Semesterprojekt

«contact manager»

Autor: Larissa Fitze im Auftrag des Zentrum für berufliche Weiterbildung

Letzte Änderung am 11.04.2024

Zentrum für berufliche Weiterbildung

Semesterprojekt «contact manager»

1. Ausgangssituation

Im Rahmen des Moduls „Programming Foundation II“ soll das erworbene Wissen an einer konkreten Problemstellung vertieft und verinnerlicht werden. Dabei kommen viele diskutierte Unterrichtsthemen zum Einsatz, teilweise werden auch Bausteine benötigt, welche nicht direkt im Unterricht besprochen wurden.

2. Vorgehen

Die Arbeit wird in Gruppen umgesetzt. Die verbleibenden Lektionen des Moduls stehen ausschliesslich für die Erarbeitung des Projekts bzw. der Beratung durch die Lehrperson zur Verfügung.

Es handelt sich um eine Prüfungssituation mit praxisorientiertem Charakter. Die Note dieses Projekts fliesst in die Endnote der Semesternote ein.

3. Aufgabenstellung

Es soll eine Microsoft Window-Forms Applikation erstellt werden, welche die Verwaltung und Pflege von Mitarbeiter- und Kundeninformationen vereinfacht. Da sie ausschliesslich Schweizer Kunden bedienen, müssen sie bei der anwendungsspezifischen Anforderungen nur die Schweiz berücksichtigen. Dabei sollen die üblichen CRUD (Create, Read, Update, Delete) Möglichkeiten umgesetzt werden. Nebst den allgemeinen zu verwaltenden Informationen soll es möglich sein, dass die Kontaktaufnahme zu Kunden in Form einer laufenden Dokumentation nachvollziehbar wird. Jeder Kontakt mit Kunden wird protokolliert und kann zu einem späteren Zeitpunkt eingesehen werden.
Folgende Informationen gilt es zu verwalten:
Anrede, Vorname, Nachname, Geburtsdatum, Geschlecht, Titel, Telefonnummer Geschäft, Mobiltelefonnummer, E-Mail-Adresse, Status (aktiv, passiv)
Mitarbeiternummer, Abteilung, AHV-Nummer, Wohnort, Nationalität, Adresse, Postleitzahl, Telefonnummer privat, Eintrittsdatum, Austrittdatum, Beschäftigungsgrad, Rolle (Tätigkeitsbezeichnung), Kaderstufe (0-5), Lehrjahre, aktuelles Lehrjahr (wenn Lernender) Firmenname, Geschäftsadresse, Kundentyp (A-E), Firmenkontakt

4. Funktionale Anforderungen

- Erfassung von Mitarbeitern und Kunden
- Mutieren von Mitarbeitern und Kunden
- Aktivieren und deaktivieren von Mitarbeitern und Kunden
- Löschen von Mitarbeitern und Kunden
- Automatische Vergabe von Mitarbeiternummern
- Protokollieren von Notizen in Kundenkontakten inkl. Historie
- Suchmöglichkeiten über gespeicherten Informationen (Name, Vorname, Geburtsdatum, Mitarbeiter/Kunde)
- Automatisches Speichern des Datenstamms auf Festplatte

5. Optionale funktionale Anforderungen

- Mutationshistorie von Kontaktdaten
- Login
- Eine Dashboard-View die sinnvoll ist
- Import von Kontakten im CSV-Format oder VCard-Format
    - Hinweis: Stellt ein Beispiel CSV bereit, welches importiert werden kann

6. Nicht funktionale Anforderungen

- Umsetzung in C# .Net und Windows-Forms Framework
- Durchdachte Applikationsarchitektur (Vererbungshierarchie der Daten ist Pflicht, siehe Abschnitt 10 - Anhang)
- Gute Benutzbarkeit (Usability)
- Hohe Stabilität (Fehleingaben abfangen, Abstürze verhindern)
- Ausreichende In-Line Dokumentation des Quellcodes (alle public Klassen/Methoden/Properties etc.)

7. Teamarbeit

- Die Arbeit wird auf die Gruppe aufgeteilt. Jedes Gruppenmitglied ist für einen Teilbereich der Implementierung zuständig.
- Das Team bestimmt eine/n Teamleiter/in, welche für die Führung der Gruppe zuständig ist.

8. Abgabe der Arbeit

- Abgabedatum ist 22.09.2024 23.00 Uhr.
- Das Projekt ist auf GitHub zuladen. Der Teamleiter sorgt dafür, dass die Lehrerin den Link zum Projekt erhält.
- Erfolgen Commits nach dem Abgabedatum, wird das ganze Projekt mit der Note 1 bewertet.
- Im Projekt existiert eine Text-Datei mit folgenden Informationen:
    - Gruppenmitglieder namentlich mit Vor- und Nachnamen
    - Eine Beschreibung was funktioniert und was nicht
    - Ggf. Zusatzinformationen wie Login (Benutzername /Passwort) etc.
- Freiwillig:
    - Es kann ein Arbeitsjournal geführt werden (das muss jedoch nicht abgegeben werden)
    - Zweck: Beweisfunktion im Falle, dass jemand nichts am Projekt macht, wird das entsprechend bei der Bewertung berücksichtigt.

9. Bewertung

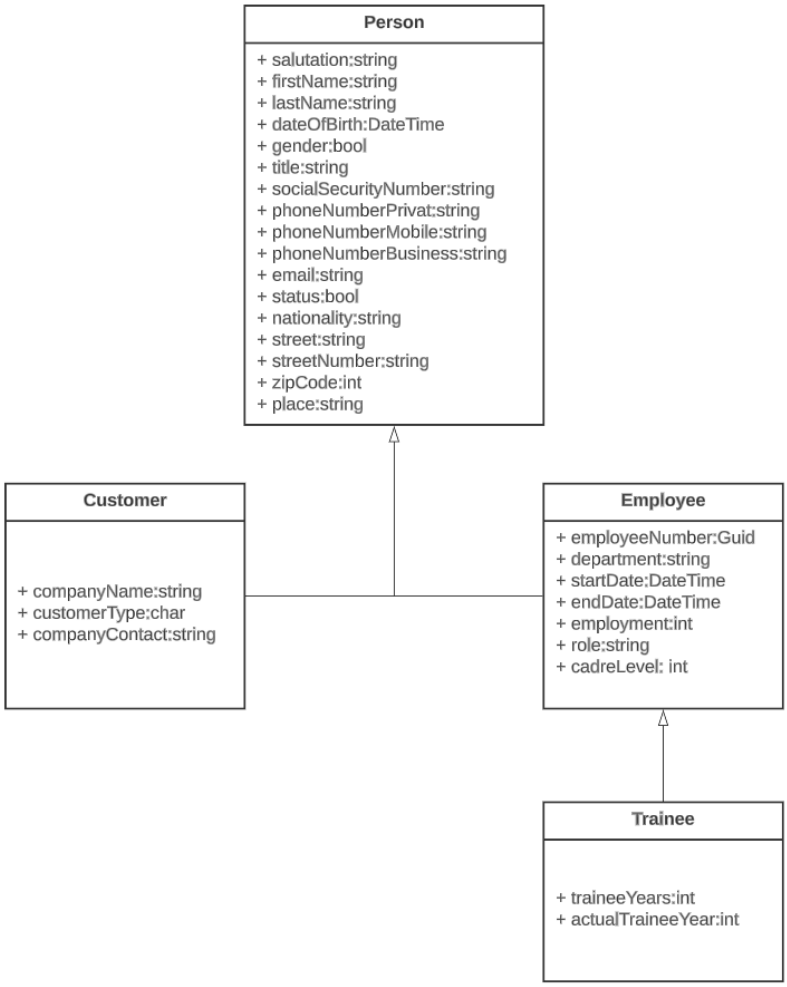
Kriterium	Gut Erfüllt	Erfüllt	Mangelhaft erfüllt	Nicht erfüllt
Punkte	2.5	2.0	1.0	0.0
1.0 Bewertungskriterien				
Funktionale Anforderungen				
Architektur der Applikation				
Qualität der Benennung von Klassen, Methoden, Properties und Feldern				
Qualität der Menge von Code in Klassen und Methoden inkl. Zuständigkeit				
Benutzerfreundlichkeit				
Quantität und Qualität der Inline-Dokumentation				
Applikationsstabilität				
	0.0	0.0	0.0	0.0
Endnote für diesen Bereich	1.0			

10. Tipps

- Klären Sie die Rollen innerhalb des Teams. Die Rolle der Teamleitung ist frühzeitig zu bestimmen.
- Planen Sie die Ihnen zu Verfügung stehende Zeit gut ein!
  Geplant sind jeweils 40h / Teilnehmer*in. Überlegen Sie sich, wie dieser geschätzte Zeitaufwand in Ihren Alltag passt. Schaffen Sie sich aber auch Pufferzeiten, denn wenn Sie auf ein Problem stossen, verfliegt die Zeit schneller als Ihnen lieb ist.
- Definieren Sie erste grobe Meilensteine für Teilresultate. Erarbeiten Sie sich eine Kommunikationsstrategie (wann und wo und wie oft treffen sich die Teilnehmenden zum Austausch).
- Starten Sie nicht gleich mit der Programmierung. Überlegen Sie in der Gruppe, wie Sie das Problem lösen könnten und teilen Sie die Aufgaben untereinander themenorientiert auf.
- Erarbeiten Sie zuerst auf Papier eine möglichst genaue visuelle Darstellung Ihrer GUI. Dies hilft Ihnen anschliessend besser zu verstehen, was dann im Hintergrund passieren muss, bzw. welche Klassen und Objekte tatsächlich notwendig sind.
- Halten Sie sich mit zu vielen zusätzlichen Anforderungen zurück. Setzen Sie zu Beginn nur das um, was von Ihnen verlangt wird.
- Verbringen Sie nicht Stunden mit einer detaillierten Ausarbeitung der grafischen Oberfläche. Versuchen Sie eine ansprechende und intuitive, aber dennoch einfache GUI zu gestalten.
- Bleiben Sie mit Ihren Teamkollegen*innen in Kontakt. Diskutieren Sie laufend Abhängigkeiten zwischen den einzelnen Aufgaben und unterstützen Sie sich gegenseitig. Je mehr Sie kommunizieren, desto stärker entwickelt sich eine gemeinsame Vorstellung des Resultats.

11. Anhang

Abstrahierte Vererbungshierarchie in Model



Wichtige Links für das Projekt

- Lesen und schreiben von Text-Dateien:
  https://docs.microsoft.com/de-de/troubleshoot/dotnet/csharp/read-write-text-file
- Objekte in Dateien schreiben und wieder einlesen:
  https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide/concepts/serialization/
- C#-CrashKurs Videos auf Youtube
  https://www.youtube.com/watch?v=l0aKBziWBH8&list=PL_pqkvxZ6ho18awjThtUMZio-yvc79TGi