Design Document for iExperiment

Peter Strauch Kevin Jacob Johannes Kunz
Jonas Barde

January 6, 2016

Contents

Part I

Low Level Design

Chapter 1

Einleitung

In diesem Dokument wird das zu entwickelnde System auf der Implementierungsebene betrachtet. Nachfolgend werden Klassendiagramme des Clients und der Servers aufgezeigt. Weiterhin folgen zu jedem Diagramm eine kurze Erläuterung zu den von uns genutzten Strukturen.

Dabei ist zu beachten, dass die Klassen User, Model, Package, Dataset, Forecast, Group und Algo sind in einem Paket zusammengefasst. Da sie zu viele Klassen bekannt sein müssen wurden in dem Diagramm die Assoziationen zur Gunsten der Übersicht nicht eingezeichnet. All diese Klassen erben von einer abstrakten Klasse Methoden die für alle relevant sind.

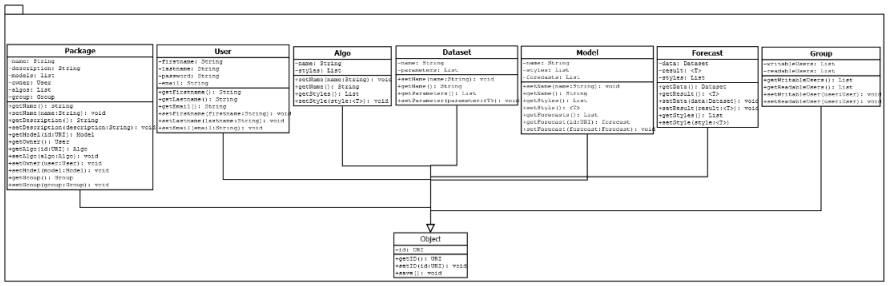

Figure 1.1: Das Paket mit den Klassen User, Model, Package, Dataset, Forecast, Group und Algo.

Chapter 2

Client

Der Client wird zum großen Teil mithilfe eines Model-View-Controller Patterns realisiert. Die verschiedenen View-Klassen stellen die Benutzeroberflche des Clients dar. Der Controller bestimmt bei Forecast, Model und Algorithm die Art der Darstellung. Beispielsweise kann der Nutzer entscheiden, ob die Forecast als Baumdiagramm, Kurvendiagramm, o.ä. ausgegeben wird. Selbiges gilt für die Models und Algorithmen. Bei Login und Admin haben die Controller die Aufgabe die möglichen Eingaben zu verarbeiten oder zu verifizieren.

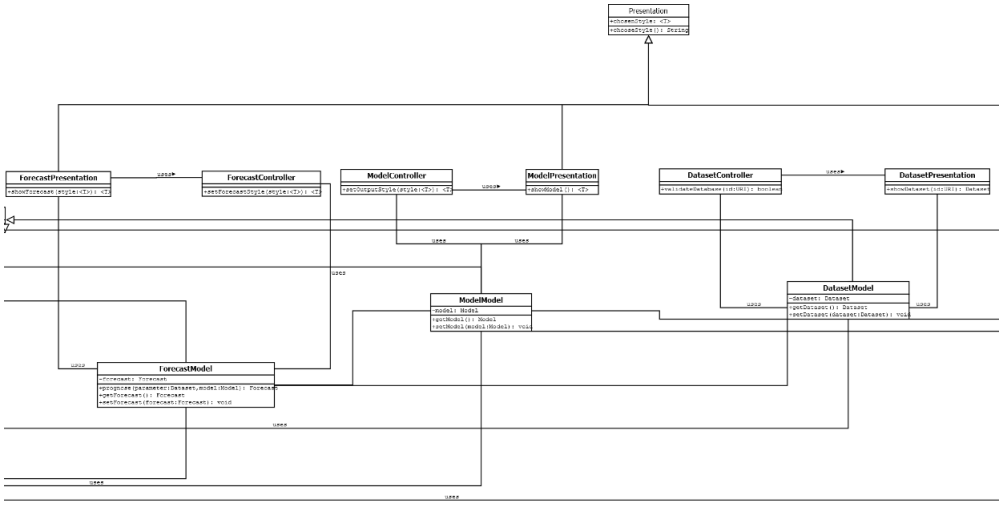

Figure 2.1: Ausschnitt aus der Client Ebene.

Chapter 3

Server

Wir verwenden Apache Tomcat als Webserver, dieser enthält ein Servlet-Container, mittels dessen wir mit dem Server kommunizieren. Als Datenbank verwenden wir eine RDF-Datenbank, diese wird durch Apache Jena TDB Triple-Store realisiert. Direkt unter den Servlets liegen die Interfaces der verschiedenen Objekte, dabei ist zu beachten, dass nur für Package, User und Group Manager nötig sind. Das liegt daran, dass die anderen Klassen (Model, Algo, Forecast und Dataset) über die geerbten Methoden alle benötigten Informationen, wie z.B. Packages abfragen können. Um mögliche Berechtigungen zu überprüfen kann in den Interfaces mit getCurrentUser der aktuelle Nutzer und somit auch seine Freigaben und Gruppen abgefragt werden.

Die Klassen ClientCrypter und ServerCrypter sind für die Ver- und Entschlüsselung der Informationen zuständig. Somit können keine Informationen direkt ausgelesen werden, falls Server und Client lokal voneinander getrennt sind.

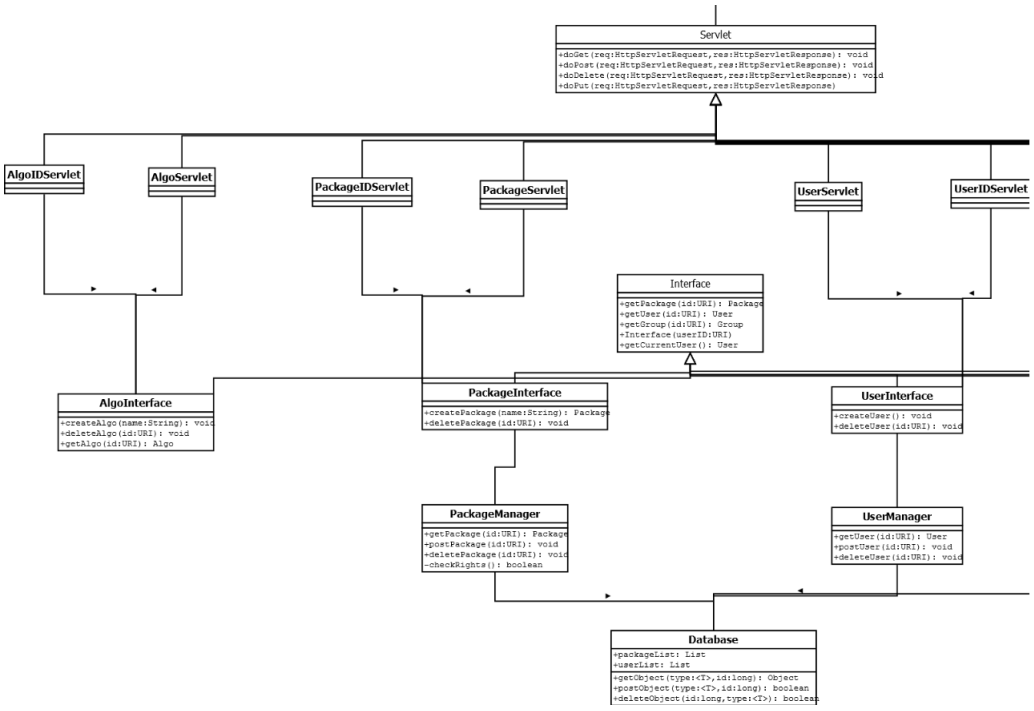

Figure 3.1: Ausschnitt aus dem UML Diagramm für die Serverseite. Hinzukommen noch Klassen für Dataset, Group und Model

Chapter 4

Beispiel mit Sequenzdiagramm

Im folgenden Sequenzdiagramm wird der Befehl

<server >: <port >/package/ <id >/model/ <id >

dargestellt. Dabei versucht der eingeloggt Nutzer ein selbsterstelltes Model aufzurufen, auf das er die nötigen Rechte besitzt.

Dieser Vorgang wird realisiert, indem die Anfrage vom ModelIDServlet entgegen genommen wird. Das Interface ruft den PackageManager auf, der das Package von der Database anfordert. Nachdem das Package angefordert wurde, wird der owner des Packages mit Hilfe der checkRights Methode mit dem currentUser verglichen. Gibt es dort eine Übereinstimmung, wird das Package an die oberen Schichten bis zum Client weitergegeben.

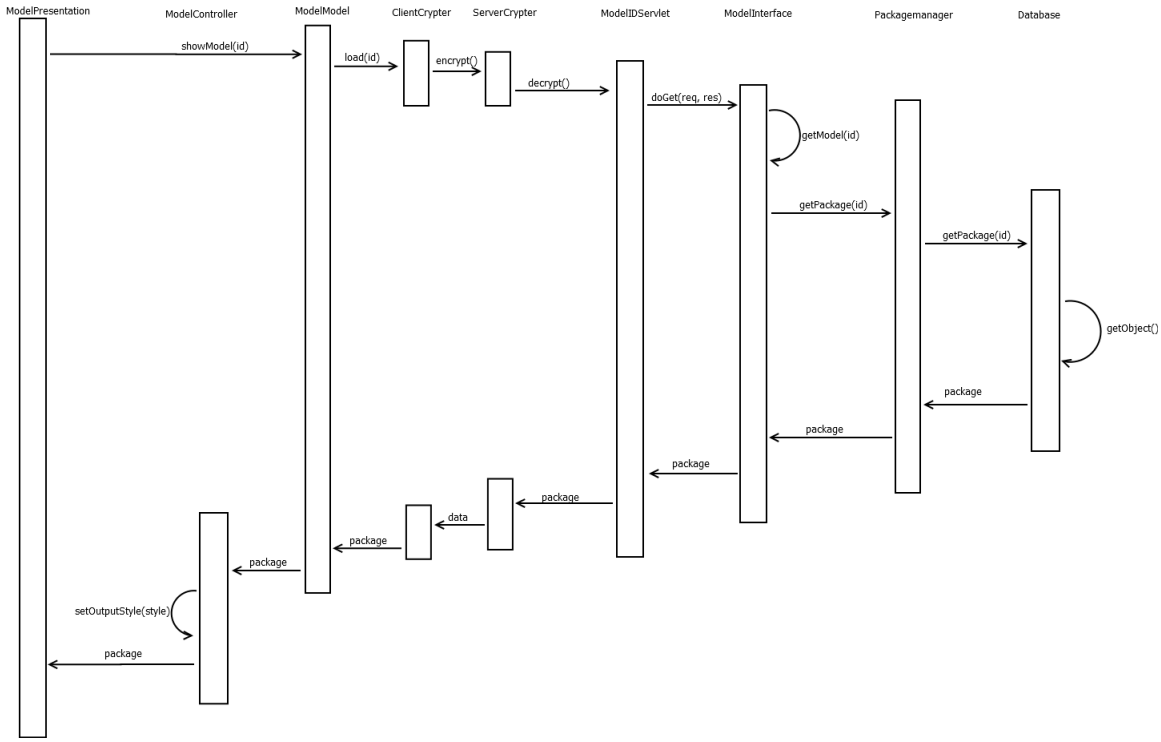

Figure 4.1: Sequenzdiagramm zum Aufrufen eines Models