Buchungssystem für Gastronomen

Moritz Großmann, Matrikelnummer 01215115

17. Januar 2018

Inhaltsverzeichnis

1 Aufgabenstellung

1.1 Funktionen

1.1.1 Stammdatenverwaltung

Der Benutzer soll alle seine Ressourcen verwalten können. Hierfür soll es eine Stammdatenverwaltung geben, in der er seine Räume, Tisch, Waren und Warengruppen pflegen kann. Tische sollen Räumen untergeordnet sein, sowie Waren den Warengruppen. Außerdem soll der Benutzer bei den Warengruppen eine Baumstruktur anlegen können. So dass es zum Beispiel die Übergruppen "Getränkeünd SSpeißen"gibt und die anderen denen jeweils untergeordnet sind.

Alle Ressourcen können angelegt, geändert und gelöscht werden.

1.2 Design

Das Programm soll für eine Mobile Anwendungen, wie z.B. auf einem Tablet-PC, optimiert sein. Hierfür sind vor allem neben großen Schaltflächen auch eine gute Übersicht nötig. Der Nutzer darf in seinem Handeln nicht dadurch beeinträchtigt werden, dass er sich durch zu kleine Schaltfächen vertippt. Es soll ein dem User vertrautes Design verwendet werden, sodass es keine lange Eingewöhnungszeit gibt.

2 Implementierung

2.1 Teilprojekte

Die Solution ist in 3 Projekte aufgeteilt. Die App, die Domain und das Repository. In der App selbst ist das eigentliche Programm angesiedelt. Hier sind alle Views sowie alle ViewModels implementiert. Außerdem sind hier alle Konverter, welche Daten von einem ViewModel zur View ,und umgekehrt, Konvertieren implementiert. In der Domain werden alle Models sowie das Interface zum Datenbankzugriff deklariert. Im Repository ist der der Datenbankzugriff implementiert. Hier wird abgehandelt, wie Daten gespeichert werden. Durch die 3 Projekte soll eine Struktur entstehen, dass die App nur auf die Domain zugreifen kann. Die App greift auf die Datenbank nur über ein in der Domain deklariertes Interface auf das Repository zu. So wird realisiert, dass das Repository-Projekt ohne Probleme gegen ein anderes Ausgetauscht werden kann, welches das Interface zur Datenpersistierung implementiert.

2.2 Datenspeicherung

Zur Datenspeicherung wird EntityFramework 6.2.0 mit einer SQL-Server Compact Edition genutzt. So ist eine leichte Portabilität gewährleistet. Wird das Programm portiert, und es ist keine Datenbank auf dem neuen System vorhanden, wird eine neue generiert. Der Datenbankzugriff erfolgt ausschließlich über das Interface <IPersistBookingsystemData>. In diesem sind alle Funktionen deklariert, die zum Datenaustausch mit der Datenbank benötigt werden. Implementiert wird dieses Interface im Projekt <Repository> von der Klasse <BookingSystemDataPersistence>. Diese ist als eine Art Database-Helper anzusehen. Die Models, welche in der Domain deklariert sind, werden nicht direkt in der Datenbank gespeichert. Hierzu gibt es für jedes Model ein Datenbank-Pendant. Zum Arbeitern mit der Datenbank werden Models in der <BookingSystemDataPersistenc> zur Speicherung erst in das Datenbankmodel geparst und umgekehrt.

2.3 Benutzerschnittstelle

Die Benutzerschnittstelle wurde mit WPF implementiert. Hierzu wurde das MVVM-Pattern strikt durchgezogen. Einzig im <MainWindow> (einziges Fenster der Applikation) muss einmal Code-Behind verwendet werden, um einen Daten-Kontext herzustellen. Ansonsten werden alle Interaktionen des Benutzers mit der Oberfläche durch ViewModels abgehändelt.

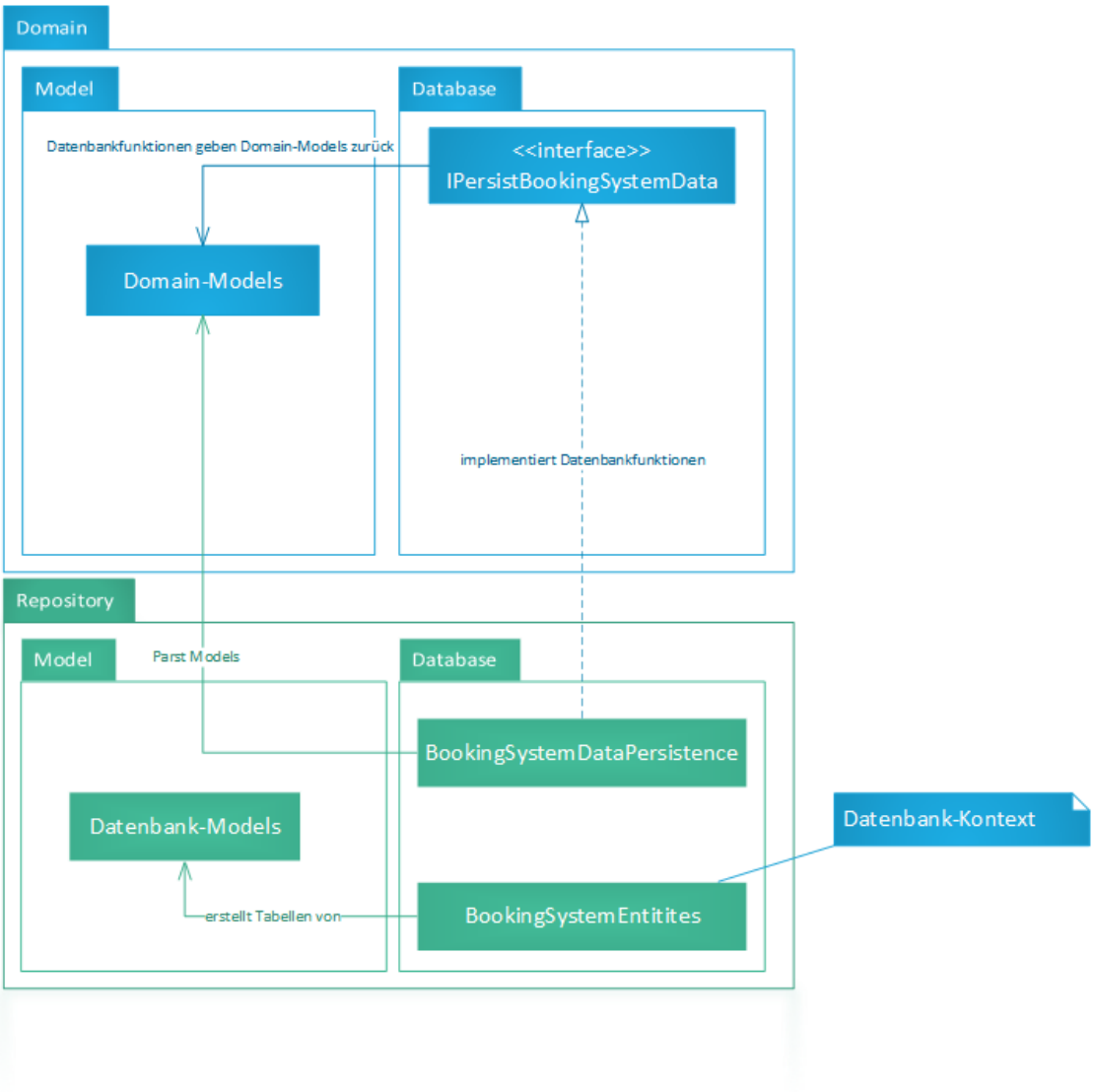

Abbildung 1: Zusammenhang zwischen Domain und Repository

Für das Layout wurde die Library MAHAPP.METRO verwendet. Dadurch wird leicht ein ansprechendes und flaches Design ermöglicht.

Die Anzeige verschiedener Seiten im <MainWindow> wird ausschließlich durch UserControls umgesetzt. Hierfür wird eine Bestimmte UserControl mit einem ViewModel verknüpft. Ändert sich in einem ViewModel eine Property, welche ein anderes ViewModel darstellt, ändert sich auch die View demensprechend. Als Beispiel: Das <MainWindow> ist mit dem <MainViewModel> verknüpft. Das <MainViewModel> besitzt eine Property <CurrentViewModel> vom Typ <BaseViewModel>. Ist das BaseViewModel vom Typ <BaseDataManagementViewModel> wird im dafür vorgesehenen Bereich die <BaseDataManagementView> angezeigt. Dies wird, auch

verschachtelt, in diesem Projekt angewandt. So wird automatisch durch Austausch des ViewModels auch die View geändert.

3 Benutzerhandbuch

3.1 Stammdatenverwaltung

3.2 Die Stammdatenverwaltung aufrufen

3.2.1 Räume bearbeiten

3.2.2 Warengruppen bearbeiten

3.2.3 Waren bearbeiten