Projektaufgabe INF2020: C# und .NET - „Wonderlist“

Das aufstrebende Jungunternehmen „7 Wonderkinder UG (haftungsbeschränkt)“ beauftragt Sie mit der Entwicklung des Prototyps einer Software zur Abbildung des Geschäftsmodells.

Der Markt für TODO-Listen-Apps ist hart umkämpft und alteingesessene Platzhirsche wie Apple und Microsoft erschweren es jungen Start-Ups enorm.

Nach anfänglichen Erfolgen beauftragt „7 Wonderkinder UG (haftungsbeschränkt)“ den IT-Dienstleister „Give me Power GmbH“ mit der Erstellung einer TODO-Listen-App. Leider investiert die „Give me Power GmbH“ das Budget vor allem in Latte Macchiato und Röhrenjeans, so dass zum Ende des Projekts lediglich eine Datenbank (SQLite-Format) vorliegt.

Geschockt von diesem Geschäftsgebaren und finanziell schwer angeschlagen ersucht „7 Wonderkinder UG (haftungsbeschränkt)“ den Star-Investor „Frank Gehlen“ (hatte mal eine Statistenrolle in einer Fernsehsendung) um Hilfe, der den Tipp gibt, man solle mal ein paar Studierende beauftragen, die wären günstig und damit wäre er immer gut gefahren.

Um mit diesem wertvollen Tipp neu durchstarten zu können, beauftragt „7 Wonderkinder UG (haftungsbeschränkt)“ Sie mit der Entwicklung einer neuen TODO-Listen-App. Da aus dem Vorprojekt gelernt wurde, wie teuer die Erstellung einer Datenbank sein kann, soll Ihre App kompatibel zur vorhandenen Datenbank sein.

Lesen Sie zuerst die allgemeinen Tipps zur Lösung der Projektaufgabe im Anhang!

Aufgabenstellung (75 Punkte)

Hinweis: Für diese Aufgabenstellung können Sie 75 Punkte erhalten. Zusammen mit den maximal 25 erreichbaren Punkten aus der Bearbeitung der Aufgabenblätter ergibt sich eine Gesamtpunktzahl von 100 maximal möglichen Punkten.

Entwickeln Sie für das Unternehmen eine moderne Anwendung, welche kompatibel zur bestehenden Datenbank ist.

Die Realisierung erfolgt mit C# und WPF. Die Kommunikation zwischen Client und Server kann wahlweise per WCF (ggf. CoreWCF) oder gRPC erfolgen.

Die Verwendung von MVVMC, dem Nuget-Paketmanager, MSTest, Autofac und Fluent NHibernate mit Repository-Pattern wird vorausgesetzt.

Zielplattform ist .NET 6 oder wahlweise das .NET Framework 4.8.

Es werden nachfolgend nur die zu realisierenden Funktionalitäten beschrieben. Bei der eigentlichen Umsetzung werden den Studierenden große Freiheiten eingeräumt. Dies soll kreative Eigenleistungen fördern und deren Bewertung ermöglichen.
Dies bezieht sich sowohl auf die Architektur der Anwendung als auch auf das Design der Oberflächen und das Bedienkonzept der Anwendung.

Die bestehende Datenbank soll zur Entwicklung der neuen Software analysiert und genutzt werden. Hierfür stehen im Internet verschiedene Open-Source-Programme zur Verwaltung von SQLite-Datenbanken zur Verfügung (z.B. SQLite Studio).

Das vorliegende Datenbankschema darf nicht verändert werden.

Bitte beachten Sie:

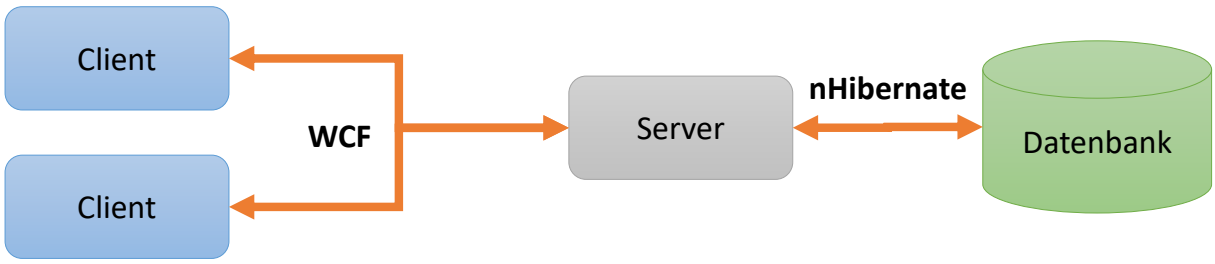
- Auftretende Programmabstürze führen zu Punktabzug!
- Fehlende Eingabevalidierungen führen ebenfalls zu Punktabzug, auch wenn das Programm nicht abstürzt.
- Lässt sich das abgegebene Projekt nicht fehlerfrei kompilieren und starten, führt dies ebenfalls zu Punktabzug!
- Fehlermeldungen müssen jederzeit klar formuliert, für den Anwender verständlich und situationsbezogen sein. Ansonsten führt dies zu Punktabzug!

Von den Studierenden wird die Fähigkeit zur Recherche im Internet mittels geeigneter Suchmaschinen erwartet.

Hinweis: *Das Punktesystem ist analog zur Bewertung von Studienarbeiten*

Client-Server-System (15 Punkte):

Nachfolgend wird der grundlegende Aufbau beschrieben, den Sie umsetzen sollen.
(Grafik zeigt Variante mit WCF)



Server:

Realisieren Sie den Server auf Basis einer Anwendung, die einen Dienst zur Kommunikation bereitstellt (Bei WCF: Konsolenanwendung).

Verwenden Sie Autofac als Dependency Injection Container.

Implementieren Sie die Kommunikation mit der Datenbank mittels Fluent-nHibernate. Setzen Sie hierzu das Konzept des „Optimistic Locking“ um.

Ermöglichen Sie den gleichzeitigen Zugriff von mehr als einer Client-Anwendung zur selben Zeit (Tipp: Testen Sie das).

Client:

Realisieren Sie den Client auf Basis einer WPF-Anwendung, die Sie mit dem MVVMC-Entwurfsmuster aufbauen.

Der Client darf nicht direkt mit der Datenbank kommunizieren. Der Client darf ausschließlich mittels WCF/gRPC mit dem Server kommunizieren.

Verwenden Sie Autofac als Dependency Injection Container.

Der Client dient sowohl zum Management der Aufträge als auch zur Administration des Systems.

<u>Navigationsmöglichkeit:</u>
Das Softwaresystem muss über eine Navigationsmöglichkeit zum Öffnen der nachgehend beschriebenen Module verfügen.
Realisieren Sie hierzu eine Menüführung, die die Auswahl der Module ermöglicht.

<u>Listenverwaltung und Listeneinträge:</u>
Der Haupteinsatzzweck des zu entwickelnden Softwaresystems ist die Verwaltung von TODO-Listen und deren Einträge. Diese Funktionalität des Softwaresystems soll daher möglichst leicht und verständlich und möglichst prominent im Softwaresystem zur Verfügung gestellt werden.

Enthalten sollen Möglichkeiten zum Erstellen, Bearbeiten und Löschen von Listen und Listeneinträgen sein. Die verschiedenen Listen sollen über eine Auflistung auswählbar sein. Listeneinträge sollen aufgelistet werden. Pro Listeneintrag sollen weitere

Informationen angebbar sein. Finden Sie eine innovative Möglichkeit, um dies benutzerfreundlich umzusetzen.

Kategorieverwaltung:
Es soll eine Möglichkeit zur Verwaltung von Kategorien geben, mit denen die Listeneinträge verknüpft werden können. Die Funktionalität zur Verwaltung der Kategorien soll weniger prominent, aber dennoch leicht verständlich, in der Anwendung enthalten sein. Entwickeln Sie hierzu eine Idee, wie Sie die Funktionalität dezent in die Anwendung integrieren.

Erscheinungsbild der Oberfläche:
Das Design soll sich an modernen Windows-Anwendungen orientieren und für Anwender ohne Expertenwissen intuitiv bedienbar sein.
Die Anwendung soll optisch ansprechend gestaltet werden. Hierbei ist auf die integrierten Funktionalitäten für Theming und Styling von WPF zurückzugreifen.

Um die ansprechende Gestaltung der Oberfläche zu unterstützen, finden Sie im Internet auch frei verfügbare Icon-Bibliotheken zur Gestaltung der Oberfläche (z.B. Schaltflächen, etc).

Erscheinungsbild der Oberflächen:

Das Design soll sich an modernen Windows-Anwendungen orientieren und für Anwender ohne Expertenwissen intuitiv bedienbar sein.
Die Anwendung soll optisch ansprechend gestaltet werden. Hierbei ist auf die integrierten Funktionalitäten für Theming und Styling von WPF zurückzugreifen.

Um die ansprechende Gestaltung der Oberfläche zu unterstützen, finden Sie im Internet auch frei verfügbare Icon-Bibliotheken zur Gestaltung der Oberfläche (z.B. Schaltflächen, etc).

Kategorien (10 Punkte):

Realisieren Sie folgende Funktionalitäten zur Abbildung der Verwaltung von Kategorien. Möglichkeiten zur Anzeige, Neuanlage, zum Speichern, Bearbeiten und Löschen von Kategorien.

Kategorien müssen sich durch einen eindeutigen Namen (ohne Beachtung von Groß- und Kleinschreibung, also wäre folgendes **nicht** möglich: „Einkauf“, „einkauf“) unterscheiden.

Eine Kategorie darf nicht gelöscht werden, solange es noch Listeneinträge gibt, die ihr zugeordnet sind.

Dem Anwender soll dieser Sachverhalt in einer verständlichen Meldung (oder einem anderen benutzerfreundlichen Vorgehen) verdeutlicht werden.

Kategorien zeichnen sich durch folgende Eigenschaften aus:

- Name (Name | Bezeichnung der Kategorie)

Vergessen Sie nicht die Abbildung des Optimistic Lockings!

Tabelle: Categories

Schreiben Sie 5 relevante Unit-Tests für diese Funktionalität.

Listen (10 Punkte):

Realisieren Sie folgende Funktionalitäten zur Abbildung der Verwaltung von Listen.

Schaffen Sie Möglichkeiten zur Anzeige, Neuanlage, zum Speichern, Bearbeiten und Löschen von TODO-Listen.

TODO-Listen zeichnen sich durch folgende Eigenschaften aus:

- Beschreibung (Description | Bezeichnung der Liste)
- Bemerkung (Comment | optionale Anmerkungen zur Liste)

Werden Listen gelöscht, zu denen es noch Listeneinträge gibt, sollen diese automatisch mitgelöscht werden.

Vergessen Sie nicht die Abbildung des Optimistic Lockings!

Tabelle: Tasklists

Schreiben Sie 5 relevante Unit-Tests für diese Funktionalität.

Listeneinträge (15 Punkte)

Realisieren Sie folgende Funktionalitäten zur Abbildung der Verwaltung der Listeneinträge.

Schaffen Sie Möglichkeiten zur Anzeige, Neuanlage, zum Speichern, Bearbeiten und Löschen von Listeneinträgen.

Listeneinträge zeichnen sich durch folgende Eigenschaften aus:

- Beschreibung (Description | Bezeichnung des Listeneintrags)
- Bemerkung (Comment | optionale Anmerkungen zum Listeneintrag)
- Status (State | ist der Eintrag erledigt oder nicht)
- Anlagedatum (CreationDate | Zeitpunkt, zu dem der Listeneintrag angelegt wurde)
- Fälligkeitsdatum (DueDate | optionaler Zeitpunkt, bis zu dem der Listeneintrag abgearbeitet sein muss.)
- Erinnerung (ReminderMinutes | Anzahl Minuten, zu welcher eine Erinnerung vor dem Fälligkeitsdatum erfolgen soll, -1 bedeutet „keine Erinnerung“)
- Priorität (Priority | Priorität der einzelnen Listeneinträge in einer Liste, Integer. Je höher, desto wichtiger)
- Listeneinträge sollen optional mit mehreren Kategorien verknüpft werden können.

Der Anwender soll möglichst einfach und intuitiv Listeneinträge zu Listen erfassen können. Listeneinträge können nie ohne Liste existieren.

Es soll eine einfache Möglichkeit geben, die Beschreibung der Listeneinträge zu verwalten.
Ebenso soll es einfach möglich sein, den Status der Listeneinträge (CheckBox) zu verändern.

Das Anlagedatum der Listeneinträge soll automatisch bei der Anlage gesetzt werden.

Die Listeneinträge sollen in einer Listendarstellung angezeigt werden. Listeneinträge sollen auswählbar sein. Für den ausgewählten Listeneintrag soll eine Detailansicht eingeblendet werden, auf der die weiteren Eigenschaften verwaltet werden können.
Optional soll ein Fälligkeitsdatum für einen Listeneintrag angegeben werden können. (DatePicker). Es soll möglich sein, ein gesetztes Fälligkeitsdatum wieder zu entfernen.

Wird ein Fälligkeitsdatum gesetzt, soll es möglich sein, einen Erinnerungszeitpunkt (relativ zum Fälligkeitsdatum) festzulegen (ComboBox):
Wird das Fälligkeitsdatum entfernt, soll auch der Erinnerungszeitpunkt entfernt werden.

Folgende Werte sollen auswählbar sein:

- Ereigniszeitpunkt
- 5 Minuten vorher
- 10 Minuten vorher
- 1 Stunde vorher
- 12 Stunden vorher
- 1 Tag vorher
- 2 Tage vorher

Die Werte müssen für die Datenhaltung umgerechnet werden.
Für jeden Listeneintrag soll eine Priorität angegeben werden können (ComboBox).

Folgende Werte sollen auswählbar sein:
- 25%
- 50%
- 75%
- 100%

Die Werte müssen für die Datenhaltung umgerechnet werden.

Beim Erstellen eines Listeneintrags soll die Priorität automatisch mit 50% vorbelegt werden.

Der optionale Kommentar zu einem Listeneintrag soll mehrzeilig eingebbar sein.

Die Verknüpfung zu Kategorien soll über eine eigene Listenansicht im Detailbereich der Listeneinträge realisiert werden.
Zum Hinzufügen der Verknüpfung sollen die vorhandenen Kategorien zur Auswahl angezeigt werden. Es darf pro Kategorie nur eine Verknüpfung zu einem Listeneintrag angelegt werden. Dies ist möglichst anwenderfreundlich zu realisieren.
Hinzugefügte Verknüpfungen zu Kategorien sollen auch wieder entfernt werden können.

Vergessen Sie nicht die Abbildung des Optimistic Lockings!

Tabellen: Tasks, TaskToCategoryRelations

Schreiben Sie 5 relevante Unit-Tests für diese Funktionalität.

Sortierung der Anzeige der Listeneinträge (5 Punkte):

Der Anwender soll die Möglichkeit haben, die Anzeigereihenfolge der Listeneinträge zu konfigurieren.

Folgende Kriterien sollen einstellbar sein:
- Anlagedatum
- Alphabetisch
- Fälligkeitsdatum
- Priorität
- Wichtigste (nach Fälligkeitsdatum und Priorität)

Standardmäßig sollen die Listeneinträge nach dem Anlagedatum sortiert werden.

Verwenden Sie zusätzlich zur nHibernate-Funktionalität LINQ um die notwendige Abfrage zu implementieren.

Schreiben Sie 5 relevante Unit-Tests für diese Funktionalität.

Import von TODO-Listen (5 Punkte):

Zusätzlich zur Eingabe über das Programm soll es möglich sein, TODO-Listen aus XML-Dateien zu importieren.

Das Datenformat der XML-Dateien können Sie folgendem Beispiel entnehmen:

```
<?xml version="1.0"?>
<TaskList>
	<Name>Einkaufsliste</Name>
	<Remark>Nur Lebensmittel</Remark>
	<Tasks>
		<Task>
			<Name>Eier</Name>
			<Remark>nur die guten Bio-Eier</Remark>
			<IsDone>False</IsDone>
			<Categories>
				<Category Name="Nicht vegan" />
				<Category Name="Backzutaten" />
			</Categories>
		</Task>
		<Task>
			<Name>Soja-Milch</Name>
			<IsDone>False</IsDone>
			<Categories>
				<Category Name="vegan" />
			</Categories>
		</Task>
		<Task>
			<Name>Hackfleisch</Name>
		</Task>
	</Tasks>
</TaskList>
```

Bieten Sie in der Software die Möglichkeit zur Auswahl einer Import-Datei über einen Datei-Öffnen-Dialog (WPF) an.

Da die Import-Dateien unter Umständen nicht alle Pflichtangaben enthalten müssen, finden Sie sinnvolle Voreinstellungen für diese Fälle.

Rechnen Sie damit, dass Sie evtl. auch korrupte Import-Dateien bekommen könnten. Sichern Sie den Import-Prozess entsprechend ab.

Schreiben Sie 5 relevante Unit-Tests für diese Funktionalität.

Export von TODO-Listen (5 Punkte):

Es soll in der Software möglich sein, TODO-Listen in XML-Dateien zu exportieren.

Das Datenformat der XML-Dateien entspricht dem Datenformat, das importiert werden soll. Exportierte Daten sollen sich auch wieder importieren lassen.

Bieten Sie in der Software die Möglichkeit zur Auswahl einer Export-Datei über einen Datei-Speichern-Dialog (WPF) an.

Schreiben Sie 5 relevante Unit-Tests für diese Funktionalität.

Dokumentation (10 Punkte):

Das Projekt und die Vorgehensweise sollen in einer kurzen Dokumentation vorgestellt werden.

Die Dokumentation soll das Oberflächen-Design vorstellen.
Bitte beschreiben Sie Ihre Beweggründe und was Sie mit dem Bedienaufbau des Softwaresystems erreichen wollen.

Die Dokumentation soll die Architektur des Softwaresystems beschreiben.
Fassen Sie die einzelnen Komponenten zu Bereichen zusammen und erläutern Sie diese.

Die Dokumentation soll aufführen, welche Aufgaben (und wie) bearbeitet wurden.

Die Dokumentation sollte (mit Bildern) ca. 10-15 DIN A4 Seiten umfassen.
Ausschließlich Bilder werden allerdings nicht akzeptiert.

Die Dokumentation muss als PDF oder als MS Word Dokument abgegeben werden.

Allgemeine Tipps zur Lösung der Projektaufgabe:

Lesen Sie zuerst alle Aufgaben durch und auch diese Liste mit Tipps. Sie sparen Zeit, wenn Sie zuerst alle Anforderungen sichten und bei der Implementierung der Aufgaben berücksichtigen. Dies ist besonders bei grundsätzlichen Entscheidungen wichtig!

Gehen Sie „datengetrieben“ vor.
Legen Sie für die notwendigen Daten das Datenmodell in Klassenform fest.

Stellen Sie danach die notwendigen Grundfunktionalitäten Richtung Datenbank sicher, bevor Sie die Anwendung als solche implementieren. (z.B. Laden, Speichern, Aktualisieren und Löschen der entsprechenden Daten)

Verschönerungen an der Oberfläche sollten Sie erst zum Schluss umsetzen.

Hängen Sie bei der Implementierung einer Teilfunktionalität, sollten Sie sich frühzeitig für die Implementierung der anderen Teilaufgaben entscheiden.
Haben Sie später noch genug Zeit, können Sie sich um die „Problemaufgabe“ kümmern.

Programmabstürze führen zu Punktabzug!

Rechnen Sie immer damit, dass Sie Fehleingaben erhalten. Prüfen Sie diese ab.
In der Realität kann es durchaus vorkommen, dass Sie fehlerhafte Daten für einen Datenimport erhalten.

Wenn Sie bei einigen geforderten Funktionalitäten nicht genau wissen, wie Sie diese implementieren können, steht es Ihnen frei, eine Suchmaschine Ihrer Wahl zu Rate zu ziehen. Hierbei hat sich die englische Sprache als besonders geeignet erwiesen.

Versuchen Sie die einzelnen Aufgaben vollständig zu lösen. Lösen Sie Aufgaben nicht nur zum Teil.

Vermeiden Sie Rechenübungen bzgl. ausreichender Punktzahl und der Anzahl bearbeiteter Aufgaben. Lösen Sie stattdessen die Aufgaben.

Vergessen Sie nicht die Dokumentation und die Unit-Tests. Wenn Sie diese Teilaufgaben nicht bearbeiten verschenken Sie wichtige Punkte!