# Anwendungsarchitektur

## Framework

Für das Projekt wurde eine Anwendungsinfrastruktur gefordert, die in Java umsetzt sein sollte, da das Know-How des Projektteams zu überwiegenden teilen im Bereich der Java-Entwicklung liegt. Außerdem sollte das Framework maximale Unterstützung bei der Umsetzung der anfallenden Tätigkeiten bieten.

Spring Roo schien zur Umsetzung am meisten geeignet, da es eine umfassende Unterstützung durch Generatoren viele allgemeine Aufgaben den Entwicklern abnimmt (z.B. CRUD-Prozesse), so dass sich das Projektteam vermehrt mit Problemstellungen der Daten- und Prozessmodellierung beschäftigen kann.

Spring Roo ist eine Erweiterung des Spring Frameworks und konzentriert sich auf das unterstützen von „rapid-development“ Entwicklungsprozessen durch Umsetzung der Prinzipien „Dont-Repeat-Yourself“ (DRY) und „Convention over Configuration“. Spring Roo ähnelt daher anderen Web-Frameworks wie „Ruby on Rails“ oder „Django“ für Python. Roo unterstützt den Entwicklungsprozess durch das bereitstellen folgender Funktionalitäten

* eine relationale Datenbank im Backend, angesprochen über die Java Persistence API (JPA),
* Spring Framework und Transaktionsmanagement
* Unterstützung von Aspektorientierung (aspectJ)
* Automatische JUnit-Tests
* Build-Konfiguration durch Maven
* und ein MVC basiertes Frontend, welches JSPs als Views verwendet.

### Hardwaresicht

Das Zielsystem besteht aus einer Client-Serverarchitektur die vorwiegend über den Browser der Endnutzer aufgerufen wird. Die Hardware besteht auf der Server-Seite aus folgenden Komponten:

* Ein J2EE-Anwendungsserver
* Eine relationales-Datenbanksystem

### Logische Sicht

Die Anwendungslogik orientiert sich am Model-View-Controller-Pattern. Die Entitäten werden in der Anwendung durch den Model-Layer wiedergegeben. Benutzerinteraktion wird durch die Views gekapselt bereitgestellt. Diese werden in Form von JSPs bereitgestellt. Während die Programmabläufe in den Controller-Klassen implementiert werden.