Corona-Contact-Tracing

MLZ-Prüfung

Fach	CS2 Programmierung
Klasse	20I, 21NI
Lehrperson	Sagi Nedunkanal
Semester	FS 2022
Prüfungsbeginn	07.07.2022
Abgabetermin	15.07.2022, 24:00 Uhr

1. Lernziel

Das Hauptlernziel ist eine Applikation so struktrurieren, dass es möglich wird, bestimme Komponenten (UI/Dateisysystem) abzukoppeln.

Damit kann man die entsprechenden Komponenten austauschbar machen und die Applikation testbar machen, indem die Komponenten für das Testen mit Platzhaltern (Stubs/Mocks) ersetzt werden können.

2. Ausgangslage

Es soll ein Contact-Tracing realisiert werden, die abhängig von Tracing-Daten der infizierten Personen ihre Kontakte benachrichtigen kann, damit die sich in Quarantäne begeben können. So kann man das Lockdown lockern, aber trotzdem «die Infektionskurve flach halten», indem neue Infektionsherde rasch eingedämmt werden.

Die Aufgabenstellung ist nicht in allen Details ausspezifiziert. Bei Unklarheiten wird erwartet, dass beim Dozenten nachgefragt wird.

3. Anforderungen

3.1. Benutzeroberfläche

Schreibe eine WPF-Applikation. Die Applikation soll vier Bereiche darstellen:

1. Ein Bereich stellt das Contact-Tracing dar, welches die Daten der Mobiltelefone darstellt.
2. Ein Bereich soll ermöglichen, Personen als «krank» zu markieren, damit später die Kontaktpersonen benachrichtigt werden können.
3. Ein Bereich soll Funktionen erhalten: einen Knopf, um die Tracing-Daten auszuwerten und die betroffenen Personen zu informieren und einen Knopf, um ein paar Testdaten einzupflegen.
4. Ein Dashboard-Bereich zeigt an, welcher Personen in Quarantäne gehen müssen und welche nicht

Wähle geeignete GUI-Controls, um die Bedienoberfläche optisch ansprechend zu gestalten und intuitiv zu bedienen.

3.2. Architektur

Verwendet das MVVM-Muster mit .NET Framework 4.8.

3.3. Bereich 1/4: Contact-Tracing-Daten anzeigen

Schau Dir das Youtube-Video (8min) «Does contact tracing necessarily sacrifice privacy? (via Nicky Case)» von 3Blue1Brown an https://www.youtube.com/watch?v=D__UaR5MQao, um das Konzept des Contact-Tracings zu verstehen.
Für unsere Applikation vereinfachen wir die Anforderung, indem wir fix vier Personen darstellen

Person 1		Person 2		Person 3		Person 4	
Gesendet	Empfangen	Gesendet	Empfangen	Gesendet	Empfangen	Gesendet	Empfangen
A B C	D	D E F	G A	G H I	D	J K L	<leer>

Die Felder mit «A, B, C» usw. sind Multi-line-WPF-TextBoxen, damit der Benutzer einfach Werte hineinkopieren kann. Falls andere Controls, wie z.B. Grids verwendet werden sollen, bitte kurz absprechen.
Zur Vereinfachung lassen wir auch Zeitstempel weg.

3.4. Bereich 2/4: Personen als «krank» markieren

In diesem Bereich kann der Benutzer den Arzt simulieren, der eine Person untersucht. Wenn das Untersuchungsergebnis ist, dass die Person mit dem Coronavirus angesteckt ist, dann kann z.B. mit einer Checkbox eine Person als «krank» oder «infiziert» markiert werden.

3.5. Bereich 3/4: Analyse durchführen und einen Kontakt simulieren

Der Benutzer kann eine Analyse starten. Die Analyse zieht die Daten aus dem Contact-Tracing an (siehe Bereich 1/4) und die Daten vom Arzt (siehe Bereich 2/4) und findet heraus, welcher Personen in Quarantäne gehen müssen und welche nicht. Alle kranken Personen müssen in Quarantäne und alle Personen, die mit der erkrankten Person in Kontakt gekommen sind. Das Ergebnis wird im Bereich 4/4 visualisiert.
Zusätzlich kann der Benutzer zu Simulationszwecken die «Gesendet»- und «Empfangen»-Daten der Personen 3 und 4 so manipulieren, dass es so aussieht, als wären sie in Kontakt gekommen.

3.6. Bereich 4/4: Dashboard mit den Benachrichtigungen

Das Dashboard zeigt das Ergebnis der Analyse (siehe Bereich 3/4) an. Für die vier Personen wird angezeigt, wer in Quarantäne muss und wer nicht.

4. Datenhaltung

Beim Starten der Applikation soll genau das Szenario aus Kapitel 3.3 geladen werden. Zur Vereinfachung wird keine Datenbank verwendet, sondern nur Textdateien. Beispielsweise:

```
Person1_Gesendet.txt
Person1_Empfangen.txt
Person2_Gesendet.txt
Person2_Empfangen.txt
Person3_Gesendet.txt
Person4_Empfangen.txt
Person5_Gesendet.txt
Person6_Empfangen.txt
```

Tipp:
https://stackoverflow.com/questions/7569904/easiest-way-to-read-from-and-write-to-files

641

Use File.ReadAllText and File.WriteAllText.

MSDN example excerpt:

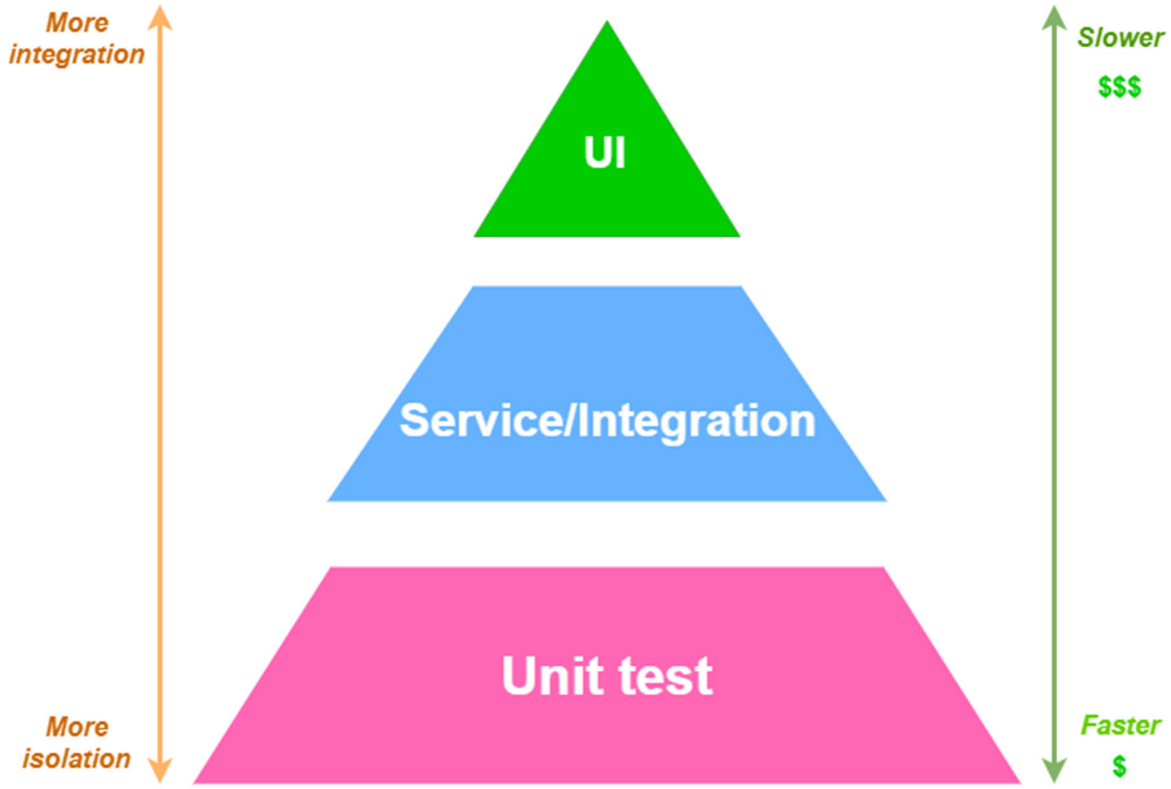
```
// Create a file to write to.
string createText = "Hello and Welcome" + Environment.NewLine;
File.WriteAllText(path, createText);

...

// Open the file to read from.
string readText = File.ReadAllText(path);
```

5. Qualitätssicherung

5.1. Testdaten

Beim Start der Applikation sollen die Daten gemäss Kapitel 4 geladen werden. Die Daten sind so konstruiert, dass Personen 1, 2 und 3 miteinander in Kontakt waren und Person 4 nicht.

5.2. Testpyramide



Quelle: https://anymindgroup.com/news/tech-blog/15053/

5.3. Manuelle UI-Tests

Es soll mindestens ein manueller Test dokumentiert werden.
Vorlage für die Dokumentation:

Schritte zum Reproduzieren
1. Applikation starten
2. Knopf X drücken
3. Im der Multi-line-Textbox der Person 1 bei “Gesendet” das und jenes eintragen
4. Knopf Y drücken

Erwartetes Ergebnis
- Person 3 wird als “Muss in Quarantäne!” markiert

Tatsächliches Ergebnis
- Im positiven Fall
  - Person 3 wurde als “Muss in Quarantäne!” markiert
  - <Screenshot einfügen>
- Im negativen Fall
  - Person 3 hätte als “Muss in Quarantäne!” markiert werden sollen, wurde sie aber nicht, stattdessen wurde Person 7 markiert
  - <Screenshot einfügen>

5.4. Automatisierte Integrationstests

Es sollen ein paar wenige Integrationstests geschrieben werden, damit sichergestellt wird, dass alle Umsysteme korrekt funktionieren: ungefähr ein bis drei Smoketests. Das heisst, hier wird das ganze System inclusive Umsysteme (aber ohne UI) getestet.

5.5. Automatisierte Unittests

Hier soll der Kern der Applikation ohne Umsysteme getestet werden, sprich die ganze Geschäftslogik.

Umsysteme sollen abgehängt und durch Platzhalter für das Testen ersetzt werden.

Es sollen mindestens fünf Unittest für die Applikation geschrieben werden. Der erste Test soll das Szenario aus Kapitel 3.3 abbilden:

// Arrange	Gegeben seien vier Personen mit den Testdaten aus Kapitel 3.3. Person 2 sei infiziert.
// Act	Die Analyse wird ausgeführt.
// Assert	Das erwartete Ergebnis ist, dass Personen 1, 2 und 3 in Quarantäne müssen (P2 wegen Infektion, P1 und P3 wegen potenzieller Ansteckung) und Person 4 nicht.

5.6. Versionsverwaltung

Verwende die Git-Versionsverwaltung, um regelmässig Sicherungen des aktuellen Standes zu erstellen (Commit) und hochzuladen (Push). Mindestens drei Commits sind gefordert.

6. Dokumentation

Es soll eine minimale Dokumentation erstellt werden:
- Lösungskizze (kann von Hand gezeichnet, fotografiert und als JPEG abgelegt werden; kein im Nachhinein generiertes Klassendiagramm)

    - Auf der Lösungsskizze soll ersichtlich sein, wo die Sollbruchstellen zu den Umsystemen definiert sind
- Screenshots aller Fenster
- Screenshot der ausgeführten Testfälle

7. Bewertungskriterien

1. Umsetzung der funktionalen Anforderungen (1/4)
2. Bewertung des GUIs (1/4)
3. Bewertung des Codes (1/4)
4. Testen der Applikation (1/4)

- Erlaubt: Internet als Quelle, Beispiele aus dem Unterricht usw.
- Nicht erlaubt: Kopieren grösserer Codemengen, Mitarbeit Dritter usw.
    - Selbstkontrolle: «Kann ich jede Zeile Code erklären?»
- Missachtung der Prüfungsregeln führen zu einer Note 1.
- Verspätete Abgabe führt zu Notenabzug:
    - Bis 6h Verspätung → 0.5 Note Abzug
    - Bis 12h Verspätung → 1.0 Note Abzug
    - >12h Verspätung → Note 1.0

8. Abgabe

Die Applikation wird via GitHub abgegeben:

1. Den Quellcode ablegen unter
   `CS2\02 Prüfungen\Prüfung 3 - MLZ\Quellcode\CoronaContactTracing`
   und dabei die vorbereitete Solution-Datei verwenden: «`Vorname Nachname CS2 Prüfung 3 MLZ.sln`»
2. Das Kompilat als ZIP ablegen unter
   `CS2\02 Prüfungen\Prüfung 3 - MLZ\Kompilat\CoronaContactTracing.zip`
3. Die Dokumentation ablegen unter
   `CS2\02 Prüfungen\Prüfung 3 - MLZ\Dokumentation`
4. Alles hochladen

9. Prüfungsreglement

MLZ-Prüfungen müssen von den Studierenden zum ordentlich geplanten Termin geschrieben werden. Andernfalls fallen Kosten von CHF 300.- an.