Entwurfsbeschreibung

WebWidget

Der Fortgang der wissenschaftlichen Entwicklung ist im Endeffekt eine ständige Flucht vor dem Staunen.
– Albert Einstein

1 Allgemeines

In diesem Dokument wird das Projekt aus den Gesichtspunkten der Objektorientierten Analyse betrachtet. Dieser Entwurf legt den grundsätzlichen Aufbau der Software fest. Während der Implementierung wird es zur Verfeinerung dieses Konzeptes kommen. Um die Änderungen der Entwurfsbeschreibung zu minimieren, sind alle Modellierungsentscheidungen gut zu überlegen und zu begründen.

Wie im Pflichtenheft dargestellt, ist das Projektziel ein intuitives Webportal, welches durch eine Schnittstelle das Implementieren neuer Widgets ermöglicht. Unser Entwurf muss dementsprechend eine allgemeine Definition eines Widgets enthalten und diese in ein Widget-Container-Framework einbetten. Weitere Anforderungen an den Entwurf stellen die zu verwendenden Framworks. GWT bzw. SmartClient fordern eine spezielle Architektur, die bei der Modellierung zu beachten ist.

2 Produktübersicht

Für das Projekt WebWidget wurden 3 Akteure definiert. Der Nutzer, der Serveradministrator und der Widgetentwickler, welche folgende Geschäftsprozesse durchführen:

- Der Nutzer kann eine Startseite aus einer Auswahl von Widgets erstellen und konfigurieren. Die Software stellt ihm den Dienst zur Verfügung, diese zu speichern und zu laden.

- Wie in den Qualitätsanforderungen festgelegt, liegt ein Hauptaugenmerk auf der Erweiterbarkeit der Software durch neue Widgets. Dazu wird dem Widgetentwickler eine Schnittstelle geboten, durch die weitere Widgets implementiert werden können.
- Der Serveradministrator ist für die Verwaltung der Widgets zuständig. Er kann Widgets entfernen oder neue Widgets registrieren.

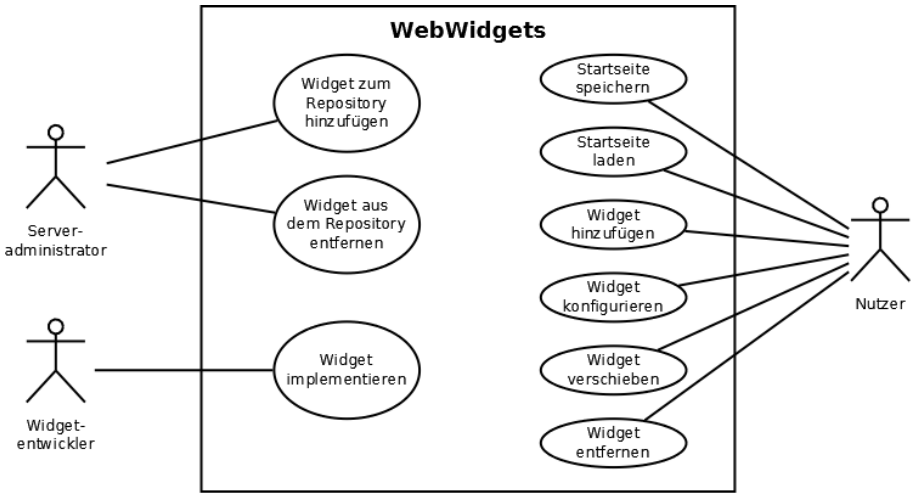

Abbildung 1: Geschäftsprozessdiagramm

3 Grundsätzliche Struktur- und Entwurfsprinzipien für das Gesamtsystem

3.1 Architektur

Das grundlegende Strukturprinzip unserer Anwendung ist eine klassische 3-Schichten-Architektur. Dabei gliedert sich jede Schicht in die Funktionalitäten des Widget-Container-Frameworks und die der konkreten Widgets. Die Serverschicht übernimmt dabei folgende Aufgaben:

1. Sie stellt den HTML- und JavaScript-Code bereit, der schließlich im Browser des Clients angezeigt bzw. ausgeführt wird.

2. Sie führt in einem Servlet den Config-Service aus. Er kann vom Client aus über den GWT-RPC-Mechanismus angesprochen werden und ermöglicht ihm das Laden und Speichern von Startseitenkonfigurationen. Dazu werden diese mittels Hibernate in der Konfigurationsdatenbank persistiert.

3. In einem weiteren Servlet wird der Widget-Repository-Service ausgeführt. Dieser erlaubt dem Client über RPCs den Zugriff auf das Widget-Repository. Bei diesem handelt es sich um einen vom Serveradministrator konfigurierbaren Pool verschiedener für den Nutzer verfügbarer Widget-Typen.

4. Jede Widget-Implementation kann weitere Widget-Services zur Verfügung stellen. Diese können vom Widget über RPCs angesprochen werden, seine Business-Logik ausführen und die Datenbereitstellung übernehmen. Diese Services sind insofern notwendig, als es das SmartClient-Framework seinen Komponenten verbietet, beliebige Anfragen in das Internet zu stellen.

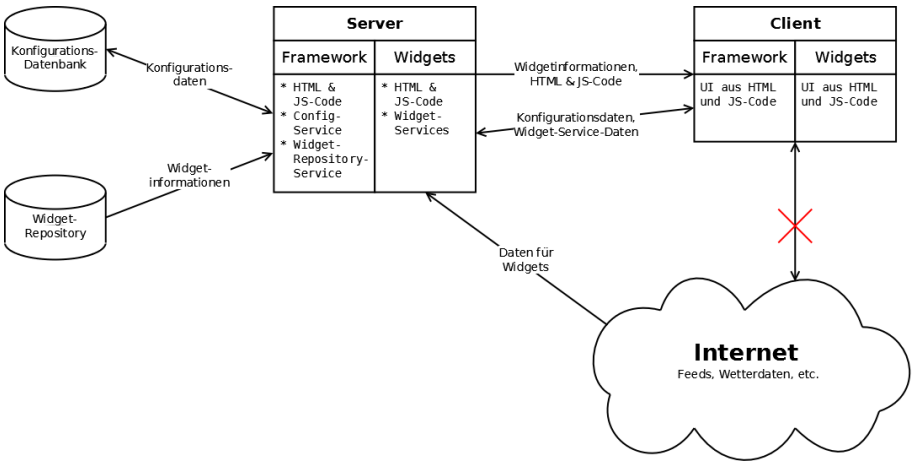

Abbildung 2: Architektur

3.2 Grundlegender Aufbau des Clients

Die clientseitige Präsentationsschicht wird in Java implementiert, vom GWT-Compiler in JavaScript übersetzt und zur Laufzeit vom Server an den Client ausgeliefert. Die wichtigsten Klassen haben dabei folgende Verantwortlichkeiten und tauschen Nachrichten entsprechend des unten stehenden Klassen-Diagramms aus:

- WebWidgetEntryPoint ist der clientseitige Einsprungpunkt in die Anwendung. Erstellt AdministrationPanel und DisplayPanel.

- Das AdministrationPanel zeigt die Buttons zum Laden und Speichern der Startseitenkonfiguration sowie zum Hinzufügen von Widgets.
- Das DisplayPanel zeigt mit Hilfe der WidgetPanel-Klasse die einzelnen Widgets der Startseite und kümmert sich um deren Anordnung sowie die Drag'n'Drop-Funktionalität.
- Die WidgetPanels sind verantwortlich für die Darstellung eines einzelnen Widgets.
- Der WidgetSelector wird geöffnet, wenn der Nutzer ein neues Widget hinzufügen will. Die Liste der verfügbaren Widgets wird vom Widget-Repository-Service erfragt.
- WidgetRepositoryServiceAsync ermöglicht dem WidgetSelector Anfragen an den Widget-Repository-Service zu stellen.
- WidgetRepositoryServiceCallback übergibt dem WidgetSelector die Antworten des Widget-Repository-Services.
- ConfigServiceAsync ermöglich dem AdministrationPanel Anfragen an den Config-Service zu stellen.
- ConfigServiceCallback übergibt dem AdministrationPanel bzw. dem DisplayPanel die Antworten des Config-Services.

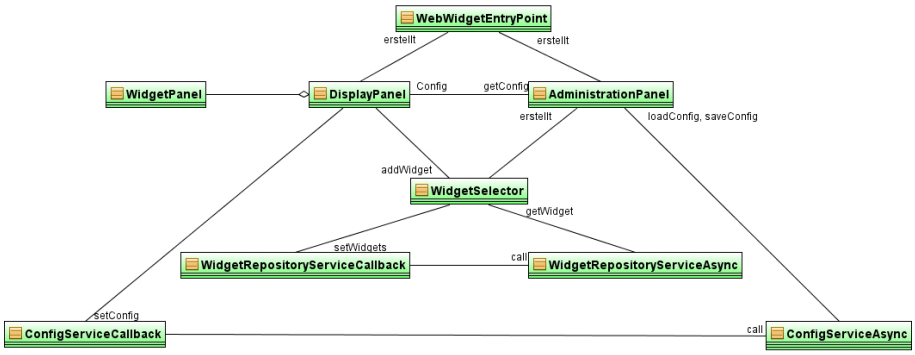

Abbildung 3: Überblick

4 Grundsätzliche Struktur- und Entwurfsprinzipien für die einzelnen Pakete

4.1 Konfigurations-Service

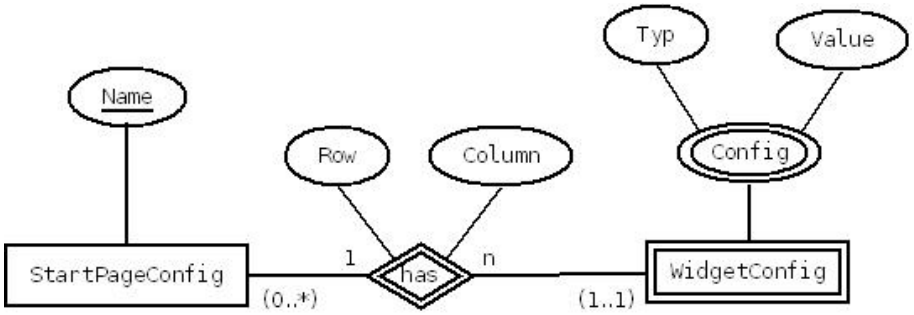

Abbildung 4: ER-Modell

Der Konfigurations-Service setzt sich aus einer serverseitigen und einer clientseitigen Komponente zusammen. Der Client benötigt Objekte aus der Persistenzschicht des Servers. Diese erhält er durch eine Implementation des GWT RPC-Modells. Dadurch wird eine effektive Verantwortlichkeitsteilung erzeugt: Der Server ist für das Abfragen und Speichern von Konfigurationen verantwortlich während der Client für die Konstruktion der entsprechenden Objekte aus den gelieferten Konfigurationsdaten zuständig ist. Dies führt zur Entlastung des Servers und setzt das „Thick-Client"-Prinzip um.

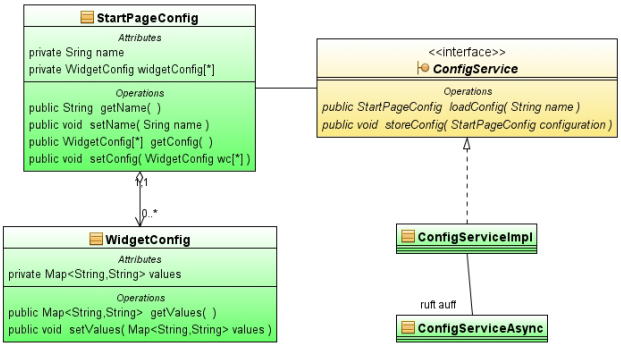

Abbildung 5: Konfiguration

4.2 Widget-Schnittstelle

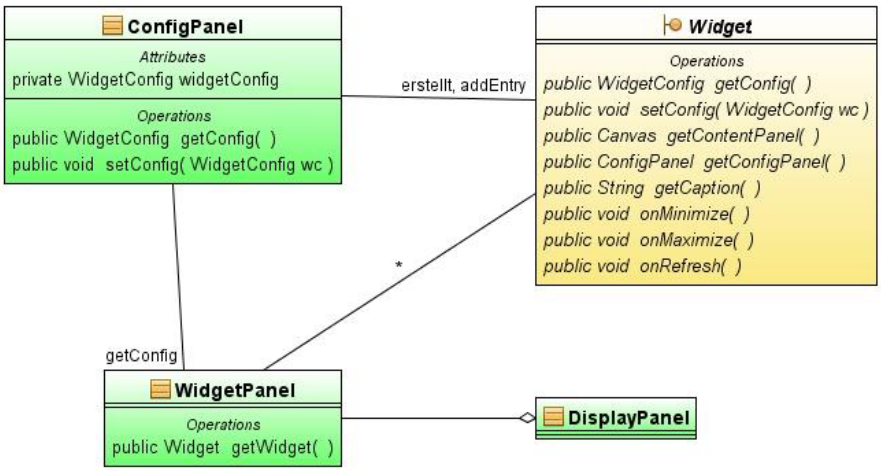

Abbildung 6: Widget

Die Widget-Schnittstelle ist der Ausgangspunkt zur Implementierung aller Widgets und muss dementsprechend folgende Voraussetzungen erfüllen: keine Restriktionen auf den Inhalt des Widgets ausüben und die Möglichkeit bieten, dass jedes Widget individuell auf alle möglichen Aktionen des Benutzers reagieren kann. Dementsprechend gibt das Interface „Widget“ vor, dass ein Widget ein Objekt vom Typ „Canvas“ als Darstellungsbereich zurückgeben muss. Da ein „Canvas“ wiederum andere Objekte vom Typ „Canvas“ enthalten kann, sind kaum Einschränkungen für den Darstellungsbereich gegeben. Zusätzlich muss ein Widget die Methoden getCaption(), onMinimize(), onMaximize() und onRefresh() implementieren. Dadurch wird die zweite Voraussetzung erfüllt.

4.3 Widget-Repository-Service

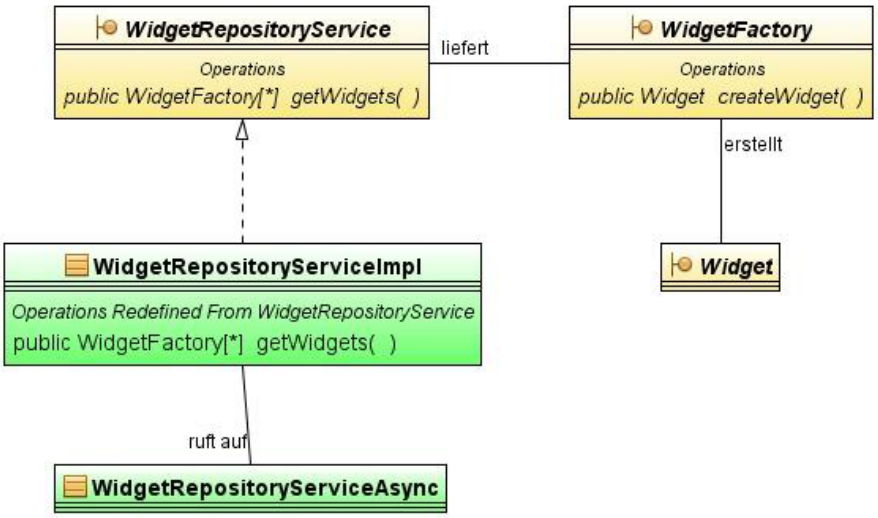

Abbildung 7: Widget-Repository-Service

Der Widget-Repository-Service dient der Abfrage von vorhandenen Widgets, um diese der Oberfläche hinzuzufügen. Um zu verhindern, dass dabei immer Objekte aller verfügabren Widgets erstellt werden müssen, wird ein Factory-Pattern verwendet. Die Klasse WidgetRepositoryService liefert WidgetFactories für die verfügbaren Widgets. Auch hier werden zur Kommunikation zwischen Server und Client GWT RPC verwendet.

4.4 Sequenzdiagramm „Widget hinzufügen“

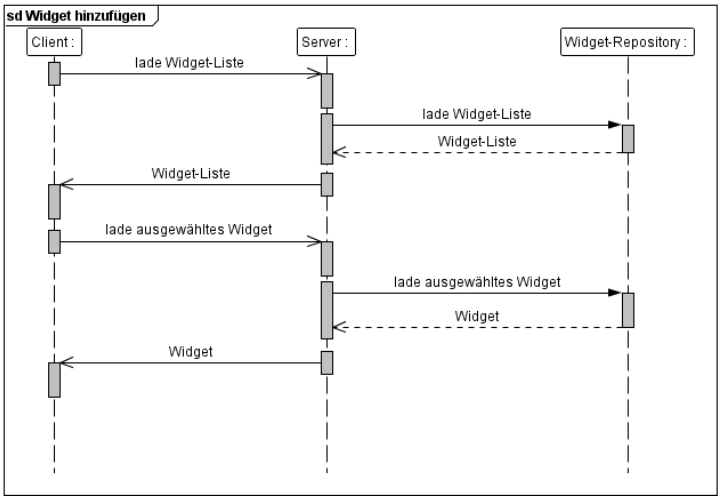

Abbildung 8: Widget hinzufügen

4.5 Sequenzdiagramm „Startseitenkonfiguration speichern“

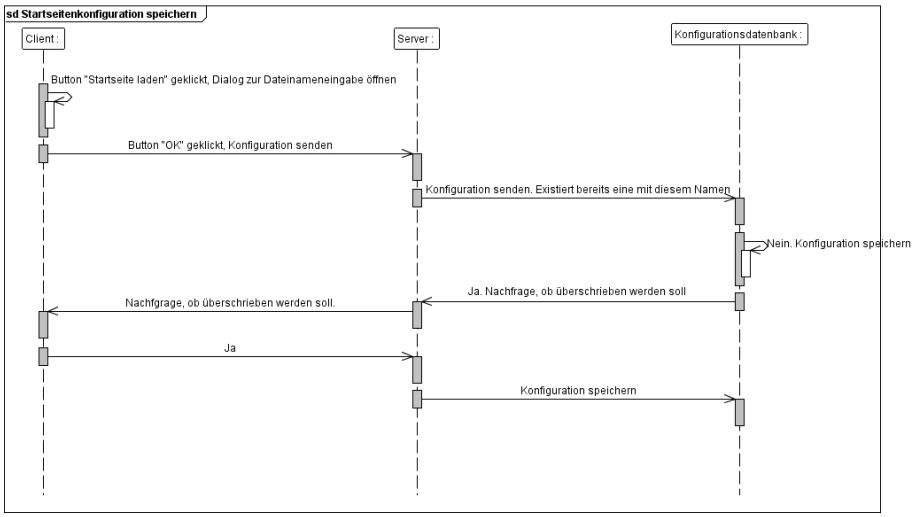

Abbildung 9: Startseitenkonfiguration speichern