HET BEGINT BIJ DON BOSCO HALLE

geïntegreerde proef

gip

Winkelbeheer Software

Zias van Nes

Klas 6TIC

Schooljaar 2021-2022

WOORD VOORAF

Dit werk werd gemaakt door Zias van Nes uit het 6e jaar Industriële Computertechnieken. Om deze opleiding te kunnen slagen moet er een geïntegreerde proef worden afgelegd. Dit is een werk waarin verschillende vakonderdelen verwerkt worden in een eigen gekozen project.

Voor mijn gip maakte ik een software die zowel als magazijn database of als kassa kan dienen. Het magazijn programma dient als een database waarin je grafisch nieuwe producten kan toevoegen aan de software. Je kan hiermee ook een product verwijderen of wijzigen. Elk product heeft specifieke eigenschappen zoals zijn prijs, eventuele korting, categorie, gewenst aantal producten aanwezig en huidig aantal producten aanwezig. De kassa is het programma waarmee de producten uit het magazijn “afgerekend” kunnen worden, dit betekent dat de prijs van de producten wordt opgeteld en een totaalsom wordt gemaakt. Nadien wordt het aantal van een product verminderd waardoor de producten die zijn gekocht niet meer in de winkel zijn geregistreerd.

Voor dit werk heb ik veel ondersteuning gekregen. Ik wil graag mijn moeder bedanken, die dit werk heeft nagekeken. Ik wil graag mijn vader bedanken, die mij heeft geholpen met de uitwerking van dit project en mij veel goede ideeën heeft gegeven voor het realiseren van dit werk. Tenslotte wil ik ook mijn vakleerkracht Becqué K. alsook mijn leerkracht Nederlands Brynaert J. bedanken voor hun steun bij dit project. Door dit project te realiseren heb ik enorm veel bijgeleerd. Ik ben dus enorm dankbaar dat ik de kans heb gekregen om dit project te maken.

INHOUDSOPGAVE

INLEIDING

Met dit project wil ik graag een software maken die een winkel autonoom zou kunnen besturen. Op die manier zouden personeelsleden van een winkel veel minder moeten doen en zou er dus minder werkkracht nodig zijn voor het aankopen en beheren van producten.

Het doel van dit project is om een software te maken met een database van producten en hun bijbehorende eigenschappen. Het is de bedoeling dat er nieuwe producten kunnen worden aangemaakt en bestaande producten aangepast kunnen worden. Vervolgens is het de bedoeling dat ik een programma maak om producten uit de database op te vragen en op die manier een kassabon aan te maken. Als de producten dan worden gekocht, moet er na een bepaalde grens van minimumproducten een melding worden gegeven welke producten opnieuw moeten worden aangekocht.

Om dit project uit te breiden zou ik een barcode kunnen maken per product, zodat ik die kan scannen en op die manier kan toevoegen aan het kassaticket. Ik zou eventueel ook een functie kunnen toevoegen die een document maakt waarop een rekening visueel wordt weergegeven (bv een pdf).

1 Database

1.1 Doel

Het doel van dit onderdeel is om een programma te schrijven die alle producten kan beheren. Men moet dus producten kunnen aanmaken, wijzigen en verwijderen. Men moet ook gegevens van een bepaald product kunnen opvragen en tijdelijke kortingen op een product kunnen geven.

1.2 Realisering

1.2.1 Acces database

Mijn eerste denkpiste voor dit project was om een acces database te maken die de producten bevat. Ik merkte snel dat dit inefficiënt zou zijn. Voornamelijk de connectie met het programma maken was moeilijk realiseerbaar en ook niet de ideale oplossing voor mijn programma.

1.2.2 SQL-server

Wanneer ik merkte dat een Acces database niet efficiënt genoeg was, zocht ik naar een andere oplossing. Ik kwam snel uit op een SQL-server database. Ik had hiermee nog nooit gewerkt en ik merkte snel dat dit niet erg intuïtief was. SQL-server is een heel krachtige database, maar ik heb dit uiteindelijk niet gebruikt omdat het te complex was voor zijn toepassing.

1.3 XML

1.3.1 Wat is XML

“De XML-indeling slaat gegevens op in een structuur die machineleesbaar en voor mensen leesbaar is. Er zijn een groot aantal programma’s die xml bestanden kunnen openen. En aangezien ze zijn opgemaakt als tekstdocumenten, kunnen ze worden bekeken en bewerkt door teksteditors.” [1]

1.3.2 Voorbeeld XML

In het databaseprogramma worden producten opgeslagen in een xml bestand. In dit bestand worden de eigenschappen van een product zoals: de prijs, de categorie, korting, … opgeslagen [2].

```
1.  <Producten>
2.    <Product>
3.      <Naam>Appel</Naam>
4.      <Categorie>Fruit</Categorie>
5.      <Aantal>15</Aantal>
6.      <Bestaantal>20</Bestaantal>
7.      <Prijs>3</Prijs>
8.      <Korting>0</Korting>
9.      <NieuwePrijs>3</NieuwePrijs>
10.   </Product>
11.   <Product>
12.     <Naam>Peer</Naam>
13.     <Categorie>Fruit</Categorie>
14.     <Aantal>15</Aantal>
15.     <Bestaantal>20</Bestaantal>
16.     <Prijs>4</Prijs>
17.     <Korting>10</Korting>
18.     <NieuwePrijs>3,6</NieuwePrijs>
19.   </Product>
20. </Producten>
```

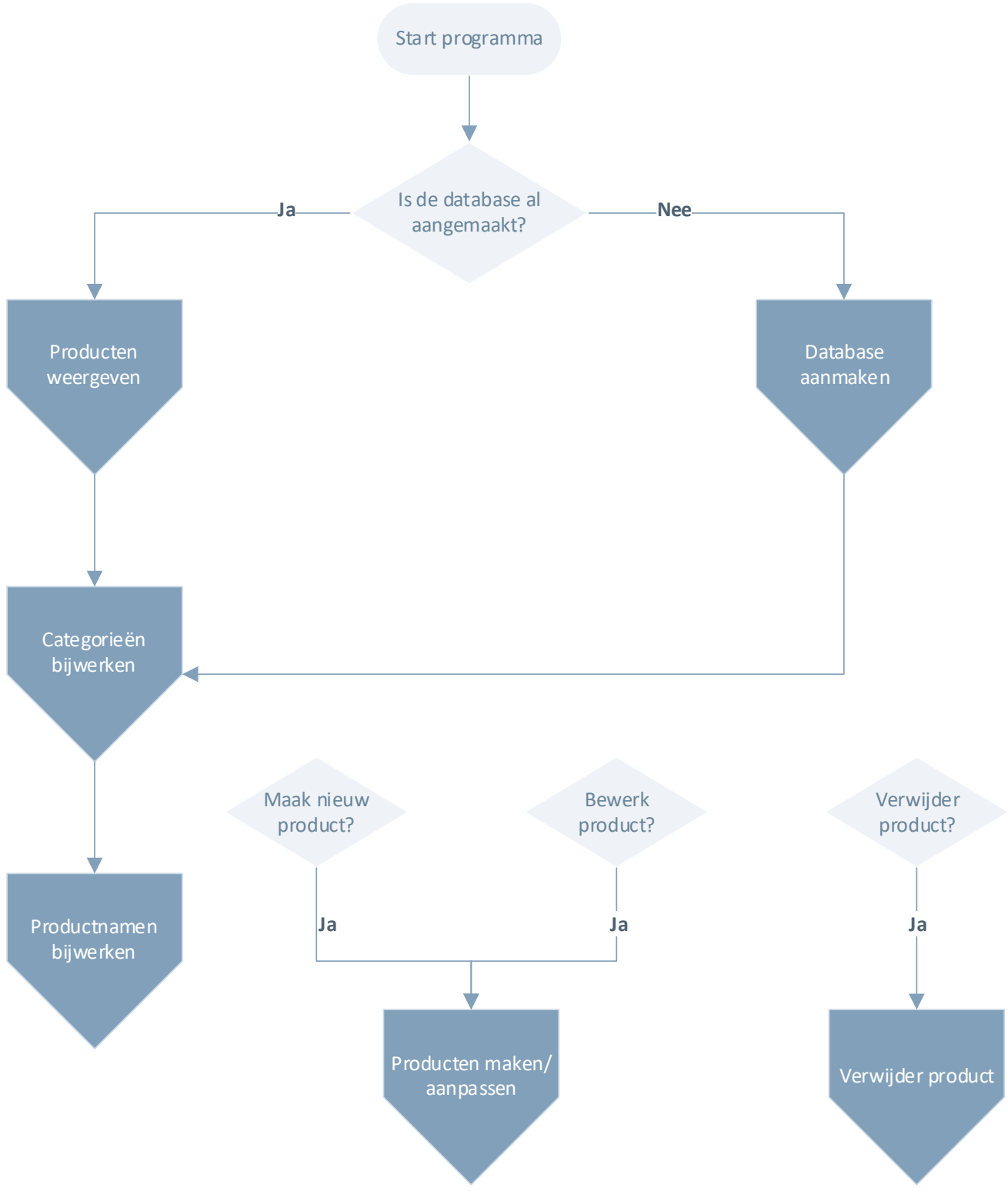
Code 1, xml code die een voorbeeld weergeeft van de xml database

[1] *(Bestandinfo, 2022)*

[2] *Zie Code 1*

1.4 Werking database

1.4.1 Overzicht blokdiagram



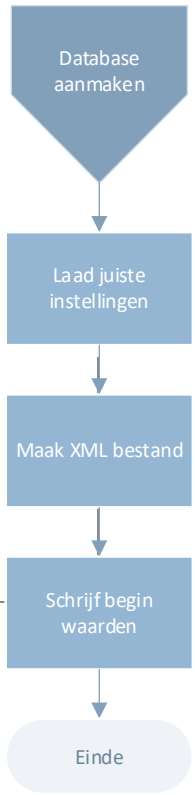
Figuur 1, overzichtsschema van het database programma

1.4.2 Database aanmaken

Wanneer het programma voor het eerst wordt opgestart is er nog geen database aangemaakt. Bij de start van het programma moet er dus worden gecontroleerd of er reeds een database is. Bovendien moet er ook worden gecontroleerd of de database met de juiste opmaak werd aangemaakt. Met deze method wordt dit gecontroleerd [1].

```
21. private void MakeXmlProducten()
22.         {
23.             //Deze method maakt een XML-bestand met de standaardwaarden. Op die manier
    kunnen er later producten in worden toegevoegd.
24.
25.             //De XMLWriter wordt klaargezet en zijn settings worden ingegeven.
26.             XmlWriter xml;
27.
28.             XmlWriterSettings settings = new XmlWriterSettings();
29.             settings.Indent = true;
30.             settings.OmitXmlDeclaration = true;
31.             settings.NewLineOnAttributes = true;
32.
33.             xml = XmlWriter.Create(filePath, settings);
34.
35.             //Het bestand wordt aangemaakt en de standaardwaarden worden ingevoerd.
36.             xml.WriteStartDocument();
37.             xml.WriteStartElement("Producten");
38.             xml.WriteEndElement();
39.             xml.Close();
40.         }
41.
```

Code 2, C# code die weergeeft wat er gebeurt om een xml bestand aan te maken



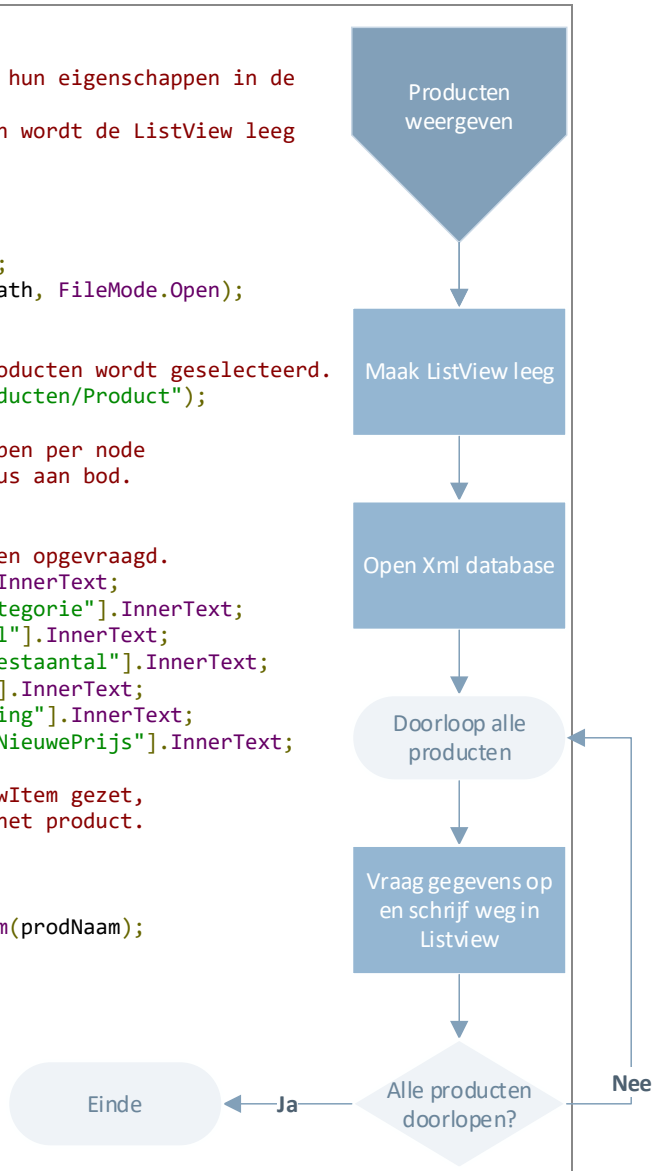
Figuur 2, blokschema dat weergeeft wat er gebeurt om een XML-bestand aan te maken

[1] *Zie Figuur 2 en Code 2*

1.4.3 Producten weergeven

Om op een efficiënte manier alle producten uit de database te kunnen weergeven is er een C# method gemaakt. In deze method worden alle producten uit de database doorlopen en één voor één aan de ListView toegevoegd. Op deze manier kunnen de producten eenvoudig opnieuw worden geladen bij bijvoorbeeld het maken van een nieuw product [1].

```
1.          private void FillListView()
2.          {
3.              //Deze method laadt alle producten met hun eigenschappen in de
    ListView.
4.              //Vooraleer de producten worden geladen wordt de ListView leeg
    gemaakt.
5.              lvProducten.Items.Clear();
6.
7.              //Het XML document wordt geopend.
8.              XmlDocument xmlDoc = new XmlDocument();
9.              FileStream file = new FileStream(filePath, FileMode.Open);
10.             xmlDoc.Load(file);
11.
12.             //De lijst XML node's met daarin de producten wordt geselecteerd.
13.             XmlNodeList = xmlDoc.SelectNodes("/Producten/Product");
14.
15.             //De lijst met XML node's wordt doorlopen per node
                --> ieder product komt op die manier dus aan bod.
16.             foreach (XmlNode in xmlNodeList)
17.             {
18.                 //De gegevens van het product worden opgevraagd.
19.                 string prodNaam = xmlNode["Naam"].InnerText;
20.                 string prodCategorie = xmlNode["Categorie"].InnerText;
21.                 string prodAantal = xmlNode["Aantal"].InnerText;
22.                 string prodBestAantal = xmlNode["Bestaantal"].InnerText;
23.                 string prodPrijs = xmlNode["Prijs"].InnerText;
24.                 string prodKorting = xmlNode["Korting"].InnerText;
25.                 string prodNieuwePrijs = xmlNode["NieuwePrijs"].InnerText;
26.
27.                 //Het product wordt in een ListViewItem gezet,
28. waarin er SubItems zijn voor de eigenschappen van het product.
29.
30.
31.
32.                 ListViewItem lvi = new ListViewItem(prodNaam);
33.
34.                 lvi.SubItems.Add(prodCategorie);
35.                 lvi.SubItems.Add(prodAantal);
36.                 lvi.SubItems.Add(prodBestAantal);
37.                 lvi.SubItems.Add(prodPrijs);
38.                 lvi.SubItems.Add(prodKorting);
39.                 lvi.SubItems.Add(prodNieuwePrijs);
40.
41.
42.                 //Het product wordt in het programma zichtbaar gemaakt door het in de ListView
    te zetten.
43.                 lvProducten.Items.Add(lvi);
44.             }
45.             //Het XML document wordt gesloten.
46.             file.Close();
47.         }
48.
```



Figuur 3, blokschema dat het proces van het opnieuw laden van de tabel visualiseert

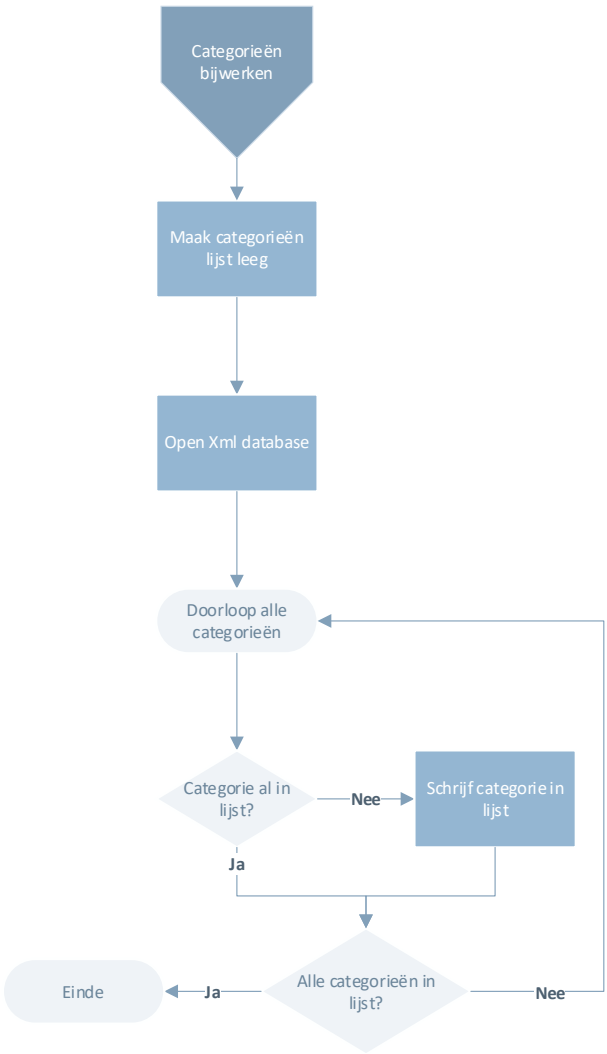
Code 3, C# code die het proces van het opnieuw laden van de tabel weergeeft

[1] *Zie Figuur 3 en Code 3*

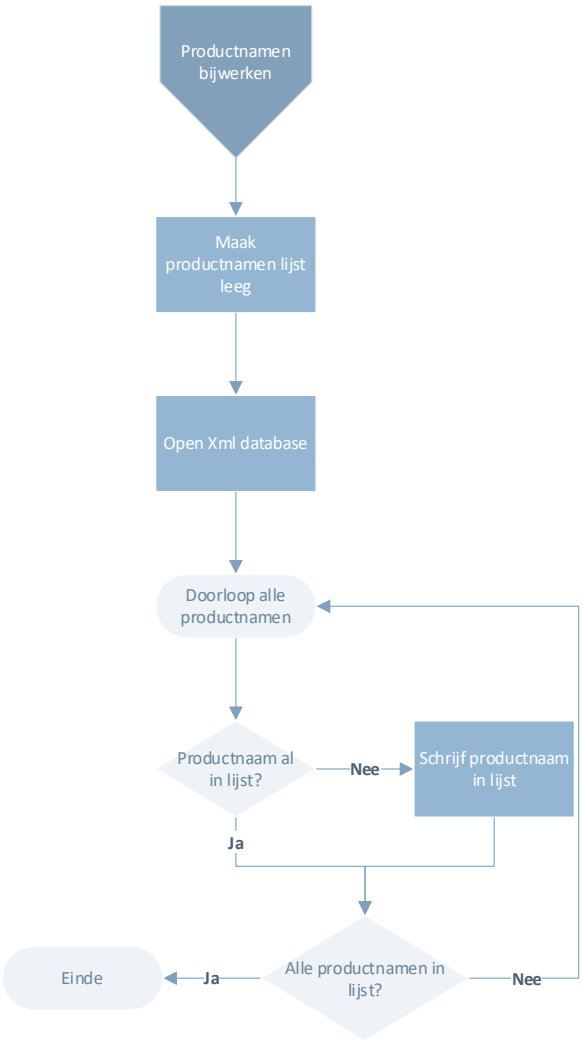
1.4.4 Productnamen en categorieën bijwerken

Als een product wordt aangemaakt moet er worden gecontroleerd of er al een product bestaat met die naam. Om dit vlot te kunnen doen is er een lijst gemaakt die wordt bijgewerkt als er een nieuw product wordt aangemaakt of verwijderd [1].

Het is erg tijdrovend om de categorie van een product telkens weer in te geven. Om dit te vermijden is er een lijst gemaakt die wordt bijgewerkt wanneer er een product wordt aangemaakt of verwijderd [2]. Hierdoor kan de gebruiker bij het maken van een product gemakkelijk kiezen tussen verschillende categorieën [3].



Figuur 5, blokschema dat het proces van categorieën bijwerken visualiseert



Figuur 4, blokschema dat het proces van productnamen bijwerken visualiseert

Figuur 6, afbeelding met voorbeeld van de auto complete van categorieën

[1] *Zie Figuur 4 en Code 4*

[2] *Zie Figuur 5 en Code 5*

[3] *Zie Figuur 6*

```
1.  private string[] wijzigNamen()
2.          {
3.              //Deze array geeft als resultaat alle product namen uit de XML-database.
4.
5.
6.              string[] namen = new string[0];
7.
8.              int count = 0;
9.
10.             XmlDocument xmlDoc = new XmlDocument();
11.             FileStream file = new FileStream(filePath, FileMode.Open);
12.             xmlDoc.Load(file);
13.
14.             XmlNodeList xmlNodeList = xmlDoc.SelectNodes("/Producten/Product/Naam");
15.
16.             foreach (XmlNode node in xmlNodeList)
17.             {
18.                 bool aanwezig = false;
19.                 foreach (string n in namen)
20.                 {
21.                     if (node.InnerText == n)
22.                     {
23.                         aanwezig = true;
24.                     }
25.                 }
26.                 if (!aanwezig)
27.                 {
28.                     Array.Resize(ref namen, namen.Length + 1);
29.                     namen[count] = node.InnerText;
30.                     count++;
31.                 }
32.             }
33.             file.Close();
34.
35.             return namen;
36.         }
37.
38.         private void werkNamenBij()
39.         {
40.             //Dit werkt de array met namen bij zodat deze up-to-date is.
41.
42.             //Lengte van de array +1.
43.             Array.Resize(ref namen, wijzigNamen().Length);
44.             //Werk namen bij met juiste array.
45.             namen = wijzigNamen();
46.         }
47.
```

Code 4, C# code die het proces van het opvragen van alle product namen weergeeft

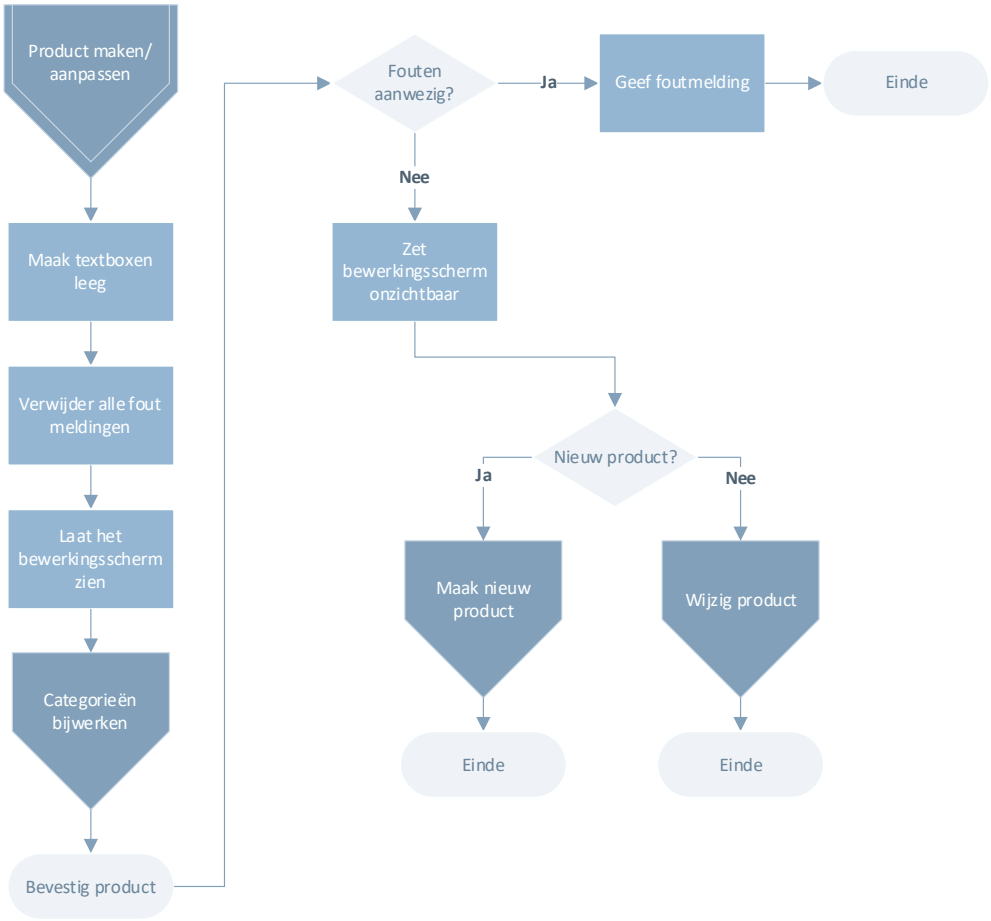
```
1.  private string[] wijzigCategorieën()
2.          {
3.              string[] categorieën = new string[0];
4.
5.              int count = 0;
6.
7.              XmlDocument xmlDoc = new XmlDocument();
8.              FileStream file = new FileStream(filePath, FileMode.Open);
9.              xmlDoc.Load(file);
10.
11.             XmlNodeList xmlNodeList = xmlDoc.SelectNodes("/Producten/Product/Categorie");
12.
13.             foreach (XmlNode node in xmlNodeList)
14.             {
15.                 bool aanwezig = false;
16.                 foreach (string n in categorieën)
17.                 {
18.                     if (node.InnerText == n)
19.                     {
20.                         aanwezig = true;
21.                     }
22.                 }
23.                 if (!aanwezig)
24.                 {
25.                     Array.Resize(ref categorieën, categorieën.Length + 1);
26.                     categorieën[count] = node.InnerText;
27.                     count++;
28.                 }
29.
30.             }
31.             file.Close();
32.
33.             return categorieën;
34.         }
35.
36.         private void werkCategorieënBij()
37.         {
38.             Array.Resize(ref categorieën, wijzigCategorieën().Length);
39.             categorieën = wijzigCategorieën();
40.         }
41.
```

Code 5, C# code die het proces van het opvragen van alle categorieën weergeeft

1.4.5 Toevoegen of wijzigen van een product

Als de gebruiker een product wil toevoegen of wijzigen kan dit door op de bijbehorende knop te klikken. Als dit gebeurt, verschijnt er een scherm [1].

Op dit scherm kunnen eigenschappen van een product ingevuld of aangepast worden. Wanneer de gebruiker op de knop “Bevestigen” klikt zal het product naargelang de gemaakte keuze toegevoegd of verwijderd worden [2].



Figuur 7, blokschema dat visualiseert wat er gebeurt om een product toe te voegen of te verwijderen

Figuur 8, afbeelding met voorbeeld voor het maken van een product

[1] *Zie Figuur 8*

[2] *Zie Figuur 7, Code 6 Code 6 en Code 7*

```
1.  private void btnMaakProduct_Click(object sender, EventArgs e)
2.          {
3.              //Deze method wordt geactiveerd bij het klikken op btnMaakproduct.
4.
5.              //Er moet een nieuw product worden gemaakt dus wordt newProd op 'true' gezet.
6.              newProd = true;
7.              //Alle textboxen worden leeggemaakt.
8.              ClearTxb();
9.              //Verwijder alle fouten van errorprovider.
10.             removeError();
11.             //Het panel om een product aan te maken wordt zichtbaar gemaakt.
12.             pnlProductEigenschappen.Visible = true;
13.
14.             werkCategorieënBij();
15.
16.             if (lvProducten.SelectedItems.Count == 1)
17.             {
18.                 selectedIndex = lvProducten.FocusedItem.Index;
19.             }
20.             else
21.             {
22.                 selectedIndex = 0;
23.             }
24.         }
25.
```

Code 6, C# code die weergeeft wat er gebeurt nadat er op de knop “Maak product” wordt geklikt.

```
1.  private void btnChangeProd_Click(object sender, EventArgs e)
2.          {
3.              //Deze method wordt geactiveerd bij het klikken op btnChangeProd.
4.
5.              //Om een product te wijzigen worden eerst de oorspronkelijke waarden ingevoerd,
    zodat er makkelijk veranderingen kunnen plaatsvinden.
6.              FillTxb();
7.              //Er is geen nieuw product.
8.              newProd = false;
9.              //De naam van het product dat wordt gewijzigd wordt extra ingegeven.
10.             huidigeNaam = GetTxbData(txbNaam);
11.
12.
13.             if (lvProducten.SelectedItems.Count == 1)
14.             {
15.                 selectedIndex = lvProducten.FocusedItem.Index;
16.             }
17.         }
18.
```

Code 7, C# code die weergeeft wat er gebeurt nadat er op de knop “Wijzig product” wordt geklikt

1.4.6 Product toevoegen

Om een product toe te voegen is er een method gemaakt in C#. Deze method maakt een nieuw product en zet dit in de xml database. De eigenschappen van het product vraagt hij op van de textbox. Vooraleer het product kan worden toegevoegd wordt er gecontroleerd of de waarden in de textbox realistische waarden zijn. Er wordt bijvoorbeeld gecontroleerd of er in de naam van het product enkel letters staan en dus geen cijfers. In deze method wordt ook onmiddellijk de prijs berekend aan de hand van de opgegeven korting.

Wanneer men er zeker van is dat de waarden realistisch zijn, wordt het product samen met zijn eigenschappen aangemaakt voor de xml database.

Wanneer de eigenschappen zijn aangemaakt wordt het product met zijn eigenschappen in de xml database toegevoegd en wordt het xml document gesloten. Hierna wordt er gecontroleerd of er een nieuwe productcategorie is [1].

```
1.  private void MaakNieuwProduct()
2.          {
3.              //Deze method maakt een nieuw product en zet deze in de XML-database.
4.
5.              //Er moet niet meer gecontroleerd worden of de waarden juist zijn, want dat werd al
    gedaan vooraleer deze method werd opgeroepen.
6.
7.
8.              //De XML-writer wordt klaargezet met het Xmldocument.
9.              XmlDocument xmlDoc = new XmlDocument();
10.             FileStream file = new FileStream(filePath, FileMode.Open);
11.             xmlDoc.Load(file);
12.
13.             //De eigenschappen van het product worden opgevraagd en opgeslagen.
14.             string prodNaam = GetTxbData(txbNaam);
15.             string prodCategorie = GetTxbData(txbCategorie);
16.             string prodAantal = GetTxbData(txbAantalAanwezig);
17.             string prodBestAantal = GetTxbData(txbAantalBestAanwezig);
18.             //Verander '.' door ',' zodat er geen verwarring is bij kommagetallen.
19.             string prodPrijs = verander(GetTxbData(txbPrijs), '.', ",");
20.             string prodKorting = GetTxbData(txbKorting);
21.
22.             //De nieuwePrijs wordt berekend, op basis van de prijs en de korting.
23.             string prijsVerander = verander(prodPrijs, '.', ",");
24.             decimal prijsProd = Convert.ToDecimal(prijsVerander);
25.             decimal kortingProd = Convert.ToDecimal(prodKorting);
26.             string nieuwePrijsProd = kortingPrijs(prijsProd, kortingProd).ToString();
27.
28.             //De elementen worden gemaakt.
29.             XmlElement product = xmlDoc.CreateElement("Product");
30.
31.             XmlElement naam = xmlDoc.CreateElement("Naam");
32.             XmlText naamText = xmlDoc.CreateTextNode(prodNaam);
33.
34.             XmlElement categorie = xmlDoc.CreateElement("Categorie");
35.             XmlText categorieText = xmlDoc.CreateTextNode(prodCategorie);
36.
37.             XmlElement aantal = xmlDoc.CreateElement("Aantal");
38.             XmlText aantalText = xmlDoc.CreateTextNode(prodAantal);
39.
40.             XmlElement aantalBest = xmlDoc.CreateElement("Bestaantal");
41.             XmlText aantalBestText = xmlDoc.CreateTextNode(prodBestAantal);
42.
43.             XmlElement prijs = xmlDoc.CreateElement("Prijs");
```

Figuur 9, blokschema dat visualiseert wat er gebeurt om een product toe te voegen

[1] *Zie Figuur 9 en Code 8*

```
44.             XmlText prijsText = xmlDoc.CreateTextNode(prodPrijs);
45.
46.             XmlElement korting = xmlDoc.CreateElement("Korting");
47.             XmlText kortingText = xmlDoc.CreateTextNode(prodKorting);
48.
49.             XmlElement nieuwePrijs = xmlDoc.CreateElement("NieuwePrijs");
50.             XmlText nieuwePrijsText = xmlDoc.CreateTextNode(nieuwePrijsProd);
51.
52.             //De elementen worden toegewezen met hun overeenkomstige waarde.
53.             naam.AppendChild(naamText);
54.             categorie.AppendChild(categorieText);
55.             aantal.AppendChild(aantalText);
56.             aantalBest.AppendChild(aantalBestText);
57.             prijs.AppendChild(prijsText);
58.             korting.AppendChild(kortingText);
59.             nieuwePrijs.AppendChild(nieuwePrijsText);
60.
61.             //De elementen worden toegevoegd aan een hoofdelement (product).
62.             product.AppendChild(naam);
63.             product.AppendChild(categorie);
64.             product.AppendChild(aantal);
65.             product.AppendChild(aantalBest);
66.             product.AppendChild(prijs);
67.             product.AppendChild(korting);
68.             product.AppendChild(nieuwePrijs);
69.
70.             //Het hoofdelement wordt toegevoegd aan het Xmldocument en het bestand wordt
    opgeslagen en gesloten.
71.             xmlDoc.DocumentElement.AppendChild(product);
72.             file.Close();
73.             xmlDoc.Save(filePath);
74.
75.             //De categorieën en namen worden bijgewerkt.
76.             werkCategorieënBij();
77.             werkNamenBij();
78.
79.             if (test)
80.             {
81.                 MessageBox.Show($"Prijs: {prodPrijs}\nKorting: {prodKorting}\nNieuwe prijs:
    {nieuwePrijsProd}");
82.             }
83.         }
```

Code 8, C# code die het proces van het toevoegen van een product weergeeft

1.4.7 Product aanpassen

Om een product aan te kunnen passen, is er een method gemaakt in C#. Deze method laat de gebruiker toe om een product zijn gegevens op te vragen en ze vervolgens aan te passen.

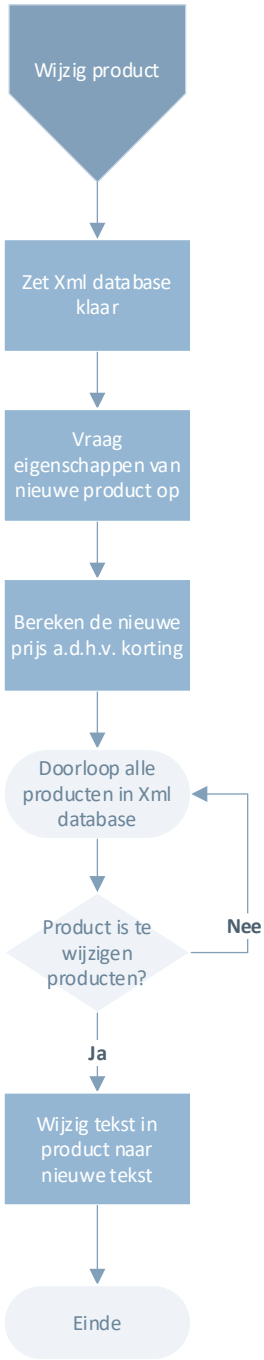
Vooraleer het product kan worden aangepast, wordt er nagegaan of er slechts één product is geselecteerd. Als dit in orde is, kunnen de gegevens van het product ingeladen worden in zijn overeenkomende textbox.

Om de eigenschappen van het product te wijzigen, worden eerst de ingevoerde gegevens gecontroleerd. Als deze waarden realistisch zijn, wordt de nieuwe prijs berekent aan de hand van zijn korting.

Vervolgens kunnen de gegevens aangepast worden. Het product wordt gezocht met behulp van de geselecteerde index uit de tabel. Hierna worden zijn gegevens overschreven met zijn nieuwe waarden [1].

```
1.  private void WijzigProduct()
2.          {
3.              //Deze method wijzigt een bepaald product en geeft zijn nieuwe
    waarden.
4.
5.              //Er moet niet meer gecontroleerd worden of de waarden juist zijn,
    want dat werd al gedaan vooraleer deze method werd opgeroepen.
6.
7.              //De XML-writer wordt klaargezet met het Xmldocument.
8.
9.              XmlDocument xmlDoc = new XmlDocument();
10.             FileStream file = new FileStream(filePath, FileMode.Open);
11.             xmlDoc.Load(file);
12.
13.             //De eigenschappen van het product worden opgevraagd en opgeslagen.
14.             string prodNaam = GetTxbData(txbNaam);
15.             string prodCategorie = GetTxbData(txbCategorie);
16.             string prodAantal = GetTxbData(txbAantalAanwezig);
17.             string prodBestAantal = GetTxbData(txbAantalBestAanwezig);
18.
19.             //Verander '.' door ',' zodat er geen verwarring is bij
    kommagetallen.
20.             string prodPrijs = verander(GetTxbData(txbPrijs), '.', ",");
21.             string prodKorting = GetTxbData(txbKorting);
22.
23.             //De nieuwePrijs wordt berekend, op basis van de prijs en de
    korting.
24.             decimal prijsProd = Convert.ToDecimal(prodPrijs);
25.             decimal kortingProd = Convert.ToDecimal(prodKorting);

26.             string nieuwePrijsProd = kortingPrijs(prijsProd,
    kortingProd).ToString();
27.
28.             //Er wordt door het Xml-bestand gezocht naar het juiste product.
29.             foreach (XmlNode in xmlDoc.SelectNodes("/Producten/Product"))
30.             {
31.                 if (xmlNode.SelectSingleNode("Naam").InnerText == prodNaam)
32.                 {
33.                     //Wanneer het juist product is gevonden worden de nieuwe
    waarden toegewezen.
34.                     xmlNode["Categorie"].InnerText = prodCategorie;
35.                     xmlNode["Aantal"].InnerText = prodAantal;
36.                     xmlNode["Bestaantal"].InnerText = prodBestAantal;
```



Figuur 10, blokschema dat het proces van het wijzigen van een product weergeeft

[1] *Zie Figuur 10 en Code 9*

```
37.                     xmlNode["Prijs"].InnerText = prodPrijs;
38.                     xmlNode["Korting"].InnerText = prodKorting;
39.                     xmlNode["NieuwePrijs"].InnerText = nieuwePrijsProd;
40.                 }
41.             }
42.             //Het Xmldocument en het bestand wordt opgeslagen en gesloten.
43.             file.Close();
44.             xmlDoc.Save(filePath);
45.         }
```

Code 9, C# code die weergeeft wat er gebeurt om een product aan te passen

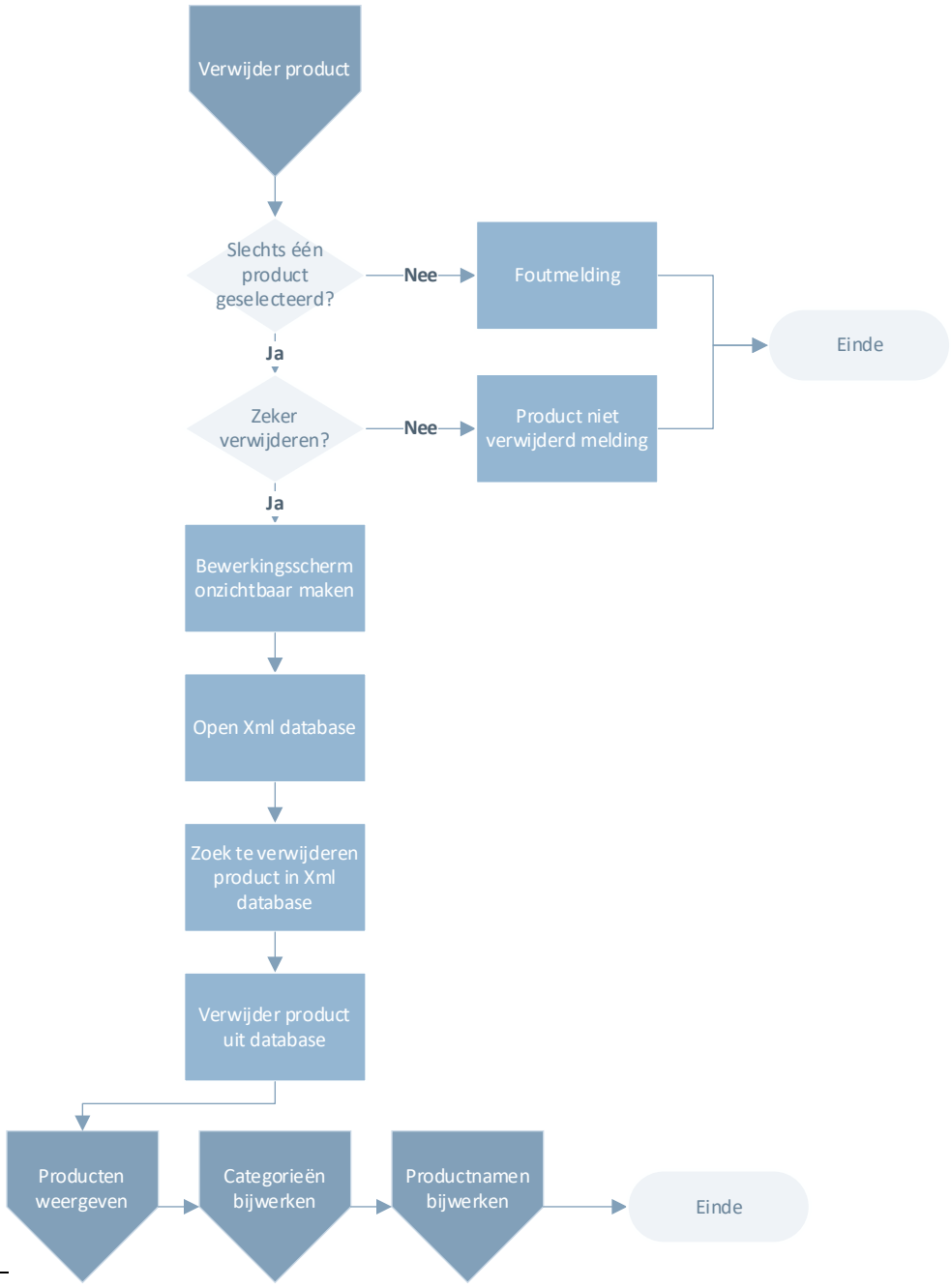
1.4.8 Product verwijderen

Om een product te verwijderen is er een method gemaakt in C#. Deze method verwijdert het geselecteerde product.

Vooraleer het product kan worden verwijderd, wordt er nagegaan of er slechts één product is geselecteerd. Hierna wordt er ook gevraagd of de gebruiker zeker is of dit product weldegelijk moet worden verwijderd.

Wanneer de gebruiker zeker is dat het product kan worden verwijderd, wordt de rijindex van het product opgevraagd. Hierna wordt het product met die index verwijderd uit de xml database.

Ten slot wordt het xml document afgesloten en wordt er gecontroleerd of er een productcategorie verwijderd moet worden [1].



[1] *Zie Figuur 11 en Code 10*

Figuur 11, blokschema dat visualiseert wat er gebeurt om een product te verwijderen

```
1.  private void DelProduct()
2.          {
3.              //Deze method verwijderd een bepaald product.
4.
5.
6.              //Er kan slechts één product tegelijkertijd worden verwijderd.
7.              if (lvProducten.SelectedItems.Count == 1)
8.              {
9.                  //Er wordt gevraagd of men zeker is of u het product wilt verwijderen.
10.
11.                 DialogResult result = MessageBox.Show(zekerMsg, "Verwijderen?",
    MessageBoxButtons.YesNo);
12.
13.                 if (result == DialogResult.Yes)
14.                 {
15.                     pnlProductEigenschappen.Visible = false;
16.
17.                     //De rijindex van een Xmldocument begint bij 1. Bij de listview is dat bij
    0.
18.                     int rowIndex = lvProducten.FocusedItem.Index + 1;
19.
20.                     //Het Xmldocument wordt geopend.
21.                     XmlDocument xmlDoc = new XmlDocument();
22.                     FileStream file = new FileStream(filePath, FileMode.Open);
23.                     xmlDoc.Load(file);
24.
25.                     //De XML-node waarin het product zich bevindt wordt geselecteerd en
    vervolgens verwijderd.
26.                     XmlNode = xmlDoc.SelectSingleNode($"/Producten/Product[{rowIndex}]");
27.                     xmlNode.ParentNode.RemoveChild(xmlNode);
28.
29.                     //Het Xmldocument en het bestand wordt opgeslagen en gesloten.
30.                     file.Close();
31.                     xmlDoc.Save(filePath);
32.                     FillListView();
33.
34.                     //De categorieën en namen worden bijgewerkt.
35.                     werkCategorieënBij();
36.                     werkNamenBij();
37.                 }
38.                 else
39.                 {
40.                     //Als men het product toch niet wil verwijderen, wordt dit ook duidelijk
    gemaakt en wordt alles teruggezet.
41.                     MessageBox.Show(abortMsg);
42.                     pnlProductEigenschappen.Visible = false;
43.                 }
44.             }
45.             else
46.             {
47.                 //Als er niet één maar meerdere of nul producten werden geselecteerd wordt er
    een foutmelding gegeven en wordt alles teruggezet.
48.                 MessageBox.Show(selectItemMsg);
49.                 pnlProductEigenschappen.Visible = false;
50.             }
51.         }
```

Code 10, C# code die weergeeft wat er gebeurt om een product te verwijderen

2 Kassa

2.1 Doel

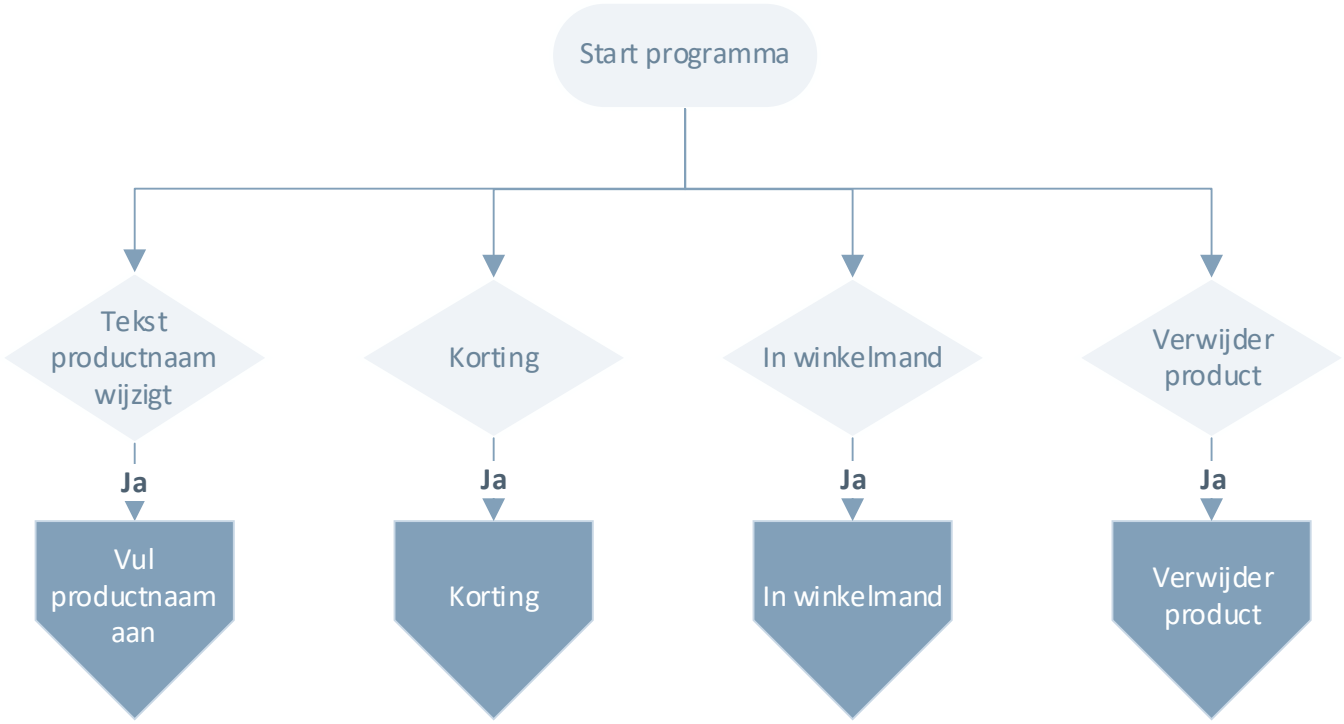
Het doel van de kassa is om een reeks producten in te kunnen voeren en vervolgens een rekening te kunnen maken.

Om dit te doen, moeten de producten geïdentificeerd worden. Hiervoor zijn er verschillende mogelijkheden, sommigen complexer dan anderen. In dit programma zullen de producten op naam geïdentificeerd worden. De naam van het product wordt ingevuld en vervolgens wordt het product aan een lijst toegevoegd.

Het is mogelijk dat de gebruiker meerdere keren hetzelfde product wil toevoegen, om dit te doen kan er met een teller het gewenste aantal geselecteerd worden.

Soms is het de bedoeling dat er korting kan worden aangerekend. Om dit te verwezenlijken kan er met een knop bepaald worden hoeveel procent korting er wordt aangerekend. Er kan ook voor worden gekozen om een zelfgekozen prijs aan te rekenen.

2.2 Blokschema



Figuur 12, overzichtsschema van het kassa programma

2.3 Werking

2.3.1 Product identificeren

2.3.1.1 Barcode

Een “Streepjescode of barcode is een opeenvolging van lijnen die een code representeert die door een scanner gelezen kan worden. Afhankelijk van het coderingssysteem kan deze code uitsluitend uit cijfers bestaan, of uit een combinatie van cijfers, letters en leestekens.” [1]

Dit is een goede manier om een product te identificeren. Een barcode kan de nodige hoeveelheid informatie opslaan. Bovendien zijn de meeste producten al voorzien van een barcode. Het nadeel van deze methode is dat het erg tijdrovend is om een programma te maken die een barcode kan inlezen.

2.3.1.2 QR-code

Een QR-code is qua werking erg vergelijkbaar met een barcode. Het is een efficiënte manier om informatie op te slaan en te lezen. Een QR-code wordt vaak gebruikt op flyers voor reclamedoeleinden. [2]

Het verschil met een barcode is dat een QR-code meer informatie kan opslaan maar over het algemeen minder wordt gebruikt in een winkel.

Een QR-code op ieder product plaatsen zou niet slim zijn, aangezien de meeste producten al een eigen barcode hebben. Bovendien kan een barcode de nodige informatie opslaan dus is het gebruik van een QR-code overbodig.

2.3.1.3 RFID

“Radio-frequency identification (identificatie met radiogolven, RFID) is een technologie om van een afstand informatie op te slaan in en af te lezen van zogenaamde RFID-tags die op of in objecten of levende wezens zitten, bijvoorbeeld in chipkaarten die gebruikmaken van Near field communication (NFC).” [3]

Om een product te identificeren zou het mogelijk zijn om in ieder product een RFID-tag te steken. Vervolgens kan de tag worden ingelezen door een tablet. Dit zou veiliger zijn dan met een barcode te werken. Door een RFID-tag in ieder product te steken zou het mogelijk zijn om vooraleer de klant de winkel verlaat te controleren of de producten wel werkelijk betaald werden.

Het nadeel van deze methode is dat het moeilijk te implementeren is. Bovendien is het best kostelijk om in ieder product een RFID-tag te steken.

---

[1] *(Wikipedia, 2022)*

[2] *(Timelessdesign, 2022)*

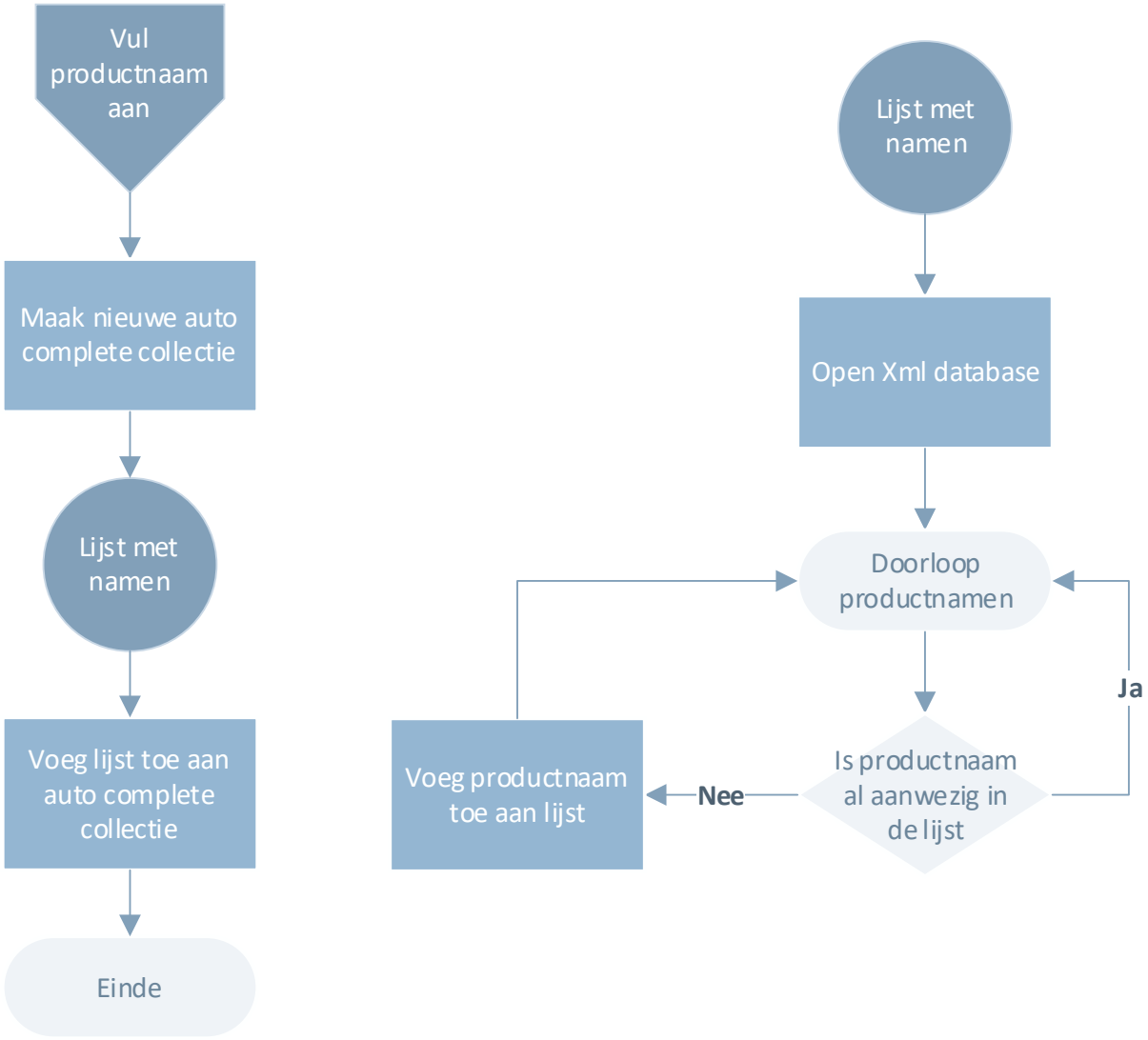
[3] *(Wikipedia, 2022)*

2.3.1.4 Productnaam intypen

Het is ook mogelijk om simpelweg de naam van het product in te typen. Bovendien is het mogelijk om met suggesties te werken zodra de gebruiker begint met typen. Hierdoor is het product snel gevonden. Een nadeel van deze methode is dat de gebruiker de naam van het product moet onthouden.

In dit programma is dit de methode die gebruikt wordt. Dit is een snelle en simpele manier om een product op te vragen.

Om het proces van productnamen in te typen te vergemakkelijken, is er gebruik gemaakt van de auto complete feature. Dit is een functie die een lijst met productnamen vraagt en die vergelijkt met de ingevulde tekst in zijn textbox. Als de tekst overeenkomt kan de gebruiker kiezen om dit product te selecteren en op die manier niet de hele productnaam in te typen. [1]



Figuur 13, blokschema dat het auto complete proces visualiseert

[1] *Zie Figuur 13, Code 11 en Code 12*

```
1.  void autocompleteTxbNaam()
2.          {
3.              AutoCompleteStringCollection autoSrc = new AutoCompleteStringCollection();
4.              autoSrc.AddRange(namenStrings());
5.
6.              txbHuidigProdNaam.AutoCompleteMode = AutoCompleteMode.Suggest;
7.              txbHuidigProdNaam.AutoCompleteSource = AutoCompleteSource.CustomSource;
8.              txbHuidigProdNaam.AutoCompleteCustomSource = autoSrc;
9.          }
10.
```

Code 11, C# code die weergeeft wat er gebeurt om auto complete toe te voegen aan een textbox.

```
1.  string[] namenStrings()
2.          {
3.              string[] namen = new string[0];
4.
5.              int count = 0;
6.
7.              XmlDocument xmlDoc = new XmlDocument();
8.              FileStream file = new FileStream(filePath, FileMode.Open);
9.              xmlDoc.Load(file);
10.
11.             XmlNodeList xmlNodeList = xmlDoc.SelectNodes("/Producten/Product/Naam");
12.
13.             foreach (XmlNode node in xmlNodeList)
14.             {
15.                 bool aanwezig = false;
16.                 foreach (string n in namen)
17.                 {
18.                     if (node.InnerText == n)
19.                     {
20.                         aanwezig = true;
21.                     }
22.                 }
23.
24.                 if (!aanwezig)
25.                 {
26.                     Array.Resize(ref namen, namen.Length + 1);
27.                     namen[count] = node.InnerText;
28.                     count++;
29.                 }
30.
31.             }
32.
33.             file.Close();
34.
35.             return namen;
36.
37.         }
38.
```

Code 12, C# code die weergeeft wat er gebeurt om de namen van alle producten in de database op te vragen,

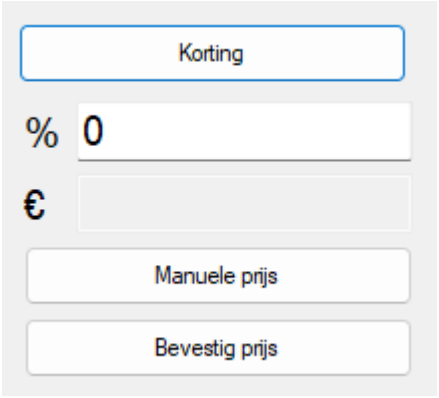
2.3.2 Korting

Om een bepaald product aan te rekenen met korting kan de gebruiker de kortingfunctie gebruiken.

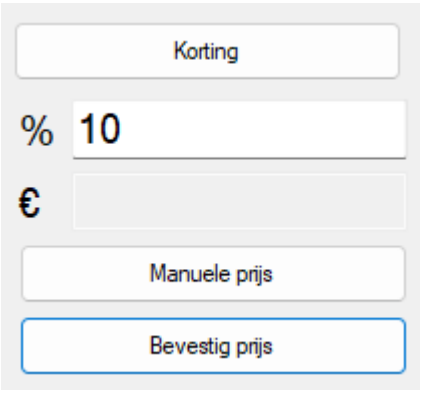
Om deze functie te gebruiken moet de gebruiker de procentuele korting of een eigen prijs ingeven. Vooraleer de korting wordt bepaald, moet het product al ingegeven zijn, vervolgens kan er op de knop “Korting” geklikt worden. Hierdoor komt een scherm tevoorschijn waarin de keuze kan worden gemaakt tussen procentuele korting of manuele prijs [1].

Als de gebruiker kiest voor procentuele korting kan de korting ingevuld worden in een daarvoor bestemde vak. Vervolgens moet er op de knop “Bevestigen” geklikt worden om de korting te bevestigen [2].

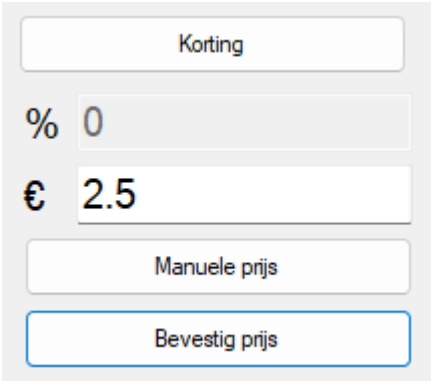
Als de gebruiker kiest om de prijs zelf in te geven moet de gebruiker klikken op de knop “Manuele prijs”. Vervolgens kan de gebruiker de gewenste prijs ingeven. Ten slot moet er op de knop “Bevestigen” geklikt worden om de korting te bevestigen [3].



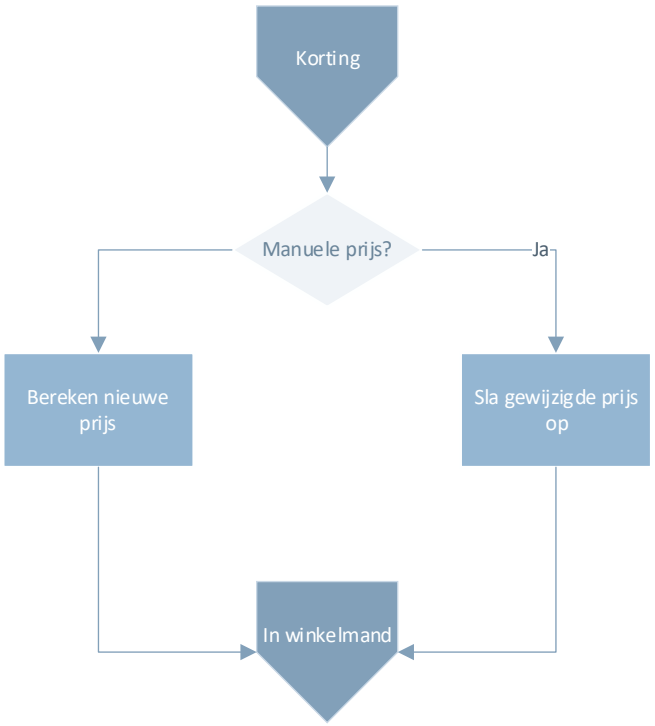
Figuur 15, afbeelding met voorbeeld van wat er gebeurt nadat er op de knop “Korting” wordt geklikt



Figuur 14, afbeelding met een voorbeeld van procentuele prijsopgave



Figuur 16, afbeelding met een voorbeeld van manuele prijsopgave



Figuur 17, blokschema dat het proces om korting toe te passen op een product weergeeft

[1] *Zie Figuur 15 en Figuur 17*

[2] *Zie Figuur 14 en Code 13*

[3] *Zie Figuur 16 en Code 13*

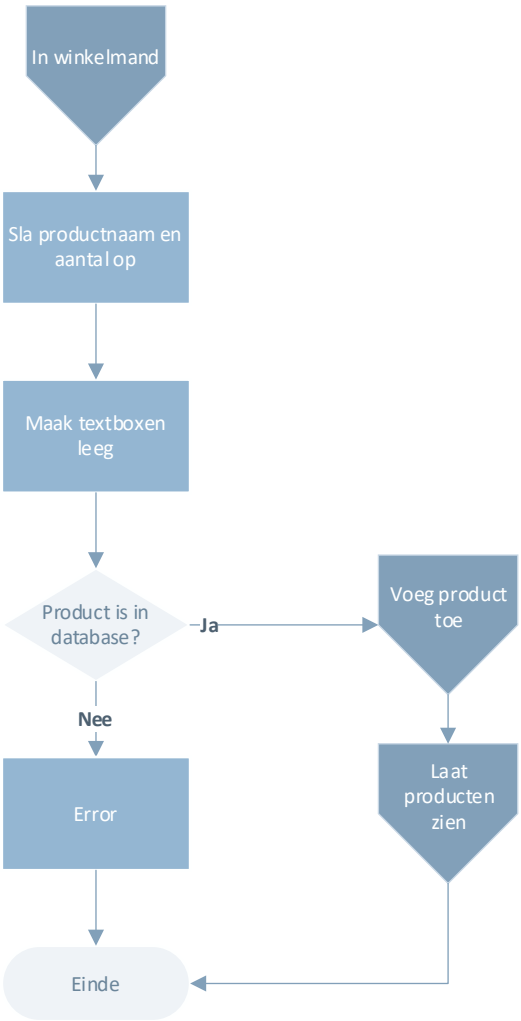
```
1.  void btnBevestigPrijs_Click(object sender, EventArgs e)
2.          {
3.              decimal prijs = 0;
4.              decimal aantal = nmudAantal.Value;
5.              string prodNaam = GetTxbData(txbHuidigProdNaam);
6.              int index = ZoekIndexInArray(prodNaam, productenArray());
7.
8.
9.              if (manueleprijs)
10.             {
11.                 try
12.                 {
13.                     prijs = Convert.ToDecimal(verander(GetTxbData(txbNieuwePrijs), '.', ","));
14.                 }
15.                 catch (Exception)
16.                 {
17.                     errorProv.SetError(txbNieuwePrijs, "De ingevulde prijs is ongeldig.");
18.                 }
19.                 txbNieuwePrijs.Enabled = !txbNieuwePrijs.Enabled;
20.                 txbKorting.Enabled = !txbKorting.Enabled;
21.
22.                 voegProductToe(index, prijs, aantal);
23.
24.                 manueleprijs = false;
25.             }
26.             else
27.             {
28.                 string oudePrijsString = productenArray()[index].Split(',')[1];
29.                 string kortingString = GetTxbData(txbKorting);
30.
31.                 decimal oudePrijs = Convert.ToDecimal(verander(oudePrijsString, '.', ","));
32.
33.                 decimal korting = 0;
34.
35.                 if (!kortingFout)
36.                 {
37.                     korting = decimal.Parse(kortingString);
38.                     prijs = kortingPrijs(oudePrijs, korting);
39.
40.                     voegProductToe(index, prijs, aantal);
41.
42.                     pnlKorting.Visible = false;
43.                     manueleprijs = false;
44.                     txbHuidigProdNaam.Text = "";
45.                     nmudAantal.Value = 1;
46.                 }
47.                 else
48.                 {
49.                     MessageBox.Show("De ingevulde korting is ongeldig.");
50.                 }
51.             }
52.
53.         }
54.
```

Code 13, C# code die weergeeft wat er gebeurt nadat er op de knop “Bevestig prijs” wordt geklikt

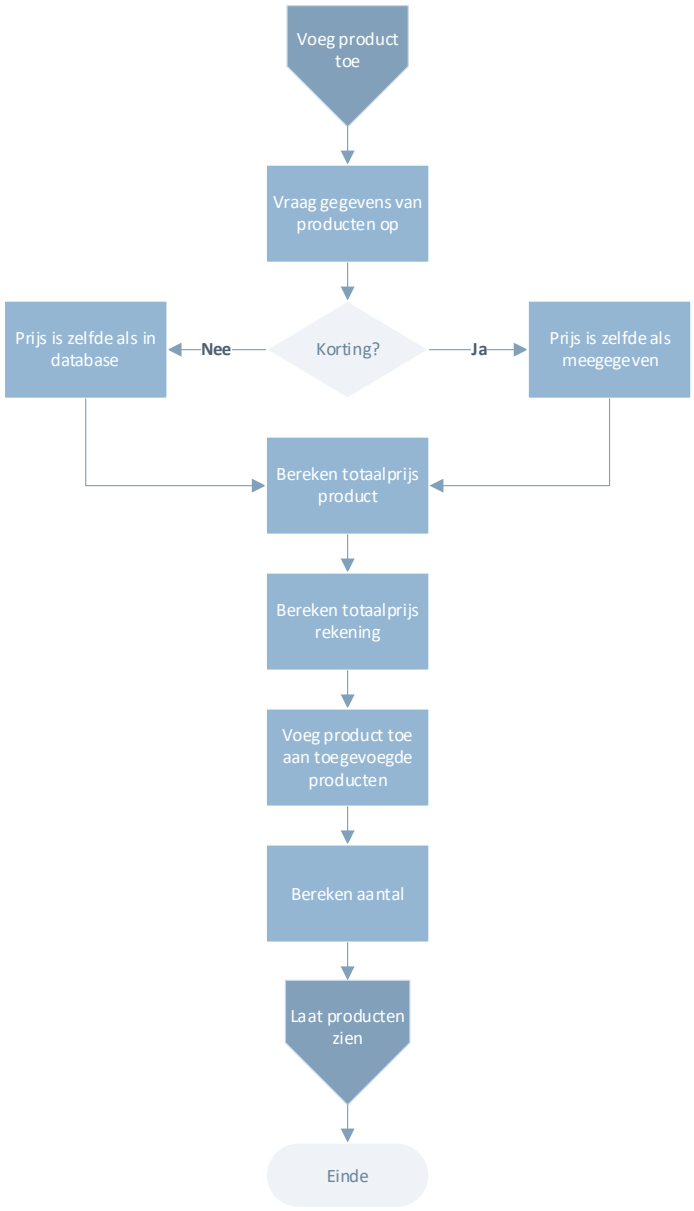
2.3.3 Product toevoegen

Nadat het product geïdentificeerd is en er eventueel korting is toegepast is het uiteraard de bedoeling dat het product kan worden toegevoegd aan de rekening. Om dit te doen moet er een proces zijn dat een product samen met zijn prijs kan toevoegen aan een lijst. Vervolgens moeten de producten uit die lijst opnieuw weergegeven worden [1].

Soms moet een product meerdere keren kunnen toegevoegd worden. Om dit te doen kan de gebruiker het aantal van een product selecteren. Vervolgens moet de prijs van het aantal producten berekend worden. Als deze prijs berekend is kan de prijs worden toegevoegd aan de tabel. Tot slot moet de tabel opnieuw worden geladen [2].



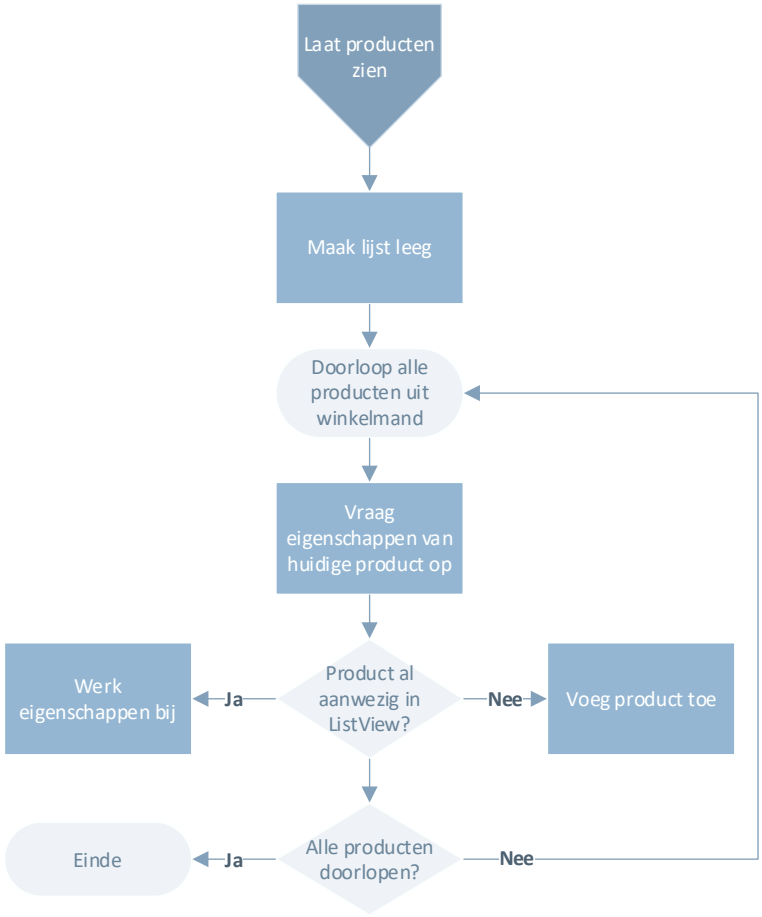
Figuur 19, blokdiagram van het proces nadat de gebruiker op de knop "In winkelmand" klikt



Figuur 18, blokdiagram van het proces waarin een product wordt toegevoegd

[1] *Zie Figuur 19, Code 14 en Code 15*

[2] *Zie Figuur 18, Figuur 20, Code 15 en Code 16*



Figuur 20, blokdiagram van het proces om de tabel van toegevoegde producten opnieuw te laden.

```
1.          private void btnVoegToe_Click(object sender, EventArgs e)
2.          {
3.              string productNaam = GetTxbData(txbHuidigProdNaam);
4.              decimal aantal = nmudAantal.Value;
5.
6.              maakTxbsLeeg();
7.
8.              if (productAanwezig(productNaam))
9.              {
10.                 voegProductToe(ZoekIndexInArray(productNaam, productenArray()), 0, aantal);
11.             }
12.             else
13.             {
14.                 errorProv.SetError(txbHuidigProdNaam, "Product niet gevonden.");
15.             }
16.         }
17.
```

Code 14, C# code die beschrijft wat er gebeurt nadat er op de knop "In winkelmand" wordt geklikt.

```
1.  private void voegProductToe(int productenArrayIndex, decimal prijs, decimal aantal)
2.          {
3.              string[] product = productenArray()[productenArrayIndex].Split(',');
4.              string productNaam = product[0];
5.              string productPrijs = "";
6.
7.              string aantalPrijs = "";
8.
9.              //Prijs is 0 als er geen korting is.
10.             if (prijs == 0)
11.             {
12.                 productPrijs = verander(product[1],',',".");
13.                 aantalPrijs = (Convert.ToDecimal(verander(productPrijs, '.', ",")) *
    aantal).ToString();
14.             }
15.             //Als er wel korting is
16.             else
17.             {
18.                 productPrijs = verander(prijs.ToString(),',',".");
19.                 aantalPrijs = (decimal.Parse(productPrijs) * aantal).ToString();
20.             }
21.
22.
23.             //Als er nog geen enkel product is toegevoegd is de totaalprijs leeg
24.             if (totaalPrijs != "")
25.             {
26.                 totaalPrijs = decimal.Add(Convert.ToDecimal(verander(aantalPrijs, '.', ",")),
27.                     Convert.ToDecimal(verander(totaalPrijs, '.', ","))).ToString();
28.             }
29.             else
30.             {
31.                 totaalPrijs = aantalPrijs;
32.             }
33.
34.             Array.Resize(ref productenToegevoegd, productenToegevoegd.Length + 1);
35.             productenToegevoegd[productenToegevoegd.Length - 1] = productNaam;
36.
37.             //Product nog nooit gekocht
38.             if (!alGekocht(productNaam))
39.             {
40.                 Array.Resize(ref winkelmandje, winkelmandje.Length + 1);
41.                 winkelmandje[winkelmandje.Length - 1] =
    $"{productNaam},{productPrijs},{aantal}";
42.
43.                 herlaadListView();
44.             }
45.
```

```
            //Product wel al eens gekocht
46.         else
47.         {
48.             int index = ZoekIndexInArray(productNaam, winkelmandje);
49.             int vorigAantal = Convert.ToInt32(winkelmandje[index].Split(',')[2]);
50.             int aantalGekocht = vorigAantal + Convert.ToInt32(aantal);
51.
52.             winkelmandje[index] = $"{productNaam},{productPrijs},{aantalGekocht}";
53.
54.             herlaadListView();
55.         }
56.
57.         totaalPrijs = berekentotaalPrijs().ToString();
58.         lblTotPrijs.Text = totaalPrijs;
59.
60.         pnlKorting.Visible = false;
61.         manueleprijs = false;
62.         txbHuidigProdNaam.Text = "";
63.         nmudAantal.Value = 1;
64.     }
65.
```

Code 15, C# code die beschrijft wat er gebeurt om een product toe te voegen aan winkelmand lijst

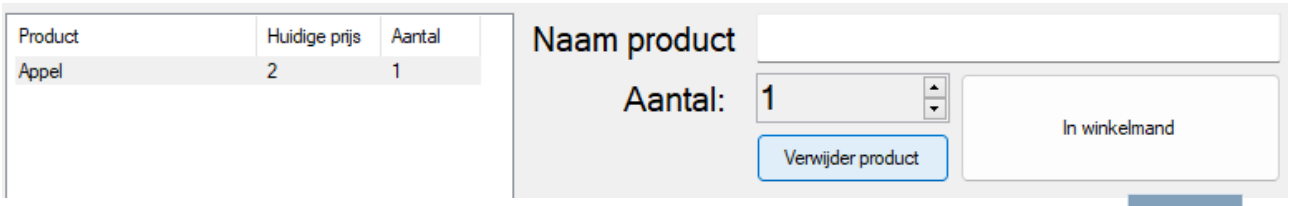
```
1.  void herlaadListView()
2.      {
3.          lvProducten.Items.Clear();
4.
5.          foreach (string product in winkelmandje)
6.          {
7.              bool toegevoegd = false;
8.
9.              string[] eigenschappen = product.Split(',');
10.
11.             string productNaam = eigenschappen[0];
12.             string productPrijs = eigenschappen[1];
13.             string aantal = eigenschappen[2];
14.
15.             foreach (ListViewItem n in lvProducten.Items)
16.             {
17.                 //Als product al aanwezig in lvProducten moeten de eigenschappen van het
    product aangepast worden
18.                 if (n.Text == productNaam)
19.                 {
20.                     n.SubItems[1].Text = verander(productPrijs,',',".");
21.                     n.SubItems[2].Text = aantal;
22.                     toegevoegd = true;
23.                 }
24.             }
25.             //Als het product nog niet werd toegevoegd moet het worden toegevoegd
26.             if (!toegevoegd)
27.             {
28.                 ListViewItem lvItem = new ListViewItem(productNaam);
29.                 lvItem.SubItems.Add(verander(productPrijs, ',', "."));
30.                 lvItem.SubItems.Add(aantal);
31.
32.                 lvProducten.Items.Add(lvItem);
33.                 toegevoegd = true;
34.             }
35.         }
36.     }
37.
```

Code 16, C# code die het proces beschrijft waarmee de tabel opnieuw wordt gevuld

2.3.4 Product verwijderen

Als de gebruiker een product wil verwijderen kan dit door op de knop “Verwijderen” te klikken. Hiervoor moet er één product geselecteerd zijn in de tabel [1].

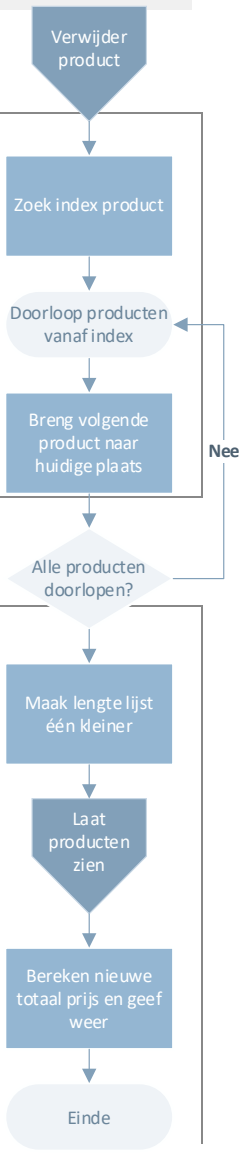
Een product verwijderen gebeurt door eerst de index van het product in de tabel te zoeken. Vervolgens wordt er gezocht naar de index in de winkelmandlijst. Hier wordt het product ook verwijderd en worden de achtervolgende producten naar voor gehaald. Hierna wordt de lengte van de lijst verminderd met één en de tabel opnieuw geladen [2].



Figuur 22, afbeelding waarop het verwijderen van een product wordt weergegeven

```
1.  private void btnVerwijderProd_Click(object sender, EventArgs e)
2.          {
3.              if (lvProducten.SelectedItems.Count == 1)
4.              {
5.                  int index = lvProducten.SelectedItems[0].Index;
6.                  verwijderProd(index);
7.              }
8.              else
9.              {
10.                 MessageBox.Show("Selecteer één product.");
11.             }
12.
13.         }
14.
```

Code 17, C# code die beschrijft wat er gebeurt nadat er op de knop “Verwijder product” geklikt wordt.



Figuur 21, blokschema dat het proces om een product te verwijderen weergeeft

```
1.          void verwijderProd(int index)
2.          {
3.              string prijs = winkelmandje[index].Split(',')[1];
4.              double prijsProduct = Convert.ToDouble(verander(prijs, ',', "."));
5.
6.              int aantal = Convert.ToInt32(winkelmandje[index].Split(',')[2]);
7.
8.              winkelmandje[index] = "";
9.              for (int i = index; i < winkelmandje.Length - 1; i++)
10.             {
11.                 winkelmandje[i] = winkelmandje[i + 1];
12.             }
13.             Array.Resize(ref winkelmandje, winkelmandje.Length - 1);
14.
15.             herlaadListView();
16.
17.             totaalPrijs = berekentotaalPrijs().ToString();
18.             lblTotPrijs.Text = totaalPrijs;
19.         }
20.         double berekentotaalPrijs()
```

[1] *Zie Figuur 22*

[2] *Zie Figuur 21, Code 17 en Code 18*

```
21.         {
22.             double totaalPrijs = 0;
23.
24.             foreach(string artikel in winkelmandje)
25.             {
26.                 string[] eigenschappen = artikel.Split(',');
27.
28.                 double aantal = Convert.ToDouble(eigenschappen[2]);
29.                 double prijs = Convert.ToDouble(verander(eigenschappen[1],'.',","));
30.                 double nieuw = prijs * aantal;
31.
32.                 totaalPrijs += nieuw;
33.             }
34.
35.             return totaalPrijs;
36.         }
37.
```

Code 18, C# code die het proces om een product te verwijderen weergeeft

3 Foutcontrole

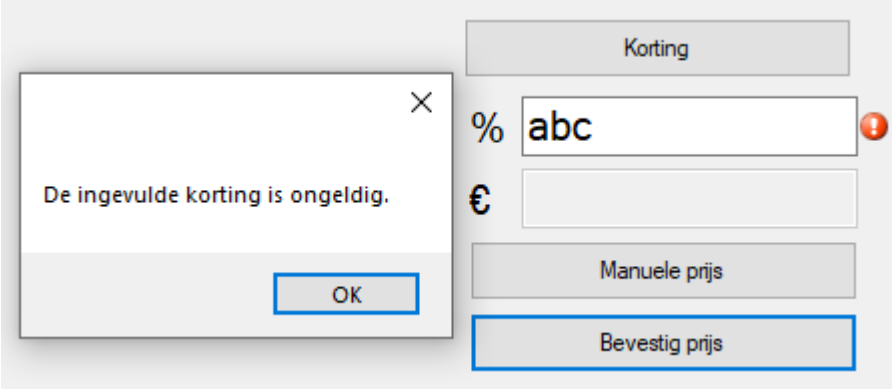
Dit programma werkt enkel omdat er ook foutcontrole is. Bij alle gegevens die worden ingegeven moet er worden gecontroleerd of de waardes wel mogelijk zijn. Als er bijvoorbeeld een korting van “abc” procent wordt gegeven moet er uiteraard een foutmelding worden gegeven [1].

Om de foutcontrole vlot te laten verlopen is er een c# method geschreven waarin er per invoer methode wordt gecontroleerd of de waarden reëel zijn.

De foutcontrole verschilt per programma, de foutcontrole van het database programma is uitgebreider aangezien er meer invoervelden zijn [2]. Ook is er bij het database programma een auto complete functie bij de categorieën en moet er bij de productnaam worden gecontroleerd of de naam al gebruikt werd [3].

Bij het kassa programma is de foutcontrole vergelijkbaar met die van de database, de method ControleerTxb werd grotendeels overgenomen [4].

Een probleem bij het ingeven van de prijs is dat sommige gebruikers liever een ‘,’ gebruiken voor een komma getal en andere gebruikers liever een ‘.’ gebruiken. Om dit probleem op te lossen is er een method gemaakt die een stuk tekst vraagt en zoekt naar een komma of punt. Dit wordt dan veranderd in een punt of komma, naargelang wat er wordt gevraagd [5].



Figuur 23, afbeelding die foutmelding weergeeft van een foute korting

[1] *Zie Figuur 23*

[2] *Zie Code 19*

[3] *Zie Code 20*

[4] *Zie Code 21*

[5] *Zie Code 22*

```
1.  private string GetTxbData(TextBox txb)
2.          {
3.              //De string geeft de tekst van een opgevraagde textbox terug.
4.
5.              //De string wordt aangemaakt.
6.              string inhoud = "";
7.              try
8.              {
9.                  //Er wordt gezocht naar de textbox. Als die is gevonden wordt zijn tekstwaarde
    in de string inhoud geplaatst.
10.                 inhoud = txb.Text;
11.             }
12.             catch (Exception ex)
13.             {
14.                 //Als de textbox niet werd gevonden wordt er een fout gestuurd.
15.                 errorProv.SetError(txb, ex.Message);
16.             }
17.             //De inhoud van de textbox wordt doorgestuurd.
18.             return inhoud;
19.         }
20.
21.         private void ControleerTxb(TextBox txb)
22.         {
23.             //Deze method dient om te controleren of de ingegeven tekst van de textbox een
    reële waarde is.
24.
25.             //Eerst wordt de tekst van de textbox opgevraagd.
26.             string txbText = GetTxbData(txb);
27.
28.             //Als de textbox txbNaam is wordt dit uitgevoerd.
29.             if (txb.Name == "txbNaam")
30.             {
31.                 //Er wordt altijd vanuit gegaan dat er een fout is. Als alles juist is wordt er
    pas vanuit gegaan dat het juist is.
32.                 naamFout = true;
33.
34.                 //Als de textbox niet enkel letters bevat of leeg is wordt er een fout
    aangegeven.
35.                 if (!txbText.All(char.IsLetter) || string.IsNullOrEmpty(txbText))
36.                 {
37.                     errorProv.SetError(txb, foutenMsg);
38.                 }
39.
40.                 //Als er nog geen fout was én de naam al eens gebruikt werd en het een nieuw
    product is, wordt er een fout gegeven (Het gaat dus over een nieuw product).
41.                 else if (naamAlAanwezig(txbText) && newProd)
42.                 {
43.                     errorProv.SetError(txb, naamAanwezigMsg);
44.                 }
45.                 //Als er nog geen fout was én de naam al eens gebruikt werd maar al een oud
    product is, wordt er geen fout gegeven (Het product werd dus gewijzigd).
46.                 else if (naamAlAanwezig(txbText) && !newProd && txbText != huidigeNaam)
47.                 {
48.                     errorProv.SetError(txb, naamAanwezigMsg);
49.                 }
50.                 else
51.                 {
52.                     //Als men er zeker van is dat er geen enkele fout is wordt de errorprovider
    leeg gemaakt en is er geen fout.
53.                     errorProv.SetError(txb, "");
54.                     naamFout = false;
55.                 }
56.             }
57.
58.             //Als de textbox txbCategorie is wordt dit uitgevoerd.
59.             else if (txb.Name == "txbCategorie")
60.             {
61.                 //Er wordt altijd vanuit gegaan dat er een fout is. Als alles juist is wordt er
    pas vanuit gegaan dat het juist is.
62.                 categorieFout = true;
```

```
63.
64.                 //Als de textbox niet enkel letters bevat of leeg is wordt er een fout
    aangegeven.
65.                 if (!txbText.All(char.IsLetter) || string.IsNullOrEmpty(txbText))
66.                 {
67.                     errorProv.SetError(txb, foutenMsg);
68.                 }
69.                 else
70.                 {
71.                     //Als men er zeker van is dat er geen enkele fout is wordt de errorprovider
    leeg gemaakt en is er geen fout.
72.                     errorProv.SetError(txb, "");
73.                     categorieFout = false;
74.                 }
75.             }
76.             //Als de textbox txbAantalAanwezig is wordt dit uitgevoerd.
77.             else if (txb.Name == "txbAantalAanwezig")
78.             {
79.                 //Tijdelijke bool om te kijken of er een fout is.
80.                 bool tijdelijkFout = false;
81.
82.                 //Er wordt altijd vanuit gegaan dat er een fout is. Als alles juist is wordt er
    pas vanuit gegaan dat het juist is.
83.                 aantalAanwezigFout = true;
84.
85.                 //Als de textbox niet enkel letters bevat of leeg is wordt er een fout
    aangegeven.
86.                 if (!txbText.All(char.IsNumber) || string.IsNullOrEmpty(txbText))
87.                 {
88.                     errorProv.SetError(txb, foutenMsg);
89.                     tijdelijkFout = true;
90.                 }
91.                 //Wordt altijd uitgevoerd.
92.                 else if (aantalAanwezigFout)
93.                 {
94.                     try
95.                     {
96.                         //Als getal kleiner is dan 0 moet er een foutmelding komen.
97.                         if (int.Parse(txbText) <= 0)
98.                         {
99.                             errorProv.SetError(txb, "Getal moet groter dan 0 zijn.");
100.                              tijdelijkFout = true;
101.                         }
102.                         //Als getal een kommagetal is dan moet er een foutmelding komen.
103.                         if (int.Parse(txbText) % 1 != 0)
104.                         {
105.                             errorProv.SetError(txb, "Getal mag geen kommagetal zijn.");
106.                             tijdelijkFout = true;
107.                         }
108.                     }
109.                     catch (Exception)
110.                     {
111.                         //Als er fout is met Int.Parse is er waarschijnlijk een letter in
    txbText. Er moet dus een foutmelding worden weergegeven.
112.                         errorProv.SetError(txb, "De inhoud van deze textbox moet een getal
    zijn.");
113.                         tijdelijkFout = true;
114.                     }
115.                 }
116.                 if (!tijdelijkFout)
117.                 {
118.                     //Als men er zeker van is dat er geen enkele fout is wordt de
    errorprovider leeg gemaakt en is er geen fout.
119.                     errorProv.SetError(txb, "");
120.                     aantalAanwezigFout = false;
121.                 }
122.             }
123.             //Als de textbox txbAantalBestAanwezig is wordt dit uitgevoerd.
124.             else if (txb.Name == "txbAantalBestAanwezig")
125.             {
126.                 //Tijdelijke bool om te kijken of er een fout is.
```

```
127.                 bool tijdelijkFout = false;
128.
129.                 //Er wordt altijd vanuit gegaan dat er een fout is. Als alles juist is wordt
    er pas vanuit gegaan dat het juist is.
130.                 aantalBestAanwezigFout = true;
131.
132.                 //Als de textbox niet enkel letters bevat of leeg is wordt er een fout
    aangegeven.
133.                 if (!txbText.All(char.IsNumber) || string.IsNullOrEmpty(txbText))
134.                 {
135.                     errorProv.SetError(txb, foutenMsg);
136.                     tijdelijkFout = true;
137.                 }
138.                 //Wordt altijd uitgevoerd.
139.                 else if (aantalBestAanwezigFout)
140.                 {
141.                     try
142.                     {
143.                         //Als getal kleiner is dan 0 moet er een foutmelding komen.
144.                         if (int.Parse(txbText) <= 0)
145.                         {
146.                             errorProv.SetError(txb, "Getal moet groter dan 0 zijn.");
147.                             tijdelijkFout = true;
148.                         }
149.                         //Als getal een kommagetal is dan moet er een foutmelding komen.
150.                         if (int.Parse(txbText) % 1 != 0)
151.                         {
152.                             errorProv.SetError(txb, "Getal mag geen kommagetal zijn.");
153.                             tijdelijkFout = true;
154.                         }
155.                     }
156.                     catch (Exception)
157.                     {
158.                         //Als er fout is met Int.Parse is er waarschijnlijk een letter in
    txbText. Er moet dus een foutmelding worden weergegeven.
159.                         errorProv.SetError(txb, "De inhoud van deze textbox moet een getal
    zijn.");
160.                         tijdelijkFout = true;
161.                     }
162.                 }
163.                 if (!tijdelijkFout)
164.                 {
165.                     //Als men er zeker van is dat er geen enkele fout is wordt de
    errorprovider leeg gemaakt en is er geen fout.
166.                     errorProv.SetError(txb, "");
167.                     aantalBestAanwezigFout = false;
168.                 }
169.             }
170.             //Als de textbox txbPrijs is wordt dit uitgevoerd.
171.             else if (txb.Name == "txbPrijs")
172.             {
173.                 //Tijdelijke bool om te kijken of er een fout is.
174.                 bool tijdelijkFout = false;
175.
176.                 //Er wordt altijd vanuit gegaan dat er een fout is. Als alles juist is wordt
    er pas vanuit gegaan dat het juist is.
177.                 prijsFout = true;
178.
179.                 //Als de textbox niet enkel letters bevat of leeg is wordt er een fout
    aangegeven.
180.                 if (!double.TryParse(txbText, out double n) || string.IsNullOrEmpty(txbText))
181.                 {
182.                     errorProv.SetError(txb, foutenMsg);
183.                     tijdelijkFout = true;
184.                 }
185.                 //Wordt altijd uitgevoerd.
186.                 else if (prijsFout)
187.                 {
188.                     try
189.                     {
190.                         //Als getal kleiner is dan 0 moet er een foutmelding komen.
```

```
191.                         if (double.Parse(txbText) <= 0)
192.                         {
193.                             errorProv.SetError(txb, "Getal moet groter dan 0 zijn.");
194.                             tijdelijkFout = true;
195.                         }
196.                     }
197.                     catch (Exception)
198.                     {
199.                         //Als er fout is met Int.Parse is er waarschijnlijk een letter in
    txbText. Er moet dus een foutmelding worden weergegeven.
200.                         errorProv.SetError(txb, "De inhoud van deze textbox moet een getal
    zijn.");
201.                         tijdelijkFout = true;
202.                     }
203.                 }
204.                 if (!tijdelijkFout)
205.                 {
206.                     //Als men er zeker van is dat er geen enkele fout is wordt de
    errorprovider leeg gemaakt en is er geen fout.
207.                     errorProv.SetError(txb, "");
208.                     prijsFout = false;
209.                 }
210.             }
211.             //Als de textbox txbKorting is wordt dit uitgevoerd.
212.             else if (txb.Name == "txbKorting")
213.             {
214.                 //Tijdelijke bool om te kijken of er een fout is.
215.                 bool tijdelijkFout = false;
216.
217.                 //Er wordt altijd vanuit gegaan dat er een fout is. Als alles juist is wordt
    er pas vanuit gegaan dat het juist is.
218.                 kortingFout = true;
219.
220.                 //Als de textbox niet enkel letters bevat of leeg is wordt er een fout
    aangegeven.
221.                 if (!txbText.All(char.IsNumber) || string.IsNullOrEmpty(txbText))
222.                 {
223.                     errorProv.SetError(txb, foutenMsg);
224.                     tijdelijkFout = true;
225.                 }
226.                 //Wordt altijd uitgevoerd.
227.                 else if (kortingFout)
228.                 {
229.                     try
230.                     {
231.                         //Als getal kleiner is dan 0 moet er een foutmelding komen.
232.                         if (int.Parse(txbText) < 0)
233.                         {
234.                             errorProv.SetError(txb, "Getal mag niet negatief zijn.");
235.                             tijdelijkFout = true;
236.                         }
237.                         //Als getal een kommagetal is dan moet er een foutmelding komen.
238.                         if (int.Parse(txbText) % 1 != 0)
239.                         {
240.                             errorProv.SetError(txb, "Getal mag geen kommagetal zijn.");
241.                             tijdelijkFout = true;
242.                         }
243.                         if (int.Parse(txbText) >= 100)
244.                         {
245.                             errorProv.SetError(txb, "Getal moet kleiner dan 100 zijn.");
246.                             tijdelijkFout = true;
247.                         }
248.                     }
249.                     catch (Exception)
250.                     {
251.                         //Als er fout is met Int.Parse is er waarschijnlijk een letter in
    txbText. Er moet dus een foutmelding worden weergegeven.
252.                         errorProv.SetError(txb, "De inhoud van deze textbox moet een getal
    zijn.");
253.                         tijdelijkFout = true;
254.                     }
```

```
255.                 }
256.                 if (!tijdelijkFout)
257.                 {
258.                     //Als men er zeker van is dat er geen enkele fout is wordt de
    errorprovider leeg gemaakt en is er geen fout.
259.                     errorProv.SetError(txb, "");
260.                     kortingFout = false;
261.                 }
262.             }
263.
264.         }
265.
266.         private bool naamAlAanwezig(string naam)
267.         {
268.             bool aanwezig = false;
269.
270.             foreach (string n in namen)
271.             {
272.                 if (n == naam)
273.                 {
274.                     aanwezig = true;
275.                 }
276.             }
277.             if (aanwezig && newProd)
278.             {
279.                 naamFout = true;
280.             }
281.
282.             return aanwezig;
283.         }
284.
285.         private bool GeenErrors()
286.         {
287.             bool geenErrors = true;
288.
289.             if (naamFout || categorieFout || aantalAanwezigFout || aantalBestAanwezigFout ||
    prijsFout || kortingFout)
290.             {
291.                 geenErrors = false;
292.             }
293.
294.             return geenErrors;
295.         }
296.
297.         private int aantalAanwezigXml(string teZoeken)
298.         {
299.             int aantalAanwezig = 0;
300.
301.             XmlDocument xmlDoc = new XmlDocument();
302.             FileStream file = new FileStream(filePath, FileMode.Open);
303.             xmlDoc.Load(file);
304.
305.             XmlNodeList xmlNodeList = xmlDoc.SelectNodes("/Producten/Product");
306.
307.             foreach (XmlNode xmlNode in xmlNodeList)
308.             {
309.                 string prodNaam = xmlNode["Naam"].InnerText;
310.                 string prodCategorie = xmlNode["Categorie"].InnerText;
311.                 string prodAantal = xmlNode["Aantal"].InnerText;
312.                 string prodBestAantal = xmlNode["Bestaantal"].InnerText;
313.
314.                 if (prodNaam == teZoeken)
315.                 {
316.                     aantalAanwezig++;
317.                 }
318.
319.                 if (prodCategorie == teZoeken)
320.                 {
321.                     aantalAanwezig++;
322.                 }
323.                 //prodAantal en prodBestAantal kunnen dezelfde zijn!!
```

```
324.                 if (prodAantal == teZoeken)
325.                 {
326.                     aantalAanwezig++;
327.                 }
328.
329.                 if (prodBestAantal == teZoeken)
330.                 {
331.                     aantalAanwezig++;
332.                 }
333.             }
334.
335.             file.Close();
336.
337.             return aantalAanwezig;
338.         }
339.
```

Code 19, C# code die weergeeft wat er gebeurt om de invoer van een textbox in het database programma te controleren

```
1.          private string[] wijzigCategorieën()
2.          {
3.              string[] categorieën = new string[0];
4.
5.              int count = 0;
6.
7.              XmlDocument xmlDoc = new XmlDocument();
8.              FileStream file = new FileStream(filePath, FileMode.Open);
9.              xmlDoc.Load(file);
10.
11.             XmlNodeList xmlNodeList = xmlDoc.SelectNodes("/Producten/Product/Categorie");
12.
13.             foreach (XmlNode node in xmlNodeList)
14.             {
15.                 bool aanwezig = false;
16.                 foreach (string n in categorieën)
17.                 {
18.                     if (node.InnerText == n)
19.                     {
20.                         aanwezig = true;
21.                     }
22.                 }
23.                 if (!aanwezig)
24.                 {
25.                     Array.Resize(ref categorieën, categorieën.Length + 1);
26.                     categorieën[count] = node.InnerText;
27.                     count++;
28.                 }
29.
30.             }
31.             file.Close();
32.
33.             return categorieën;
34.         }
35.
36.         private void werkCategorieënBij()
37.         {
38.             Array.Resize(ref categorieën, wijzigCategorieën().Length);
39.             categorieën = wijzigCategorieën();
40.         }
41.
42.         private void txbCategorie_TextChanged(object sender, EventArgs e)
43.         {
44.             autocompleteTxbCat();
45.         }
46.
47.         private void autocompleteTxbCat()
48.         {
49.             AutoCompleteStringCollection autoSrc = new AutoCompleteStringCollection();
50.             autoSrc.AddRange(categorieën);
51.
52.             txbCategorie.AutoCompleteMode = AutoCompleteMode.Suggest;
53.             txbCategorie.AutoCompleteSource = AutoCompleteSource.CustomSource;
54.             txbCategorie.AutoCompleteCustomSource = autoSrc;
55.         }
56.
```

Code 20, C# code die het proces van de auto complete functie verduidelijkt

```
1.          public void ControleerTxb(TextBox txb)
2.          {
3.              //Deze method dient om te controleren of de ingegeven tekst van de textbox een
    reële waarde is.
4.
5.              //Eerst wordt de tekst van de textbox opgevraagd.
6.              string txbText = txb.Text;
7.
8.              //Als de textbox txbNaam is wordt dit uitgevoerd.
9.              if (txb.Name == "txbHuidigProdNaam")
10.             {
11.                 //Er wordt altijd vanuit gegaan dat er een fout is. Als alles juist is wordt er
    pas vanuit gegaan dat het juist is.
12.                 naamFout = true;
13.
14.                 //Als de textbox niet enkel letters bevat of leeg is wordt er een fout
    aangegeven.
15.                 if (!txbText.All(char.IsLetter) || string.IsNullOrEmpty(txbText))
16.                 {
17.                     errorProv.SetError(txb, foutenMsg);
18.                 }
19.                 else
20.                 {
21.                     //Als men er zeker van is dat er geen enkele fout is wordt de errorprovider
    leeg gemaakt en is er geen fout.
22.                     errorProv.SetError(txb, "");
23.                     naamFout = false;
24.                 }
25.             }
26.
27.
28.             //Als de textbox txbKorting is wordt dit uitgevoerd.
29.             else if (txb.Name == "txbKorting")
30.             {
31.                 //Tijdelijke bool om te kijken of er een fout is.
32.                 bool tijdelijkFout = false;
33.
34.                 //Er wordt altijd vanuit gegaan dat er een fout is. Als alles juist is wordt er
    pas vanuit gegaan dat het juist is.
35.                 kortingFout = true;
36.
37.                 //Als de textbox niet enkel letters bevat of leeg is wordt er een fout
    aangegeven.
38.                 if (!txbText.All(char.IsNumber) || string.IsNullOrEmpty(txbText))
39.                 {
40.                     errorProv.SetError(txb, foutenMsg);
41.                     tijdelijkFout = true;
42.                 }
43.                 //Wordt altijd uitgevoerd.
44.                 else if (kortingFout)
45.                 {
46.                     try
47.                     {
48.                         //Als getal kleiner is dan 0 moet er een foutmelding komen.
49.                         if (int.Parse(txbText) < 0)
50.                         {
51.                             errorProv.SetError(txb, "Getal mag niet negatief zijn.");
52.                             tijdelijkFout = true;
53.                         }
54.                         //Als getal een kommagetal is dan moet er een foutmelding komen.
55.                         if (int.Parse(txbText) % 1 != 0)
56.                         {
57.                             errorProv.SetError(txb, "Getal mag geen kommagetal zijn.");
58.                             tijdelijkFout = true;
59.                         }
60.                         if (int.Parse(txbText) >= 100)
61.                         {
62.                             errorProv.SetError(txb, "Getal moet kleiner dan 100 zijn.");
63.                             tijdelijkFout = true;
64.                         }
```

```
65.                     }
66.                     catch (Exception)
67.                     {
68.                         //Als er fout is met Int.Parse is er waarschijnlijk een letter in
    txbText. Er moet dus een foutmelding worden weergegeven.
69.                         errorProv.SetError(txb, "De inhoud van deze textbox moet een getal
    zijn.");
70.                         tijdelijkFout = true;
71.                     }
72.                 }
73.                 if (!tijdelijkFout)
74.                 {
75.                     //Als men er zeker van is dat er geen enkele fout is wordt de errorprovider
    leeg gemaakt en is er geen fout.
76.                     errorProv.SetError(txb, "");
77.                     kortingFout = false;
78.                 }
79.             }
80.
81.             //Als de textbox txbNieuwePrijs is wordt dit uitgevoerd.
82.             else if (txb.Name == "txbNieuwePrijs")
83.             {
84.                 //Tijdelijke bool om te kijken of er een fout is.
85.                 bool tijdelijkFout = false;
86.
87.                 //Er wordt altijd vanuit gegaan dat er een fout is. Als alles juist is wordt er
    pas vanuit gegaan dat het juist is.
88.                 prijsFout = true;
89.
90.                 //Als de textbox niet enkel cijfers bevat of leeg is wordt er een fout
    aangegeven.
91.                 if (!double.TryParse(txbText, out double d) || string.IsNullOrEmpty(txbText))
92.                 {
93.                     errorProv.SetError(txb, foutenMsg);
94.                     tijdelijkFout = true;
95.                 }
96.                 //Wordt altijd uitgevoerd.
97.                 else if (prijsFout)
98.                 {
99.                     try
100.                      {
101.                          //Als getal kleiner is dan 0 moet er een foutmelding komen.
102.                          if (double.Parse(txbText) <= 0)
103.                          {
104.                              errorProv.SetError(txb, "Getal moet groter dan 0 zijn.");
105.                              tijdelijkFout = true;
106.                          }
107.                      }
108.                      catch (Exception)
109.                      {
110.                          //Als er fout is met Int.Parse is er waarschijnlijk een letter in
    txbText. Er moet dus een foutmelding worden weergegeven.
111.                          errorProv.SetError(txb, "De inhoud van deze textbox moet een getal
    zijn.");
112.                          tijdelijkFout = true;
113.                      }
114.                  }
115.                  if (!tijdelijkFout)
116.                  {
117.                      //Als men er zeker van is dat er geen enkele fout is wordt de
    errorprovider leeg gemaakt en is er geen fout.
118.                      errorProv.SetError(txb, "");
119.                      prijsFout = false;
120.                  }
121.              }
122.          }
123.
124.          string GetTxbData(TextBox txb)
125.          {
126.              //De string geeft de tekst van een opgevraagde textbox terug.
127.
```

```
128.             //De string wordt aangemaakt.
129.             string inhoud = "";
130.             try
131.             {
132.                 //Er wordt gezocht naar de textbox. Als die is gevonden wordt zijn
    tekstwaarde in de string inhoud geplaatst.
133.                 inhoud = txb.Text;
134.             }
135.             catch (Exception ex)
136.             {
137.                 //Als de textbox niet werd gevonden wordt er een fout gestuurd.
138.                 errorProv.SetError(txb, ex.Message);
139.             }
140.
141.             //De inhoud van de textbox wordt doorgestuurd.
142.             ControleerTxb(txb);
143.
144.             if (txb.Name == "txbHuidigProdNaam")
145.             {
146.                 if (!naamFout)
147.                 {
148.                     return inhoud;
149.                 }
150.                 else
151.                 {
152.                     return "Fout!";
153.                 }
154.             }
155.             else if (txb.Name == "txbKorting")
156.             {
157.                 if (!kortingFout)
158.                 {
159.                     return inhoud;
160.                 }
161.                 else
162.                 {
163.                     return "Fout!";
164.                 }
165.             }
166.             else if (txb.Name == "txbNieuwePrijs")
167.             {
168.                 if (!prijsFout)
169.                 {
170.                     return inhoud;
171.                 }
172.                 else
173.                 {
174.                     return "Fout!";
175.                 }
176.             }
177.             else
178.             {
179.                 return "Textbox niet gevonden!";
180.             }
181.         }
182.
```

Code 21, C# code die weergeeft wat er gebeurt om de invoer van een textbox in het kassa programma te controleren

```
1.  private string verander(string getal, char oorspronkelijk, string nieuw)
2.          {
3.              string output = "";
4.
5.              foreach (char n in getal)
6.              {
7.                  if (n == oorspronkelijk)
8.                  {
9.                      output += nieuw;
10.                 }
11.                 else
12.                 {
13.                     output += n;
14.                 }
15.             }
16.
17.             return output;
18.         }
19.
```

Code 22, C# code die weergeeft wat er gebeurt om de komma van een getal in tekstvorm aan te passen

4 Besluit

Toen ik dit project koos, wist ik meteen dat het niet makkelijk zou zijn en er veel uitdagingen op mijn pad zouden komen. Ik ben daarom enorm trots dat ik er in ben geslaagd (om met veel vallen en opstaan), dit project te verwezenlijken. Er waren uiteraard veel dingen die ik achteraf gezien beter had kunnen doen.

Bij het begin van dit project wist ik dat ik op een manier een database moest maken en kunnen bewerken in mijn programma. Ik ben de eerste twee weken veel te lang bezig geweest met het instellen van SQL-server. Dit is natuurlijk de meer bekende software hiervoor, maar eigenlijk was dit veel te complex voor mijn project. Ik vind het jammer dat ik daarmee zoveel tijd ben verloren. Wanneer ik had gekozen om xml te gebruiken kwam ik op nieuwe problemen: het was veel complexer dan ik eerst had gehoopt om een xml-bestand te maken en te lezen in een programma. Naarmate mijn programma complexer werd en er meer functies werden toegevoegd, werd het alsmaar moeilijker om het overzicht te bewaren. Ik was dus beter eerst begonnen met een blokschema. Ook was het goed geweest als ik mijn code vanaf het begin al had gedocumenteerd, zodat ik meteen kon zien wat mijn code deed als ik dit vergat.

Als er meer tijd was geweest had ik het enorm tof gevonden om een product te kunnen scannen via een barcode. Dat had mijn werk veel interessanter en duidelijker gemaakt. Het is erg jammer dat mijn werk enkel een computerprogramma is.

Ik heb erg veel bijgeleerd en het was enorm leerzaam om dit programma volledig zelf te maken. Doordat ik niet veel hulp heb ingeschakeld bij de uitwerking van dit project heb ik geleerd om problemen efficiënt op te zoeken en op te lossen. Ik heb bij dit werk ook geleerd hoe je je code ordelijk en efficiënt kan bijhouden: ik heb GitHub gebruikt voor het beheren van mijn code en om deadlines bij te houden. Via deze link is mijn project beschikbaar om in te kijken: https://github.com/ziasvann/GIP-WinkelOnderhoudSoftware. Ondanks sommige tegenslagen ben ik zeer tevreden over mijn geleverde werk. Ik heb veel geleerd uit dit project en ik denk dat ik veel geleerde dingen in mijn latere leven als IT'er nog zal kunnen gebruiken.

5 Bronnen/literatuurlijst

Bestandinfo. (2022, 05 28). *XML bestand - hoe te openen xml file?* Opgehaald van Bestandinfo: https://bestand.info/extension/xml

Timelessdesign. (2022, 05 28). *Wat is een QR code en hoe werkt een QR code?* Opgehaald van Timelessdesign: https://www.timelessdesign.nl/blog/wat-is-een-qr-code-en-hoe-werkt-een-qr-code

Wikipedia. (2022, 05 16). *Radio-frequency identification*. Opgeroepen op 05 28, 2022, van Wikipedia: https://nl.wikipedia.org/wiki/Radio-frequency_identification

Wikipedia. (2022, 01 20). *Streepjescode*. Opgeroepen op 05 28, 2022, van Wikipedia: https://nl.wikipedia.org/wiki/Streepjescode

Figuurtabel

Codetabel