispielprojekt „Aufgabenverwaltung“

Abgesehen von der Active Directory Anbindung zeigen wir das Zusammenspiel der im Artikel vorgestellten Packages in einem Beispielprojekt, das eine einfache Aufgabenverwaltung realisiert. Jede Aufgabe besteht aus einem Kurz- und einem Langtext (optional) sowie einem Vorlagedatum, das zur zeitlichen Einordnung der Aufgabe genutzt werden kann. Zusätzlich verweisen Aufgaben auf einen Status und eine Liste. Der Status kann beispielsweise „aktiv“, „erledigt“ oder „wartend“ sein. Listen können Aufgaben noch einmal sortieren, zum Beispiel nach „Arbeit“, „Zuhause“, „Verein“ oder ähnlichem. Die möglichen Einträge von Status und Liste sind nicht vordefiniert, sondern werden im Admin-Bereich gepflegt (Abbildung „django\_admin“).

Den Quellcode zum Ausprobieren gibt es unter [3]. Getestet ist die Anwendung mit Python 3.8 und Django 3.1. Legen Sie den Quellcode in ein neues Verzeichnis und erzeugen Sie eine virtuelle Umgebung mit venv. Anschließend werden die erforderlichen Packages mit

*pip install -r requirements/dev.txt* installiert. Der für die Entwicklung in Django eingebaute Webserver startet anschließend durch: *python manage.py runserver*

Bildunterschrift:

„model\_view\_grafik“: Django nutzt konsequent das Model und erspart dem Entwickler so viel Tipp-Arbeit.