Studienarbeit ERP-Programmierung

„Business-Objekt FlightBooking“

Wintersemester 2018/2019

bei Prof. Dr. Martin Thost

von Thomas Zerr

Inhaltsverzeichnis

[Das Business-Objekt FlightBooking 1](#_Toc530590670)

[„Confirm“ 1](#_Toc530590671)

[„Cancel“ 2](#_Toc530590672)

[„CreateFromData“ 2](#_Toc530590673)

[„GetList“ 2](#_Toc530590674)

[Das Business-Objekt „Flight” und ihre Methode „GetList“ 2](#_Toc530590675)

[Architektur der Software 2](#_Toc530590676)

[„LoginWindow“ 3](#_Toc530590677)

[„FlightBookingMainWindowFactory“ 4](#_Toc530590678)

[„FlightBookingMainWindow“ 4](#_Toc530590679)

[„FlightBookingCreateWindowFactory“ 5](#_Toc530590680)

[„FlightBookingCreateViewModel“ 6](#_Toc530590681)

[„FlightBookingEditWindowFactory“ 7](#_Toc530590682)

[„FlightBookingEditViewModel“ 7](#_Toc530590683)

[„FlightBookingFactory“ 7](#_Toc530590684)

[„FlightBooking“ 7](#_Toc530590685)

[„FlightFactory“ 7](#_Toc530590686)

[Problem mit BapiServiceTransactionCommit 7](#_Toc530590687)

[Handbuch 7](#_Toc530590688)

[Login 7](#_Toc530590689)

[Flüge suchen 7](#_Toc530590690)

[Flugbuchung erstellen 7](#_Toc530590691)

[Flugbuchungen suchen 7](#_Toc530590692)

[Flugbuchung bestätigen 7](#_Toc530590693)

[Flugbuchung stornieren 7](#_Toc530590694)

# Das Business-Objekt FlightBooking

Das Business-Objekt FlightBooking wurde für diese Studienarbeit ausgewählt. Es stellt die Buchung für einen Flug dar. Für weitere Informationen siehe in die BAPI Dokumentation dieses Objektes.

## „Confirm“

Durch diese Methode kann eine reservierte Flugbuchung bestätigt werden. Ist die Buchung bereits bestätigt oder storniert ist es nicht möglich sie zu bestätigen.

Für die Implementierung des Aufrufes dieser Methode siehe [„FlightBooking“](FlightBooking#_). Für eine Anleitung, um eine Flugbuchung zu bestätigen, siehe [„Flugbuchung bestätigen“](#_Flugbuchung_bestätigen).

## „Cancel“

Durch diese Methode kann ein reservierte oder bestätigte Flugbuchung storniert werden.

Für die Implementierung des Aufrufes dieser Methode siehe [„FlightBooking“](FlightBooking#_). Für eine Anleitung, um eine Flugbuchung zu stornieren, siehe [„Flugbuchung stornieren“](#_Flugbuchung_stornieren).

## „CreateFromData“

Durch diese Methode kann eine neue Flugbuchung zu einem Flug erstellt werden. Diese neue Flugbuchung kann als Reservierung oder als bestätigte Flugbuchung erstellt werden.

Für die Implementierung des Aufrufes dieser Methode siehe [„FlightBookingFactory“](FlightBookingFactory#_). Für eine Anleitung, um eine Flugbuchung zu erstellen, siehe [„Flugbuchung erstellen“](#_Flugbuchung_erstellen).

## „GetList“

Durch diese Methode kann nach bereits angelegten Flugbuchungen gesucht werden.

Für die Implementierung des Aufrufes dieser Methode siehe [„FlightBookingFactory“](FlightBookingFactory#_). Für eine Anleitung, um nach Flugbuchungen zu suchen, siehe [„Flugbuchung suchen“](#_Flugbuchung_suchen).

# Das Business-Objekt „Flight” und ihre Methode „GetList“

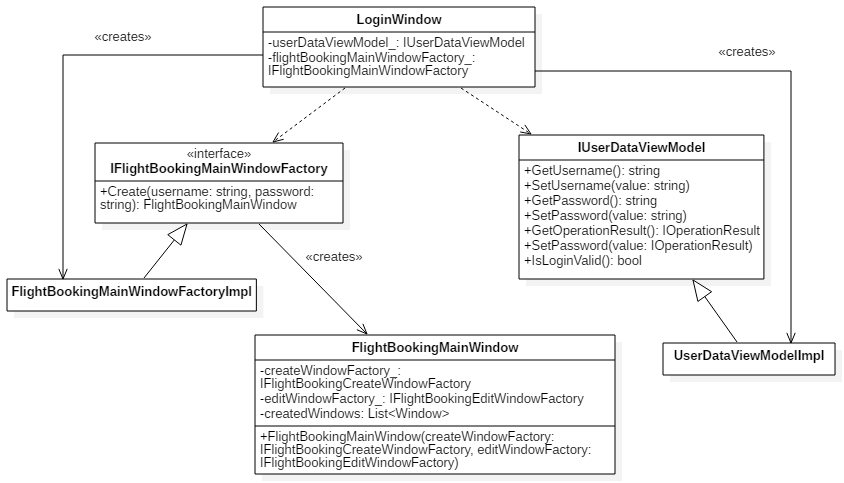
Jede Flugbuchung gehört zu genau einem Flug. Daher muss beim Erstellen einer Flugbuchung den entsprechenden Flug mit angeben. Um die dazu benötigten Informationen des Fluges zu beschaffen, wird die „GetList“-Methode des Business-Objekts „Flight“ verwendet. Da bei dieser Studienarbeit der Fokus auf das Business-Objekt „FlightBooking“ liegt, wurden nicht alle Suchparameter von „GetList“ (des Objektes „Flight“) verwendet.

Für die Implementierung des Aufrufes dieser Methode siehe [„FlightFactory“](#_„FlightFactory“). Für eine Anleitung, um nach Flügen zu suchen, siehe [„Flüge suchen“](#_Flüge_suchen).

# Architektur der Software

Bei dieser Software handelt es sich um eine WPF-Anwendung in C#. Die Software ist nach dem MVVM-Architekturmuster designend. Die wichtigsten Klassen und Vorgänge werden in den anschließenden Unterkapiteln erläutert.

## „LoginWindow“



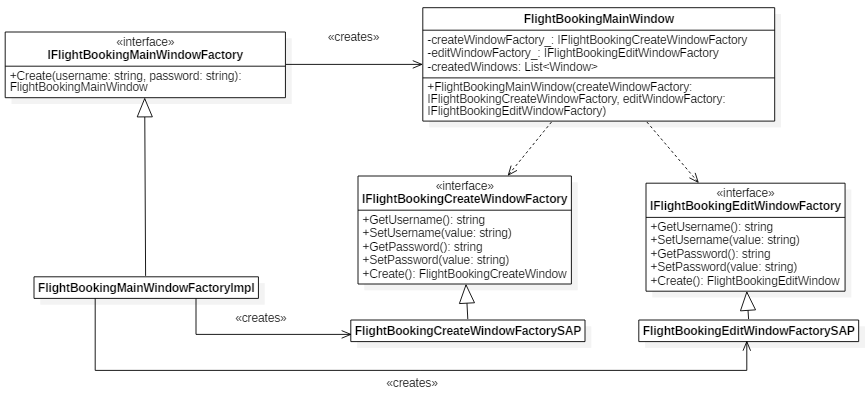
Beim Starten der Anwendung wird das „LoginWindow“ initialisiert. Aufgabe dieses Fensters ist es, die Login-Daten des Benutzers entgegen zu nehmen und den Login-Vorgang zu starten.

Beim Login-Vorgang muss überprüft werden, ob die angegebenen Login-Daten gültig sind. Dafür werden die Login-Daten, in den Eigenschaften „Username“ und „Password“ eines „IUserDataViewModels“ gespeichert und mit der Methode „IsLoginValid“ überprüft. Bei der Implementierung von „UserDataViewModelImpl“ wird lediglich überprüft, ob der „Username“ und das „Password“ nicht leer sind.

Sind die Login-Daten gültig, wird als nächstes ein „FlightBookingMainWindow“ erzeugt. Dazu wird die „Create“-Schnittstelle einer „IFlightBookingMainWindowFactory“ verwendet. Sobald dieses Fenster erzeugt wurde, wird es angezeigt und das „LoginWindow“ schließt sich.

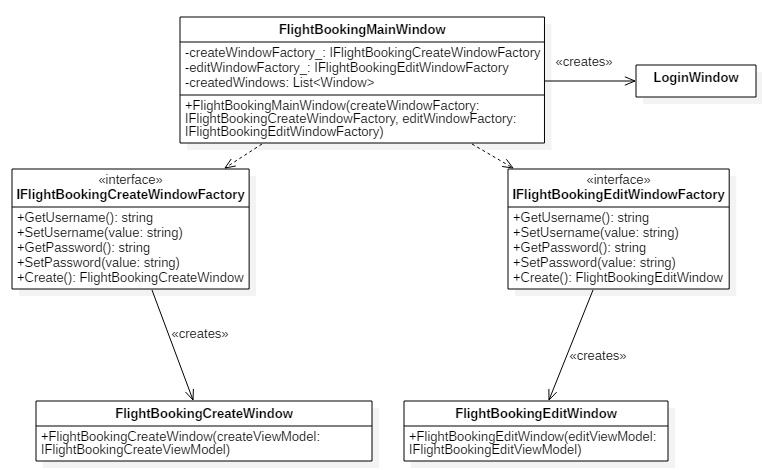
Die konkreten „IFlightBookingMainWindowFactory“ und „IUserDataViewModels“ werden vom „LoginWindow“ selbst erzeugt.

## „FlightBookingMainWindowFactory“



Um ein „FlightBookingMainWindow“ zu erzeugen, muss man es mit den Abhängigkeiten „IFlightBookingCreateWindowFactory“ und „IFlightBookingEditWindowFactory“ versorgen. Dementsprechend muss die „FlightBookingMainWindowFactoryImpl“ diese Abhängigkeiten erzeugen. Die konkreten Klassen dieser Abhängigkeiten werden aktuell über eine Einstellung in der „app.config“-Datei festgelegt. Aktuell ist „SAP“ der einzig gültige Wert für diese Option, dementsprechend können nur die konkreten Klassen „FlightBookingCreateWindowFactorySAP“ und „FlightBookingEditWindowFactorySAP“ instanziiert werden.

## „FlightBookingMainWindow“



Im „FlightBookingMainWindow“ kann der Benutzer drei Vorgänge starten:

1. Logout
2. Ein „FlightBookingCreateWindow“ erstellen und öffnen
3. Ein „FlightBookingEditWindow“ erstellen und öffnen

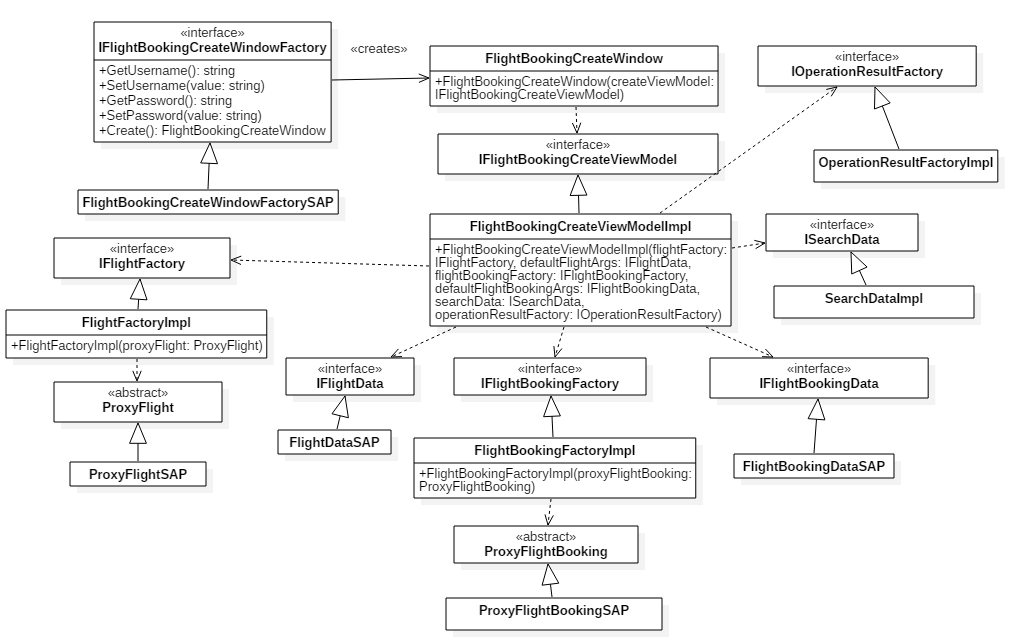
Beim Logout erstellt das „FlightBookingMainWindow“ einen neuen „LoginWindow“ und schließt sich selbst.

Beim zweiten und dritten Vorgang wird über die „Create“-Schnittstelle der jeweiligen Fabrik, dass entsprechende Window-Objekt initialisiert und vom „FlightBookingMainWindow“ geöffnet.

Alle dadurch geöffneten Fenster, werden vom „FlightBookingMainWindow“ in einer Liste verwaltet. Wird „FlightBookingMainWindow“ geschlossen, so werden alle Window-Objekte in dieser Liste ebenfalls geschlossen.

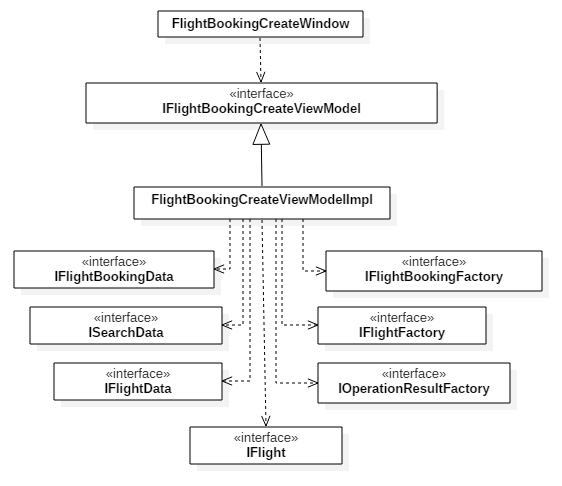
Die Benötigten Objekte der Fabriken, werden dem „FlightBookingMainWindow“ über die Konstruktor Parameter injiziert.

## „FlightBookingCreateWindowFactory“



Um ein „FlightBookingCreateWindow“ zu erzeugen muss man ein „IFlightBookingCreateViewModel“ über den Konstruktor injizieren. Eine Instanz dieses „ViewModels“ wird ebenfalls von dieser Fabrik erzeugt. Die vom „ViewModel“ benötigten Abhängigkeiten werden ebenfalls von der Fabrik erzeugt. Von jeder konkreten Klasse (im Diagramm sichtbar) wird von der „FlightBookingCreateWindowFactorySAP“ eine Instanz erzeugt. Die konkreten Klassen wurde so ausgewählt, damit diese Software mit einem SAP-System über BAPI-Schnittstellen kommuniziert.

## „FlightBookingCreateViewModel“

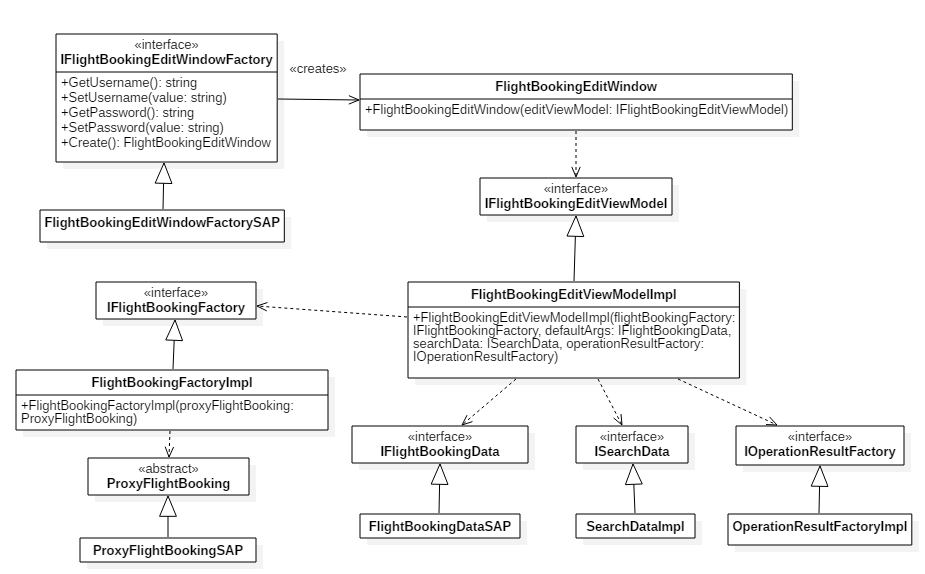


Sämtliche eingaben des Benutzers im „FlightBookingCreateWindow“ sind an Eigenschaften oder Methoden von „IFlightBookingCreateViewModel“ gebunden. Die implementierung „FlightBookingCreateViewModelImpl“ delegiert das Setzen und Abfragen der Eigenschaften an ihre Entsprechende Abhängigkeit. Diese übernehmen dann die Validierung der zu setzenden Werte und werfen eine Exception in Fehlerfälle.

Die „DoFlightSearch“- und „DoCreateFlightBooking“-Methoden delegieren an die Schnittstellen „Retrieve“ von „IFlightFactory“ und an „Create“ von „IFlightBookingFactory“ weiter.

Das „ViewModel“ fängt Exceptions von ihren Abhängigkeiten ab und erstellt dazu eine neue „OperationResult“ und setzt die „OperationResult“-Eigenschaft damit.

## „FlightBookingEditWindowFactory“



Die Factory für das „FlightBookingEditWindow“ verhält sich analog zu dem der „FlightBookingCreateWindowFactory“.

## „FlightBookingEditViewModel“

## „FlightBookingFactory“

## „FlightBooking“

## „FlightFactory“

# Problem mit BapiServiceTransactionCommit

Führt man über eine BAPI-Schnittstelle eine schreibende Methode aus, so werden die Änderungen nicht direkt übernommen und auf die Datenbank geschrieben. Man müsste nach Ausführung der schreibenden Methode die „BapiServiceTransactionCommit“-Methode ausführen. Allerdings wird die „BapiServiceTransactionCommit“-Methode scheinbar nicht ausgeführt, da auch nach Aufruf dieser Methode die Änderungen nicht übernommen sind. Laut BAPI-Dokumentation muss die Antwort der „BapiServiceTransactionCommit“-Methode enthalten, ob sich erfolgreich, fehlerhaft oder ähnliches verlaufen ist, stattdessen ist die Antwort leer. Daher werden die Änderungen durch einen Benutzer vom SAP-System nicht übernommen.

# Handbuch

## Login

## Flüge suchen

## Flugbuchung erstellen

## Flugbuchungen suchen

## Flugbuchung bestätigen

## Flugbuchung stornieren