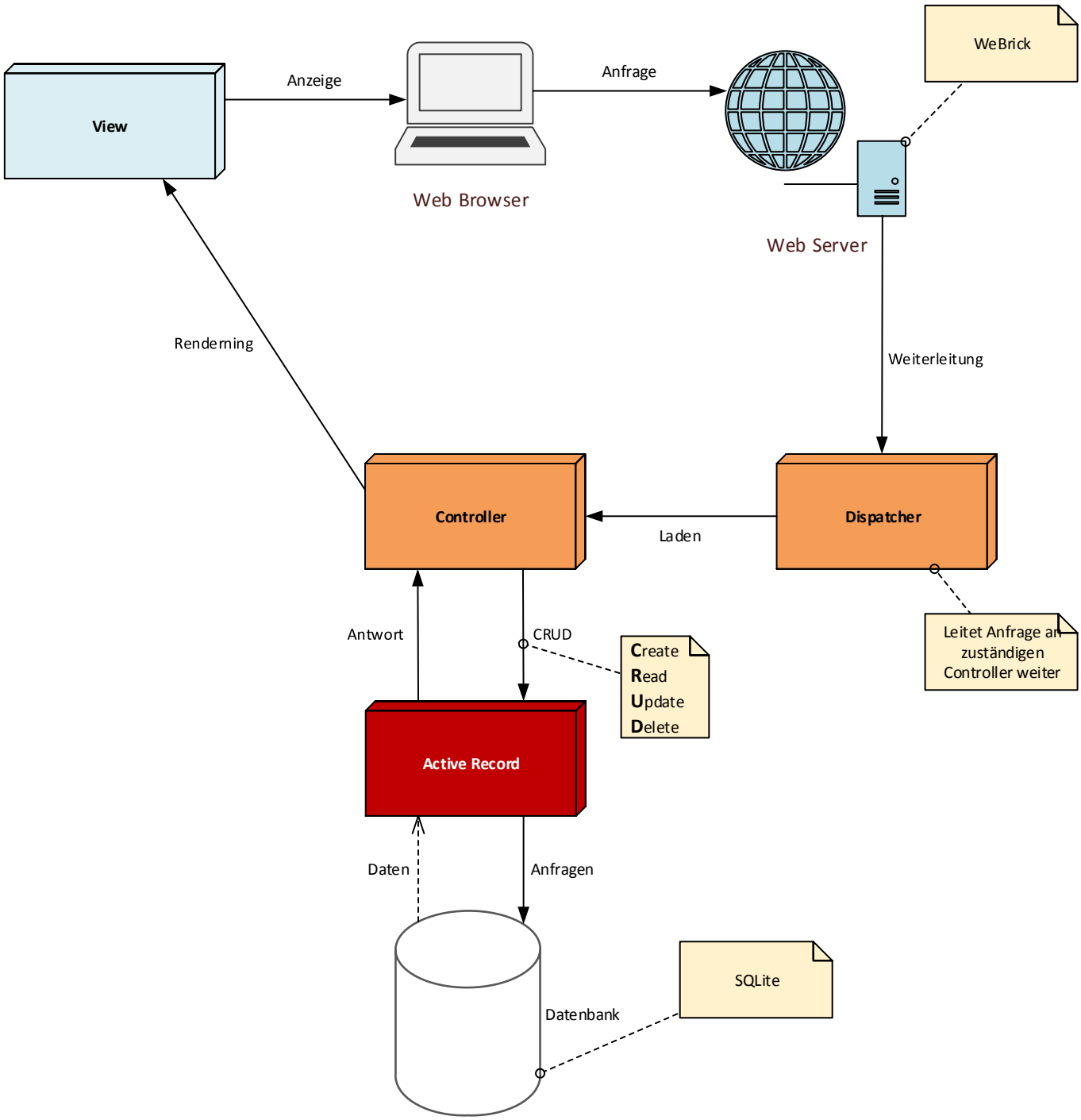
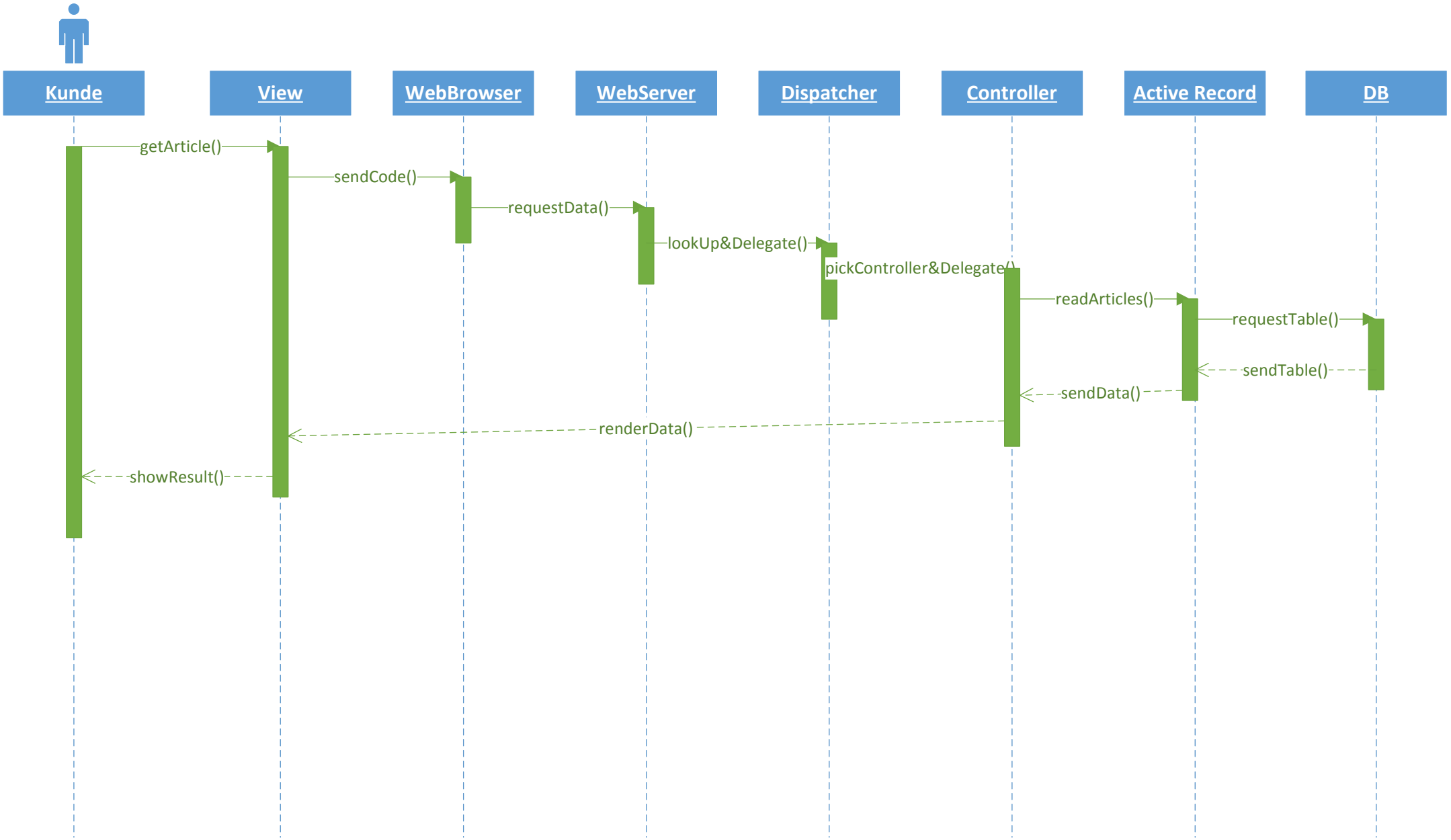
Betriebswirtschaft 2 Praktikum Ausarbeitung - Aufgabe 1

WS 14/15 – Marjan Bachtiari, Tim Hartig

Inhaltsverzeichnis

Architekturübersicht (mit Ruby On Rails)




Kunde
View
WebBrowser
WebServer
Dispatcher
Controller
Active Record
DB
getArticle()
sendCode()
requestData()
lookUp&Delegate()
pickController&Delegate()
readArticles()
requestTable()
sendTable()
sendData()
renderData()
showResult()

Warum “Ruby On Rails”?

BEGRÜNDUNG ZUR TECHNOLOGIEAUSWAHL

Zugrunde liegt die dynamisch typisierte Programmiersprache “Ruby”, für die wir uns aufgrund ihrer hohen Ausdrucksmächtigkeit (wenig Code, große Wirkung) entschieden haben. Zudem haben wir bereits damit im 1. Semester gearbeitet.

RAILS

Das in Ruby entwickelte Web-Application-Framework “Rails” nutzt die Model-View-Control (**MVC**) Architektur (siehe Architekturübersicht) und ist geprägt vom **CoC** (Convention over Configuration) Paradigma und damit auch vom **DRY** Prinzip (“don’t repeat yourself”), sowie **KISS** (“Keep it short [and] simple”).

Durch CoC wird die Komplexität von Konfigurationen reduziert, so dass durch Einhaltung von Namenskonventionen (Bsp. Modellname: Article, Tabellenname: articles) Beziehungen zwischen vielen Applikationsaspekten automatisch verwaltet werden können. So werden in Rails die zum Model zugehörige View und der Controller über den Namen vom Model automatisch zugeordnet.

BEISPIEL-MODELL ARTICLE:

Model → `/app/models/article.rb`

View → `/app/views/articles/{`*`Embedded-Ruby-Dateien für Views der Model-bezogenen Operationen`*`}`

Controller → `/app/controllers/articles_controller.rb`

Dank dieses Prinzips kann Rails, Generatoren für diese und andere Aspekte anbieten, mit denen diese Programmteile automatisch erzeugt und eingegliedert werden können, was einige Handarbeit erspart.

Zur Generierung von Inhalt auf Webseiten, kommt Embedded-Ruby (**ERB**) zum Einsatz, bei dem Ruby-Code - eingebettet in “*.html.erb” Dateien - evaluiert und ausgegeben werden kann.
<% AnweisungX %> veranlasst Rails die *AnweisungX* durchzuführen. Soll das Ergebnis einer *AnweisungY* auf dem Dokument ausgegeben werden, wird das ERB-Statement um nach dem ersten **%** Zeichen um ein **=** ergänzt → ***<%= AnweisungY %>***

Das bedeutet, HTML und Ruby verschmelzen miteinander und der Inhalt der Seite kann mit Ruby-Anweisungen dynamisch generiert werden (vgl. PHP-Script - nur viel einfacher).

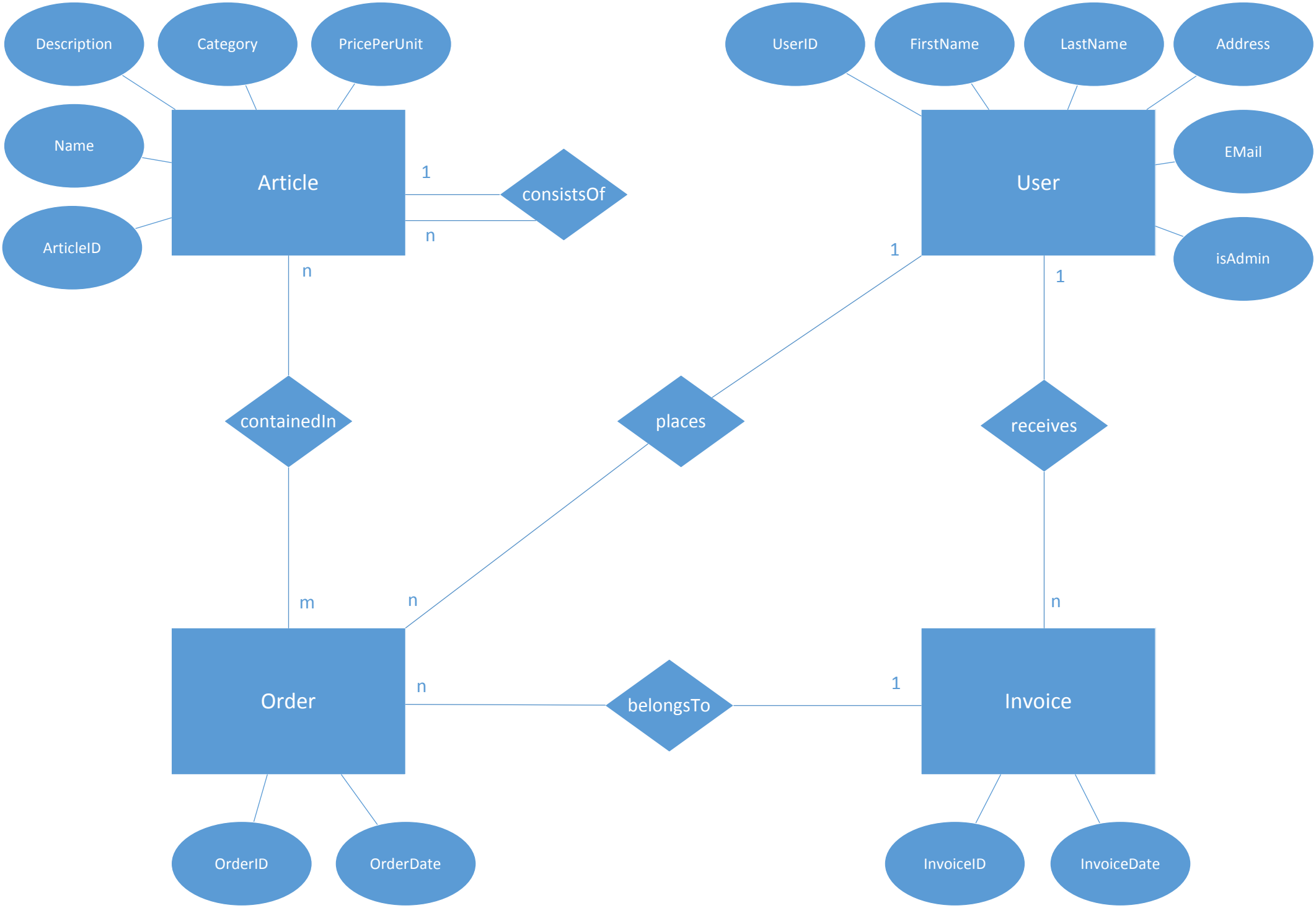
Zur kompakten Syntax von Ruby kommen zusätzlich Funktionen zur Generierung von häufig in der Webentwicklung benötigten Codetemplates, wie z.B. das Kreieren eines Links.

```
<%= link_to text_to_show, object_to_show %>
```

ACTIVE RECORD – SCHNITTSTELLE ZUR PERSISTENZ

Alle Modelldaten werden mittels in Ruby on Rails mit **Active Record** auf Datenbank-Datensätze gemappt (Objektrelationale Abbildung). Active Record abstrahiert vom benutzten Datenbank-System, so dass im Ruby-Code keine SQL-Statement formuliert werden müssen, um Daten in der Persistenz ablegen zu können. Die Übersetzung erfolgt über einen Adapter für das jeweilge Datenbanksystem. Dessen Auswahl erfolgt lediglich über eine Konfigurationsdatei.

Damit erhält man die Möglichkeit, das Datenbanksystem frei wählen zu können und ggf. auf ein anderes DBSystem umsteigen zu können, ohne die Implementationen des Projekts anpassen zu müssen.


Description
Category
PricePerUnit
Name
Article
ArticleID
1
consistsOf
n
n
UserID
FirstName
LastName
Address
User
EMail
isAdmin
1
1
containedIn
places
receives
m
n
n
n
Order
belongsTo
1
Invoice
OrderID
OrderDate
InvoiceID
InvoiceDate