Miniprojekt – Teil XAML/WPF

Für Teilnehmer des .MsTe -Moduls:
Für Teilnehmer des .NET-Moduls steht eine alternative Aufgabenstellung zur Verfügung, welche auf dem dort implementierten Service Stack für Auto-Reservationen aufbaut (vgl. separate Aufgabenstellung).

Neben den Vorlesungen und Übungsaufgaben ist auch ein Miniprojekt Teil von Mobile and GUI Engineering (MGE). Das Projekt gliedert sich – wie auch die Vorlesung – in einen Android-Teil und einen WPF-Teil. Ziel des Projekts ist es, das Gelernte selbständig auf ein neues Szenario anwenden zu können.

Miniprojekt bedeutet, dass das Projekt über mehrere Wochen läuft und von Ihnen selbständig, in den Übungslektionen oder im Selbststudium, bearbeitet wird. Sie dürfen das Projekt auch mit einem Partner oder in einer Gruppe von maximal drei Personen durchführen (Teams mit Mitgliedern aus mehreren Studiengängen wären toll). Tragen Sie die Gruppen bitte hier ein: https://docs.google.com/spreadsheets/d/1elvtQfkUvOriI8bsM-_Ut4EkPsTDIRi8Q9wVpqfZNRY/edit#gid=173728802 (Sheet „Gruppeneinteilung - Teil WPF“)

Die beiden Teile des Miniprojekts werden bewertet, aber nicht benotet. Nur wenn Sie beide bestandenen haben sind Sie zur Prüfung zugelassen! Achten Sie also darauf, die Bewertungskriterien einzuhalten und sprechen Sie bei Unklarheiten mit Ihrem Übungsbetreuer.

Bewertungskriterien

Die folgenden Bewertungskriterien dienen als Richtlinien für die Bewertung. Um das Projekt zu bestehen, müssen nicht alle Punkte vollumfänglich erfüllt sein, bei zu grossen Mängeln wird die Prüfungszulassung aber nicht erteilt:

Szenarios	Können die vorgegebenen (oder selbst definierten) Szenarios durchgespielt werden?
Design	Entspricht das User Interface den besprochenen Prinzipien?
Interaktion	Behandelt die App Fehler und gibt dazu Feedback? Ist die Nutzerführung verständlich?
Code	Ist der Code sauber strukturiert (einheitlicher Stil, keine TODOs, Dokumentation nicht-trivialer Stellen)? Ist der Code in einem Repository eingecheckt? Sind Tests vorhanden?

Die Bewertung erfolgt während einer kurzen Demonstration der App in Ihrer Übungslektion in der Semesterwoche 14 (also der letzten XAML/WPF-Woche).

Aufgabenstellung Gadgeothek-Admin-App

Sie kennen vielleicht die Gadgeothek der HSR-Bibliothek in welcher Sie verschiedene Gadgets ausleihen können. Im WPF-Miniprojekt werden Sie dazu eine einfache Admin-App entwickeln.

Eigene Aufgabenstellungen

Alternativ können Sie auch ein eigenes Thema bearbeiten. Schreiben Sie dazu eine eigene Aufgabenstellung, definieren Sie Szenarios die Sie umsetzen werden und senden Sie sie an martin.seelhofer@hsr.ch. Sie werden innert nützlicher Frist einen go/no go Entscheid erhalten.

In Umfang und Komplexität sollte die eigene Aufgabe in etwa der Gadgeothek-Admin-App entsprechen (Domainmodell mit >3 Klassen, Data Binding, Listen, Datenbankanbindung [kann auch In-Memory sein]) und muss mit der Windows Presentation Foundation (WPF) in C# und XAML entwickelt werden. Auf Anfrage – und eigenes Risiko – können auch UWP Apps oder Xamarin Apps, die mit XAML und C# geschrieben werden, bewilligt werden.

Szenarios/User Stories der Gadgeothek Admin App

Szenarios helfen Ihnen zu verstehen, in welchem Kontext die Software verwendet wird. Die untenstehenden Szenarien beschreiben, was ihre App am Ende des Miniprojektes unterstützen sollte. Verwenden Sie sie zum Testen ihrer App und schauen Sie darauf, dass Sie möglichst alle davon optimal unterstützen können.

Szenario/Story Gadgets anzeigen

Als Administrator möchte ich alle verfügbaren Gadgets angezeigt bekommen. Ich bin zufrieden, wenn die Gadgets mit allen Eigenschaften in einer Liste/Tabelle angezeigt werden können und ich diese nach den Eigenschaften/Spalten sortieren kann.

Szenario/Story Gadget hinzufügen

Als Administrator möchte ich ein neues Gadget hinzufügen können. Ich bin zufrieden, wenn ich in einem Fenster ein neues Gadget mit all seinen Eigenschaften erfassen kann und dieses anschliessend in die Liste der Gadgets aufgenommen wird.

Szenario/Story Gadget editieren

Als Administrator möchte ich ein bestehendes Gadget anpassen können. Ich bin zufrieden, wenn ich das zu bearbeitende Gadget in einem Fenster mit all seinen Eigenschaften bearbeiten kann und die Änderungen anschliessend gespeichert werden können.

Szenario/Story Gadgets entfernen

Als Administrator möchte ich ein Gadget entfernen können. Ich bin zufrieden, wenn ich ein Gadget in der Liste/Tabelle markieren und dieses nach Bestätigen einer Sicherheitsrückfrage löschen kann.

Szenario/Story Kunden anzeigen

Als Administrator möchte ich alle verfügbaren Kunden angezeigt bekommen. Ich bin zufrieden, wenn die Kunden mit allen Eigenschaften in einer Liste/Tabelle angezeigt werden können und ich diese nach den Eigenschaften/Spalten sortieren kann.

Szenario/Story Kunden hinzufügen

Als Administrator möchte ich einen neuen Kunden hinzufügen können. Ich bin zufrieden, wenn ich in einem Fenster einen neuen Kunden mit all seinen Eigenschaften erfassen kann und dieser anschliessend in die Liste der Kunden aufgenommen wird.

Szenario/Story Kunden editieren

Als Administrator möchte ich einen bestehenden Kunden anpassen können. Ich bin zufrieden, wenn ich den zu bearbeitenden Kunden in einem Fenster mit all seinen Eigenschaften bearbeiten kann und die Änderungen anschliessend gespeichert werden können.

Szenario/Story Kunden entfernen

Als Administrator möchte ich einen Kunden entfernen können. Ich bin zufrieden, wenn ich einen Kunden in der Liste/Tabelle markieren und diesen nach Bestätigen einer Sicherheitsrückfrage löschen kann.

Szenario/Story Ausleihvorgang starten

Als Administrator möchte ich einen Ausleihvorgang starten können. Ich bin zufrieden, wenn ich in einem Fenster einen neuen Ausleihvorgang mit den nötigen Eckdaten starten kann und dieser anschliessend gespeichert wird.

Szenario/Story Ausleihvorgänge überwachen

Als Administrator möchte ich die Ausleihvorgänge anzeigen können. Ich bin zufrieden, wenn ich diese in einer chronologisch sortierten Liste/Tabelle ansehen kann, welche sich selbst in regelmässigen Zeitabständen von einigen Sekunden aktualisiert (Optional: Aktualisierung in Echtzeit).

Szenario/Story Ausleihvorgang beenden

Als Administrator möchte ich einen Ausleihvorgang beenden können. Ich bin zufrieden, wenn ich in der Liste/Tabelle einen Ausleihvorgang auswählen und diesen nach Bestätigung einer Sicherheitsrückfrage beenden kann.

Installation

In der Anfangsphase des Projekts wird ein grosser Teil der Domain und der Businesslogik der Gadgeothek-Admin-App zur Verfügung gestellt. Setzen sie bei ihrer Lösung diesen Code ein! Die nachfolgenden Klassendiagramme bietet ihnen eine Übersicht über den vorgebebenen Code. Sollten sie das Bedürfnis haben diesen Code grundlegend zu modifizieren, so dürfen sie dies gerne tun. Der Code steht in Form einer vorbereiteten Visual Studio Solution auf GitHub zur Verfügung.

Der Serverteil besteht aus einer einfachen Node.js-Anwendung die eine REST-Schnittstelle anbietet. Sie finden den Code dazu unter github.com/HSR-MGE/Miniprojekt-Server. Starten Sie den Server lokal. Dazu müssen Sie Node.js installiert haben. Kopieren Sie dann die package.json und server.js Dateien in ein Verzeichnis (bzw. klonen Sie einfach das Repository). In diesem Verzeichnis installieren Sie dann mit npm (das ist der Node-Packagemanager) die benötigten Libraries und starten den Server:

```
$ npm install
...
$ node server.js
```

Der Server hört auf Port 8080 (HTTP) und gibt Debug-Output aus.

Ihren WPF-Client müssen Sie anschliessend natürlich jeweils auf denselben Server/Port verbinden. Versuchen Sie in Ihrer Gruppe auch, über das Netzwerk auf den Rechner der anderen Gruppenmitglieder zuzugreifen und die Admin-App über das Netzwerk laufen zu lassen.

Vorlagen

Die Kommunikation mit dem Server erfolgt über zwei vorgegebene Service-Klassen (in C#). Diese implementieren zwei einfache Clients für eine REST-Schnittstelle, je einmal zur Nutzung als Admin und als Client. Das Hauptaugenmerk lag bei der Entwicklung auf dem Admin-Teil, um für das Miniprojekt eine möglichst passende Ausgangslage zur Verfügung zu stellen.

Die Service-Klassen erwarten im Konstruktor die Web-Adresse (URL), an welcher der Server zu finden ist. Tragen Sie die entsprechende Adresse dazu z.B. als App-Setting in der App.config Ihres WPF-Projekts ein (alternativ können Sie eine ComboBox mit allen verfügbaren Adressen in Ihr User Interface einbauen).

Für den Zugriff auf eine lokal laufende Instanz lautet die entsprechende Zeile im Abschnitt `<appSettings>` somit:

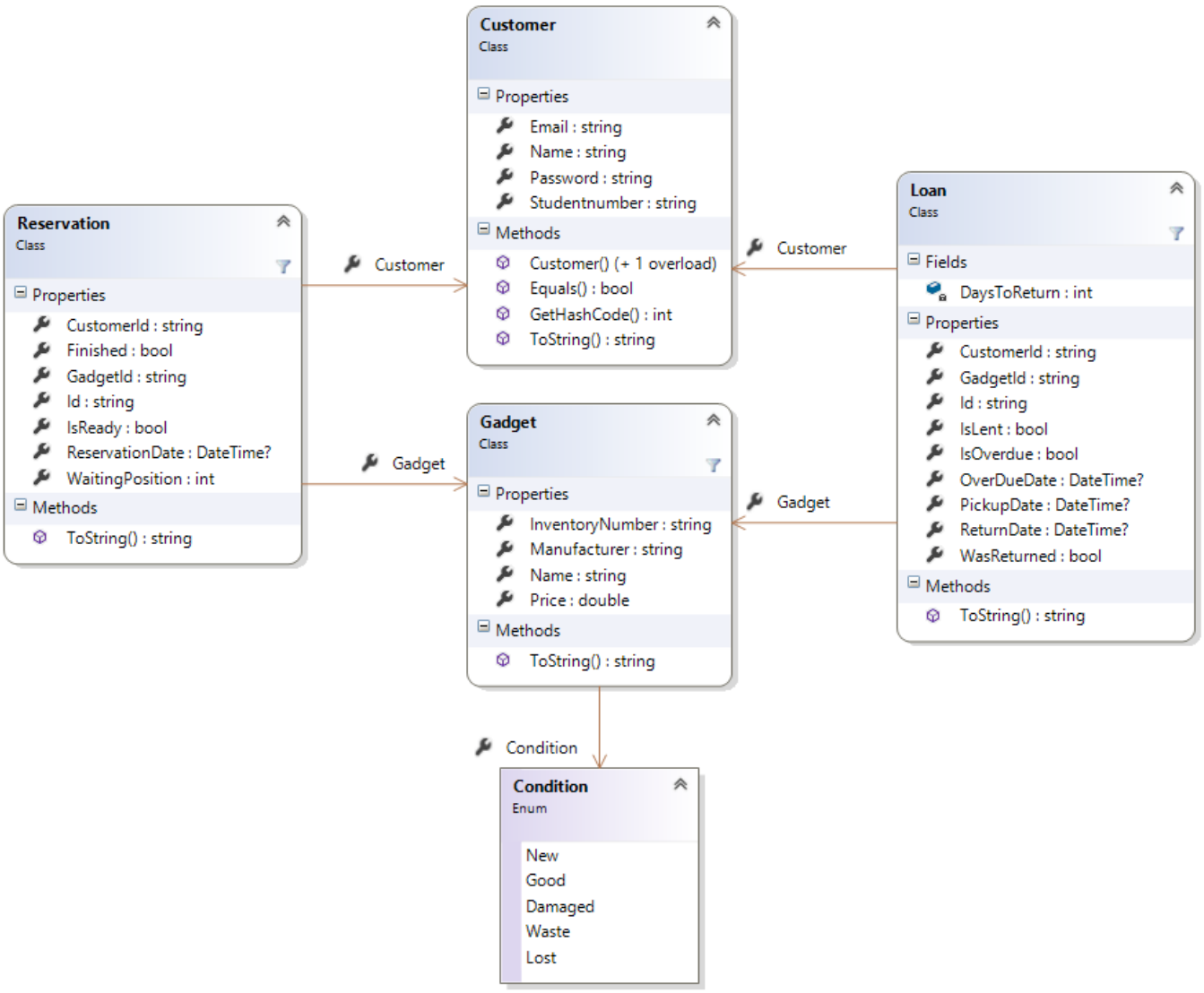
```
<add key="server" value= "http://localhost:8080" />
```

Weitere Hinweise zur Nutzung der Klassen finden Sie in den (vielen) XML-Kommentaren des C#-Quellcodes der Service-Klassen, sowie den beiden Konsolen-Apps, welche in der Projektvorlage zu finden sind.

In der Vorlage finden Sie zudem das Domain-Package, welches die vier Klassen `Gadget` (entspricht einem konkreten Gerät), `Customer` (entspricht einem Kunden), `Loan` (eine Ausleihe eines Gerätes) und `Reservation` enthält. Die Klassen sollten hoffentlich selbsterklärend sein, bei Fragen zögern Sie aber bitte nicht diese in den Übungen zu stellen.

Achtung:
Der Einfachheit können Sie die Aufrufe der Service-Klassen synchron tätigen, d.h. wenn diese Methoden direkt aus dem User Interface aufgerufen werden, blockiert das User Interface kurz. Normalerweise ist dies nicht so schlimm, da ein Desktop-Rechner i.d.R. über eine schnelle Internetverbindung verfügt. Trotzdem sollten Sie darüber nachdenken, die Aufrufe in einem Background-Thread (asynchron) auszuführen.

Klassendiagramm des Domain-Packages



Klassendiagramm des Service-Packages