Assessment

Um im Kontext von SlideLizard zu bleiben sollen 2 sehr rudimentäre Applikation erstellt werden, welche beide es ermöglichen Präsentationstermine von Vortragenden zu verwalten.

Grundsätzlich soll damit festgestellt werden ob ein Verständnis für verteilte Applikationen und Webservices, sowie Frontend Development besteht. Es werden Technologien von uns vorgegeben, welche auch im praktischen Einsatz bei uns eingesetzt werden. Falls diese Technologien noch nicht bekannt sind, wird ein oberflächliches Einarbeiten nötig sein.

Wesentlich ist aber eine saubere Code-Strukturierung und nicht Prototypen-hafte Entwicklung. Neben der Funktionalität wird auch dies überprüft.

1. Aufgabenstellung I

Die 1. Aufgabenstellung beschäftigt sich mit einer kleinen verteilten Verwaltungsapplikation.

Aufgeteilt wird das ganze in 2 Applikationen, einer Backend Applikation die Daten zurückliefert und einer Frontend Applikation, die die Daten anzeigt.

1.1. Backend Applikation

Technologie: ASP.NET CORE 6

Die Applikation stellt einen REST-basierten WebService zur Verfügung der neue Präsentationstermine entgegen nimmt und bestehende Präsentationstermine zurückgibt. Jeder Präsentationsname darf **NUR 1 mal existieren**, der Webservice soll also einen Fehler zurückgegeben falls versucht wird einen schon bestehenden Präsentationsnamen erneut zu senden. Es gibt keine Authentifizierung/Authorizierung oder Userverwaltung, sprich darum soll sich nicht gekümmert werden.

Folgende Schnittstellen sollen existieren:

1. GET /api/presentation
    a. Eine Liste an Präsentaitonen wird zurückgegeben.
    b. Returns: [{ name, fromdate, todate, location }]
2. GET /api/presentation/statistic?fromdate={fromdate}&todate={todate}
    a. Die Anzahl an Präsentationen im jeweiligen Zeitraum wird zurückgegeben
    b. Returns: { count }
3. POST /api/presentation
    a. Eine neue Präsentation wird erstellt
    b. BODY: { name, fromdate, todate, location }

WICHTIG: Es muss keine Datenbank oder ähnliches persistentes Speichermedium verwendet werden dafür. Die Daten können in-memory bleiben. Der Webservice soll einfach lokal ausgeführt werden ohne Deployment auf einen Server.

1.2. Frontend Applikation

Technologie: Blazor (https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor)

Diese Web-Applikation soll nun die Daten des in 1.1 definierten Webservices anzeigen und einpflegen. Das Design/Aussehen spielt dabei keine Rolle.

Es soll 3 Seiten geben:

1. Startseite – Auflistung aller Präsentationstermine (Abholung vom Webservice)
    a. Kann z.b. in einer kleinen Tabelle gemacht werden oder in einem Kalender etc. ganz nach Belieben
2. Unterseite – Anlegen von Präsenttaionsterminen (Senden an Webserivce)
    a. Ein Formular mit den im Webservice definierten Felder
    b. Nach erfolgreichem Abschicken des Formulars soll die aktualisierte Startseite (mit dem neuen bestehenden Eintrag) angezeigt werden.
3. Unterseite – Abfrage der “Präsentationsstatistik”
    a. Ein Formular mit zwei Eingabefeldern (from/to)
    b. Zeigt vom Webserivce die entsprechende Anzahl an Präsentationen im Zeitraum an

2. Aufgabenstellung II

Die 2. Aufgabenstellung die es zu lösen gilt, ist eine Desktop Applikation, welche es ermöglicht Präsentationstermine von Vortragenden “zu verwalten”.

Es ist nicht notwendig die komplette Applikation zu erstellen, ein großteil davon wurde schon vorgegeben. Es müssen jedoch einige Elemente noch ausprogrammiert werden, bei denen derzeit einfach statische Daten zurückgegeben werden.

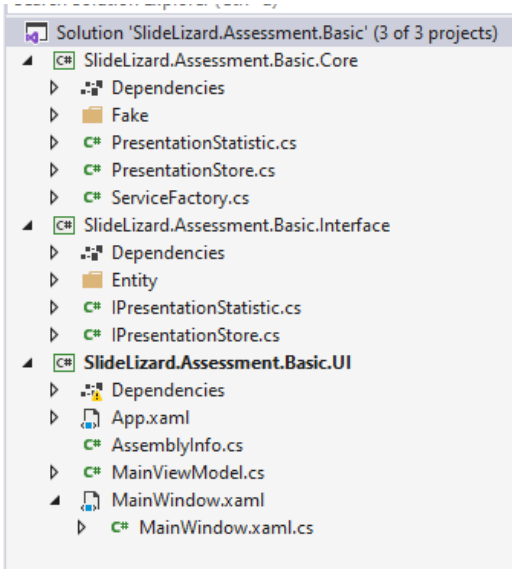
Wichtig! Dies ist lediglich ein Beispiel und sollte in der Praxis zumindest teilweise nicht 1:1 so umgesetzt werden, des dient nur zur Veranschaulichung von Grundkenntnissen in den wesentlichen Bereichen.

1.3. Einrichtung

- Das Template für die Applikation läuft auf .NET 6
- Außerdem wird ein Visual Studio 2022 (zumindest die gratis Community Edition) benötigt
    - Alternativ kann auch VS 2019 verwendet werden, dann muss auf .NET 5 umgestellt werden
- Die Warnings können ignoriert werden

Nach dem öffnen der .sln Datei sollten nun 3 Projekte sichtbar sein.

- SlideLizard.Assessment.Basic.Core
- SlideLizard.Assessment.Basic.Interface
- SlideLizard.Assessment.Basic.UI



SlideLizard.Assessment.Basic.UI

Dieses Projekt muss nicht verändert werden! Es muss aber als Start-Projekt gesetzt sein (fett hinterlegt). Wenn dies nicht der Fall ist (Rechts klicken auf das Projekt -> Set as StartUp Project)

Hier ist das UserInterface (also die Desktop Applikation) enthalten. Es greift die Daten in .Core ab und zeigt diese entsprechend dem User an.

SlideLizard.Assessment.Basic.Interface

In Diesem Projekt sind lediglich die Interfaces definiert auf denen die UI aufbaut. Diese Interfaces selbst müssen auch nicht verändert werden, sowie auch nicht die Presentation-Klasse selbst zur Datenhaltung. Sie beinhalten in Kommentaren die erforderlichen Implementierungsdetails der einzelnen auszuprogrammierenden Methoden.

SlideLizard.Assessment.Basic.Core

Hier werden die Im .Interface definierten Schnittstellen implementiert, lediglich hier muss für die Aufgabenstelle Code verändert bzw. erzeugt werden.

Die beiden Schnittstellen wurden einmal als “Fake” implementiert, dies sind Implementierungen die schlichtweg statische Daten zurückgeben (also keine Funktionalität erfüllen) rein um zu sehen wie die UI mit entsprechenden Daten aussehen würde.

Die zwei Klassen **PresentationStatistic** und **PresentationStore** müssen ausprogrammiert werden!

Um zwischen den statischen Fake-Daten und den realen Implementierungen zu wechseln gibt es die Klasse ServiceFactory, welche eine Konstante `FAKE_SERVICES` zu Beginn definiert, die von true auf false gesetzt werden muss um die “echten” Implementierungen von der UI verwenden zu lassen.

Die Applikation sollte sich ohne Veränderungen starten lassen. Einmal *Rebuild* dann *Start*.