Projektarbeit M120 / M226B

Leo Haas & Noah Zemljic

Inhalt

[Kurze Beschreibung der Projektidee 2](#_Toc106357055)

[Paket- und Zeitplan 2](#_Toc106357056)

[Grobkonzept I User Stories 3](#_Toc106357057)

[Grobkonzept II: Scribbles 4](#_Toc106357058)

[Scribbles ohne Details 4](#_Toc106357059)

[Variante 1 4](#_Toc106357060)

[Variante 2 5](#_Toc106357061)

[Detaillierte Scribbles 6](#_Toc106357062)

[Verbindungsseite 6](#_Toc106357063)

[Kartenseite 7](#_Toc106357064)

[Detailkonzept II: GUI Detailkonzept User Scenario 8](#_Toc106357065)

[Detailkonzept III: Usability-Test 8](#_Toc106357066)

[Vorüberlegungen 8](#_Toc106357067)

[Usability-Test 8](#_Toc106357068)

[Anwendungsfall 1(Spezifisch) 8](#_Toc106357069)

[Anwendungsfall 2(Spezifisch) 9](#_Toc106357070)

[Anwendungsfall 3(Allgemein) 9](#_Toc106357071)

[Umsetzung II: Einarbeiten MVVM-Pattern 9](#_Toc106357072)

[Leo 9](#_Toc106357073)

[Umsetzung III: Klassendiagramm 10](#_Toc106357074)

# Kurze Beschreibung der Projektidee

In den Modulen M120 und M226B wird als Abschlussarbeit ein Projekt verlangt. Dieses Projekt muss entweder mit JavaFX oder WPF erledigt werden. Vor der eigentlichen Umsetzung des Projekts werden einzelne Zwischenschritte überprüft. In diesem Dokument wird unsere Idee für ein Projekt in User Stories zusammengefasst.

Angelehnt an ein Projekt eines Überbetrieblichen Kurses wollen wir eine SBB Fahrplan App entwickeln. Wir hatten die Idee das gleiche Projektidee mit dem Flugverkehr umzusetzen. Wir fanden dafür aber kein passende API. Deshalb verwenden wir die frei verfügbare API Swiss Public Transport API (<https://transport.opendata.ch/>).

# Paket- und Zeitplan

**Grobkonzept I: User Stories**

Erledigt von: Leo Haas

**Grobkonzept II: Scribbles**

Erledigt von: Noah Zemljic

**Detailkonzept I: Einarbeitung in AdobeXD**

Erledigt von: Leo Haas & Noah Zemljic

**Detailkonzept II: GUI Detailkonzept**

Erledigt von: Noah Zemljc

**Detailkonzept III: Usability Test**

Erledigt von: Leo Haas

**Umsetzung I: View erstellen**

Erledigt von: Leo Haas

**Umsetzung II: Einarbeitung MVVM-Pattern**

Erledigt von: Leo Haas & Noah Zemljic

**Umsetzung III: MVVM-Pattern implementieren**

Erledigt von: Leo Haas & Noah Zemljic

|  |  |  |
| --- | --- | --- |
| **Schritt** | **Sollzeit** | **Istzeit** |
| Grobkonzept I: User Stories | 60min | 30min |
| Grobkonzept II: Scribbles | 30min | 40min |
| Detailkonzept I: Einarbeitung in AdobeXD | 156min | Leo: 60min  Noah: 150min |
| Detailkonzept II: GUI Detailkonzept | 180min | 210min |
| Detailkonzept III: Usability Test | 180min | 90min |
| Umsetzung I: View erstellen | 120min | 180min |
| Umsetzung II: Einarbeitung MVVM-Pattern | 240min | Leo: 180min  Noah:150min |
| Umsetzung III: MVVM-Pattern implementieren | 300min | Leo: 300min  Noah: 300min |

# Grobkonzept I User Stories

|  |  |
| --- | --- |
| **Titel:** Verbindung suchen | |
| Als Benutzer möchte ich die nächsten 4 Verbindungen zwischen Start- und Endstation angezeigt bekommen. | |
| **Priorität: A**  **Sollzeit: 2h**  **Istzeit: 3h** | **Abnahmekriterien:**  -Ich kann eine Startstation eingeben  -Ich kann eine Endstation eingeben  -Vier Verbindungen zwischen Startstation und Endstation werden angezeigt  -Wenn keine Start-/Endstation gefunden wird, wird eine Fehlermeldung angezeigt und man kann die Station neu eingeben  -Wenn keine Verbindungen gefunden werden, wird eine Fehlermeldung angezeigt |
|  | |
| **Titel:** Stationssuche | |
| Als Benutzer möchte ich eine Station suchen können. | |
| **Priorität: B**  **Sollzeit: 1h**  **Istzeit: 30min** | **Abnahmekriterien:**  -Ich kann den Stationsnamen eingeben  -Die Korrekte Station wird gefunden  -Der Stationsname wird angezeigt  -Wenn Keine Station gefunden wurde, wird eine Fehlermeldung angezeigt |
|  | |
| **Titel:** Abfahrtstafel | |
| Als Benutzer möchte von einer Station alle ausgehenden Verbindungen angezeigt bekommen. | |
| **Priorität: B**  **Sollzeit: 1h**  **Istzeit: 30min** | **Abnahmekriterien:**  -Alle ausgehenden Verbindungen werden im Tabellenformat angezeigt  -Wenn keine Verbindungen gefunden werden, wird eine Fehlermeldung angezeigt |
|  | |
| **Titel:** Suchvorschläge | |
| Als Benutzer möchte ich während dem Eintippen einer Station Vorschläge angezeigt bekommen. | |
| **Priorität: B**  **Sollzeit: 1h**  **Istzeit: 1h** | **Abnahmekriterien:**  -Während dem Tippen werden übereinstimmende Vorschläge angezeigt  -Diese Vorschläge können ausgewählt werden  -Der Text wird vervollständigt |
|  | |
| **Titel**: Zukünftige Verbindungen | |
| Als Benutzer möchte ich ein Datum und eine Uhrzeit eingeben können, um zukünftige Verbindungen zu suchen. | |
| **Priorität: C**  **Sollzeit: 30min**  **Istzeit:**  **15min** | **Abnahmekriterien:**  -Ich kann ein Datum per Datumauswahl eingeben  -Ich kann eine Uhrzeit per Uhrzeitauswahl eingeben  -Die zukünftigen Verbindungen werden korrekt angezeigt |
|  | |
| **Titel:** Stationsanzeige auf Karte | |
| Als Benutzer möchte ich auf einer Karte sehen, wo sich die Station befindet. | |
| **Priorität: D**  **Sollzeit: 1h**  **Istzeit: Nicht implementiert** | **Abnahmekriterien:**  -Die Station wird korrekt auf der Karte angezeigt  -Die Station wird per Pin markiert |
|  | |
| **Titel:** Stationen in der Nähe | |
| Als Benutzer möchte ich auf einer Karte die nächstgelegene Station angezeigt bekommen. | |
| **Priorität: D**  **Sollzeit: 1h**  **Istzeit: Nicht implementiert** | **Abnahmekriterien:**  -Die Stationen in der Nähe werden korrekt auf der Karte angezeigt  -Die Stationen werden per Pin markiert |

# Grobkonzept II: Scribbles

## Scribbles ohne Details

### Variante 1

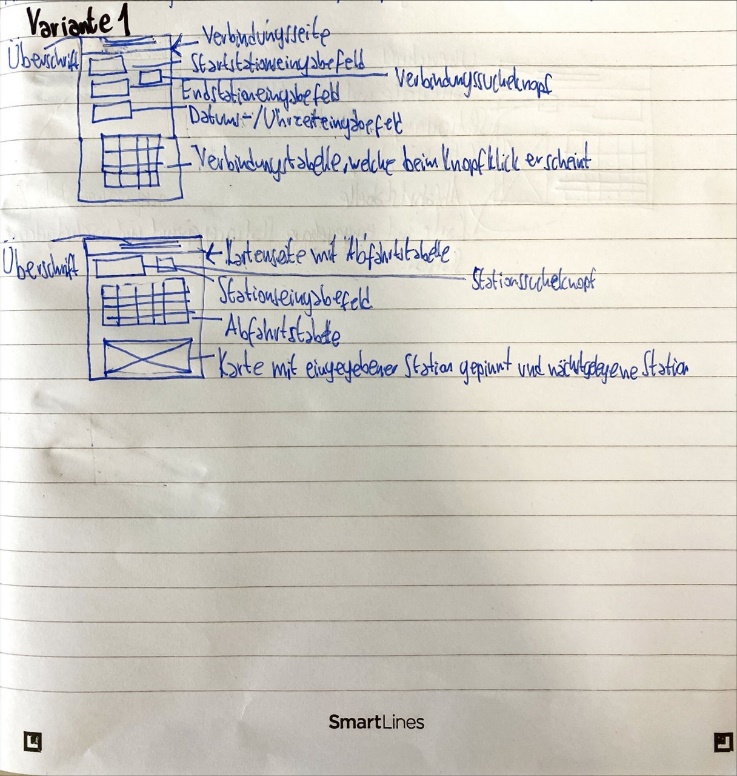


Abbildung 1 Scribbles Variante 1, Erstellt von: Noah Zemljic

### Variante 2

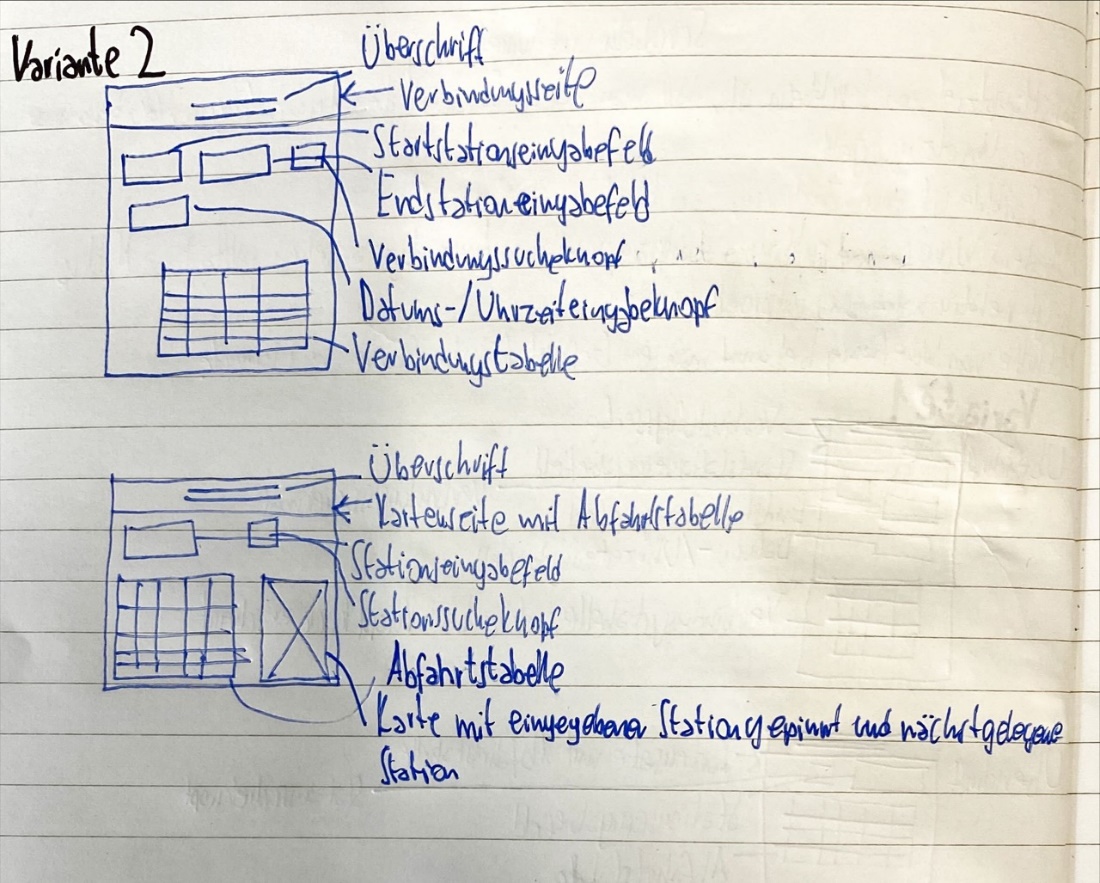


Abbildung 2 Scribbles Variante 2, Erstellt von: Noah Zemljic

## Detaillierte Scribbles

### Verbindungsseite

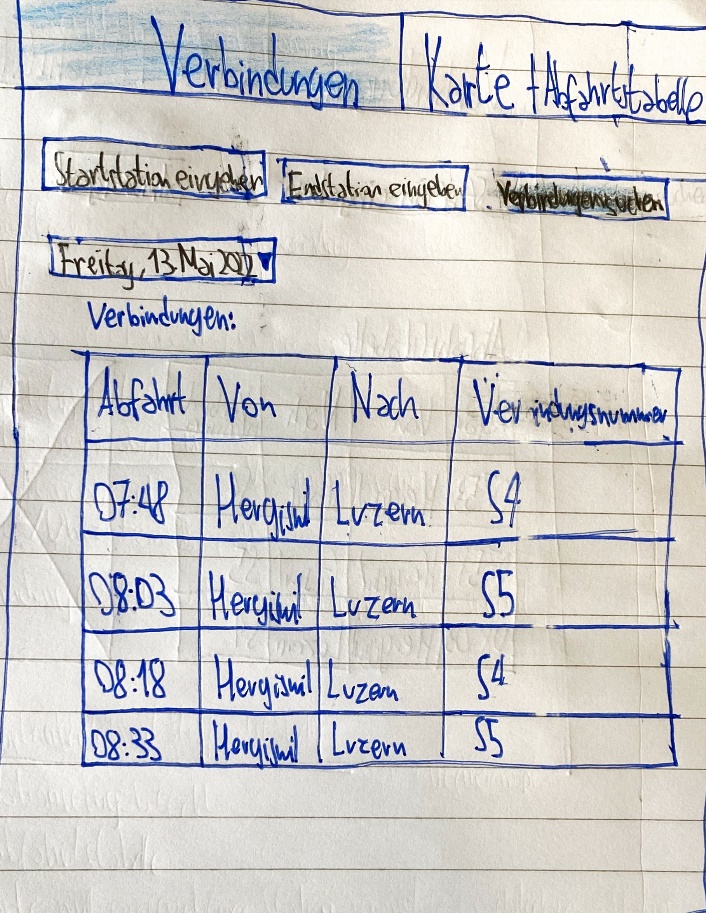
**Aktiver Bereich in Navigation markiert**Der Benutzer weiss, wo er sich befindet.

Abbildung 3 Detaillierte Scribbles Verbindungen, Erstellt von: Noah Zemljic

**Stationseingabefelder**

Momentan mit einem Platzhalter / Placeholder

(Vielleicht ein Label zu jedes Eingabefeld  
hinzufügen?)

**Verbindungssuche-Knopf**  
Beim Klicken erscheint eine Verbindungstabelle  
(Vielleicht ein Versteckknopf hinzufügen?)

**Datums-/Uhrzeitauswahl**Es können auch zukünftige Verbindungen  
angezeigt werden, in dem man ein  
zukünftiges Datum / zukünftige  
Uhrzeit auswählt.

**Verbindungsüberschrift**

**Verbindungstabelle**

### Kartenseite

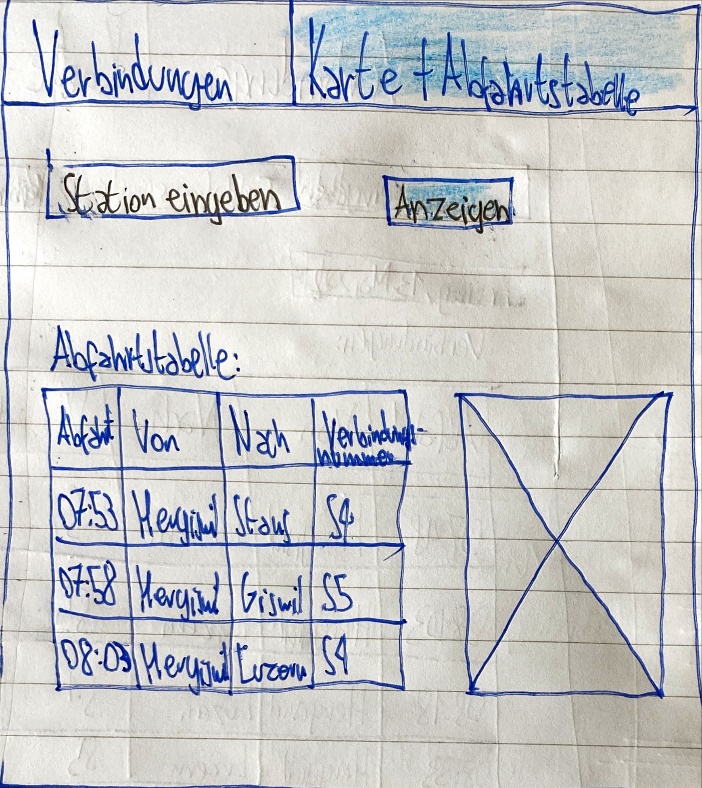
**Aktiver Bereich in Navigation markiert**

Abbildung 4 Detaillierte Scribbles Karte und Abfahrtstabelle, Erstellt von: Noah Zemljic

**Stationseingabefeld**Momentan mit einem Platzhalter / Placeholder  
(Vielleicht ein Label zu jedes Eingabefeld  
hinzufügen?)

**Karten- und Abfahrtstabellenanzeigeknopf  
  
Abfahrtstabellenüberschrift**

**Abfahrtstabelle  
  
Karte mit eingegebener Station und  
nächstgelegene Station**

# Detailkonzept II: GUI Detailkonzept User Scenario

Thomas, 25 Jahre alt, befindet sich in Hergiswil und möchte die nächstgelegene ÖV-Station finden. Er sucht nach Stationen in seiner Nähe. Er findet den Bahnhof Hergiswil, NW. Er schaut auf der Anzeigetabelle von Hergiswil, NW, ob bald ein Zug Richtung Luzern fährt. Da die Verbindung zu früh fährt, sucht er nach Verbindungen zwischen Hergiswil, NW und Luzern um 07:40.

# Detailkonzept III: Usability-Test

## Vorüberlegungen

Frage à Vermutung

1. Gibt der Benutzer die Daten, um eine Verbindung zu suchen, korrekt und in der richtigen Reihenfolge ein?

à Einige Benutzer werden die Daten in anderer Reihenfolge eingeben wie geplant. Im Prototyp ist dies nicht ohne Weiteres möglich. Im fertigen Produkt können die Daten in jeder Reihenfolge eingegeben werden, solange alle Felder ausgefüllt sind, bevor der Button «Verbindungen suchen» geklickt wird.

1. Können die Benutzer die Abfahrtstafel einer Station finden und verstehen Sie den Unterschied der Verbindungen auf der Abfahrtstafel von den Verbindungen zwischen zwei gewählten Stationen?

à Die Abfahrtstafel ist gleich aufgebaut, wie die Abfahrtstafel an einer Station. So sollte die Funktion für alle Benutzer offensichtlich und verständlich sein. Die Information ist nicht auf der Startseite und muss deshalb zuerst gefunden werden. Durch den Link in der Tab-Leiste sollte die Route aber verständlich sein.

1. Erkennen die Benutzer den Unterschied zwischen «Station suchen» und Station in meiner Nähe suchen?

à Der Benutzer erkennt den Unterschied zwischen den Funktionen der beiden Knöpfen.

## Usability-Test

### Anwendungsfall 1(Spezifisch)

**Betroffenes User Scenario:**

|  |  |
| --- | --- |
| **Titel:** Verbindung suchen | |
| Als Benutzer möchte ich die nächsten 4 Verbindungen zwischen Start- und Endstation angezeigt bekommen. | |
| **Priorität: A**  **Sollzeit: 2h**  **Istzeit:** | **Abnahmekriterien:**  -Ich kann eine Startstation eingeben  -Ich kann eine Endstation eingeben  -Vier Verbindungen zwischen Startstation und Endstation werden angezeigt  -Wenn keine Start-/Endstation gefunden wird, wird eine Fehlermeldung angezeigt und man kann die Station neu eingeben  -Wenn keine Verbindungen gefunden werden, wird eine Fehlermeldung angezeigt |

**Aufgabe:** Finden Sie die nächsten Verbindungen von Hergiswil, NW nach Luzern am 22. Mai 2022 um 07.40.

### Anwendungsfall 2(Spezifisch)

**Betroffenes User-Scenarios:**

|  |  |
| --- | --- |
| **Titel:** Stationssuche | |
| Als Benutzer möchte ich eine Station suchen können. | |
| **Priorität: B**  **Sollzeit: 1h**  **Istzeit:** | **Abnahmekriterien:**  -Ich kann den Stationsnamen eingeben  -Die Korrekte Station wird gefunden  -Der Stationsname wird angezeigt  -Wenn Keine Station gefunden wurde, wird eine Fehlermeldung angezeigt |
| **Titel:** Abfahrtstafel | |
| Als Benutzer möchte von einer Station alle ausgehenden Verbindungen angezeigt bekommen. | |
| **Priorität: B**  **Sollzeit: 1h**  **Istzeit:** | **Abnahmekriterien:**  -Alle ausgehenden Verbindungen werden im Tabellenformat angezeigt  -Wenn keine Verbindungen gefunden werden, wird eine Fehlermeldung angezeigt |

**Aufgabe:** Suchen die die Abfahrtstafel für die Station Hergiswil, NW.

### Anwendungsfall 3(Allgemein)

**Betroffenes User Scenario:**

|  |  |
| --- | --- |
| **Titel:** Stationen in der Nähe | |
| Als Benutzer möchte ich auf einer Karte die nächstgelegene Station angezeigt bekommen. | |
| **Priorität: D**  **Sollzeit: 1h**  **Istzeit:** | **Abnahmekriterien:**  -Die Stationen in der Nähe werden korrekt auf der Karte angezeigt  -Die Stationen werden per Pin markiert |

**Aufgabe:** Sie wollen den Öffentlichen Verkehr nutzen, kennen sich aber an ihrem jetzigen Standort nicht aus. Suchen Sie nach Informationen, die Ihnen weiterhelfen könnten.

# Umsetzung II: Einarbeiten MVVM-Pattern

## Leo

Verwendetes Tutorial:

<https://www.tutorialspoint.com/mvvm/index.htm>

Das erstellte Beispiel ist im GitHub-Repo des Projekts hinterlegt.

Die Tutorials sind in Englisch, was für mich kein Problem war.

Das Tutorial erklärt zu Beginn, was das MVVM-Pattern ist und welche Vor- und Nachteile es hat. Danach kommt gleich ein praktisches Beispiel. Das finde ich sehr gut, weil ich Dinge besser mit einem Code-Beispiel verstehe. Diese Beispielaufgabe habe ich nachgebaut und im GitHub-Repo hinterlegt. Ich habe dann versucht das Pattern direkt in unserem Projekt zu implementieren. Damit hatte ich aber Mühe. Vor allem habe ich Commands noch nicht verstanden. Dabei hat mir das folgende Tutorial geholfen:

[WPF MVVM Tutorial](https://www.youtube.com/playlist?list=PLA8ZIAm2I03hS41Fy4vFpRw8AdYNBXmNm)

Mit dieser Videoreihe konnte ich das MVVM-Pattern für die erste View implementieren. Ich verstand ich durch die zusätzlichen Erklärungen im Video die Commands besser. Ausserdem hat die Videoreihe mir sehr beim Strukturieren meines Codes geholfen. Also wo welche Funktionen hingehören.

Ich habe vor allem die Videos zum ViewModel und den Commands genau durchgeschaut. Durch die Videos habe ich auch PropertyChanged, also das Melden von Veränderungen an die View, verstanden.

Keines der beiden Tutorials hat mir dabei geholfen Daten an einen DataGrid zu binden. Dafür habe ich folgende Anleitung verwendet:

<https://wpf-tutorial.com/de/89/das-datagrid-steuerelement/datagrid-spalten/>

Von dieser Erklärung zum DataGrid konnte ich mir an das DataBinding mit Listboxen herleiten.

# Umsetzung III: Klassendiagramm

Das Klassendiagramm befindet sich im doc-Ordner im Github-Repo als PDF und als Visio-Dokument.

In der ViewModelBase wurde das Interface INofityPropertyChanged implementiert. Beide ViewModels ConnectionsViewModel und StationboardViewModel erben von der ViewModelBase.

Die View ConnectionsView befindet sich in einem DataContext mit dem ConnectionsViewModel und stellt Properties mit DataBinding dar. StationboardView befindet sich in einem DataContext mit dem StationboardViewModel.

Die CommandBase implementiert das Interface ICommand. Die Methode Execute ist abstract und muss von jedem Command implementiert werden. Die Commands GetConnectionsCommand und GetStationboardCommand erben von der Commandbase.

Das ConnectionsViewModel steht in einer bidirektionalen Assoziationsbeziehung mit dem GetConnectionsCommand. Die StationboardViewModel mit dem GetStationboardCommand.

Die ViewModels stehen auch in Assoziationsbeziehungen mit den Models. Das ViewModel ConnectionsViewModel verfügt über Properties mit den Typen Connections und Stations. Das StationboardViewModel über Properties mit dem Typen Stationboard und Stations.

Das Model Connection hat ein gerichtete Assoziationsbeziehung mit ConnectionPoint.

ConnectionPoint hat eine gerichtete Assoziationsbeziehung mit Station.

Connections hat eine gerichtete Assoziationsbeziehung mit Connection. Es hat eine ObservableCollection von Connection.

Station hat eine gerichtete Assoziationsbeziehung mit Coordinate.

Stationboard hat eine gerichtete Assoziationsbeziehung mit StationboardEntry und Station.

StationboardEntry hat eine gerichtete Assoziationsbeziehung mit Stop.

Stations hat eine gerichtete Assoziationsbeziehung mit Station. Es hat eine ObservableCollection von Station.