|  |
| --- |
| Thomas More campus De Nayer |
| Wireshell |
| Practice Enterprise 1 |

|  |
| --- |
| Stijn Geerts & Jeroen Vereecken  2018 - 2019  Electronica-ICT Application Development Fase 1 |

# Voorwoord:

We hebben Wireshell ontwikkeld voor het vak Practice Enterprise 1.

In dit verslag zal je meer lezen over de opbouw van het project en hoe we uiteindelijk aan het eindresultaat zijn geraakt inclusief alle problemen die daarbij horen. Daarnaast zal er meer uitleg in staan over de belangrijkste stukken code van elk onderdeel van het project (de applicatie, de website en de databank).

Stijn Geerts: R0697679

Jeroen Vereecken: R0699140

Inhoud

[Voorwoord: 2](#_Toc8637502)

[Projectvoorstel 1: 6](#_Toc8637503)

[Inleiding: 6](#_Toc8637504)

[Welk probleem lost deze innovatie op? 6](#_Toc8637505)

[Wat is de behoefte die erachter schuilgaat? 6](#_Toc8637506)

[Wie zijn de gebruikers? 6](#_Toc8637507)

[Wat zijn de trends & ontwikkelingen binnen dit vakgebied/technologie? 6](#_Toc8637508)

[Problemen: 7](#_Toc8637509)

[Tijdverdeling: 8](#_Toc8637510)

[Projectvoorstel 2: 10](#_Toc8637511)

[Inleiding: 10](#_Toc8637512)

[Welk probleem lost deze innovatie op? 10](#_Toc8637513)

[Wat is de behoefte die erachter schuilgaat? 10](#_Toc8637514)

[Wie zijn de gebruikers? 10](#_Toc8637515)

[Blokschema: 10](#_Toc8637516)

[Plan van aanpak: 11](#_Toc8637517)

[Doel: 11](#_Toc8637518)

[Ontwerpspecificaties: 11](#_Toc8637519)

[Functionele analyse: 11](#_Toc8637520)

[Stappen: 11](#_Toc8637521)

[Blokschema: 12](#_Toc8637522)

[Databank Diagram: 13](#_Toc8637523)

[Opgeslagen gegevens: 13](#_Toc8637524)

[Talen: 15](#_Toc8637525)

[Website: 15](#_Toc8637526)

[Applicatie: 15](#_Toc8637527)

[Belangrijke Code: 16](#_Toc8637528)

[MySQL: 16](#_Toc8637529)

[Uitleg SQL-code: 16](#_Toc8637530)

[C#: 17](#_Toc8637531)

[Uitleg C# code: 19](#_Toc8637532)

[PHP: 20](#_Toc8637533)

[Uitleg PHP Code: 21](#_Toc8637534)

[Poster: 22](#_Toc8637535)

# Projectvoorstel 1:

## Inleiding:

Onze innovatie is een website waar de gebruiker een account op kan maken. En dan gaat de website zoeken wat er over de gebruiker te vinden is op bepaalde websites zoals Google, Facebook, etc. Wat er dan gevonden is wordt opgeslagen op het profiel van de gebruiker.

## Welk probleem lost deze innovatie op?

Veel mensen weten tegenwoordig niet meer dat er heel veel informatie over hun te vinden is op het internet. Via onze innovatie willen we de mensen duidelijk maken dat er veel informatie te vinden is en dat dit wel eens gevaarlijk kan zijn wanneer dit allemaal openbaar staat.

## Wat is de behoefte die erachter schuilgaat?

Sinds dit jaar is er een nieuwe privacy regulatie de GDPR genaamd. We willen via de website aantonen dat dit gevaarlijk kan zijn en zo de mensen helpen hun persoonlijke gegevens beter te beschermen. Onze innovatie biedt daarom een makkelijkere manier om gegevens te vinden en ze daarna zelf te gaan beschermen. Bijvoorbeeld oudere mensen weten niet goed hoe het allemaal moet en zij moeten evenzeer de kans krijgen om hun gegevens van de buitenwereld te beschermen.

## Wie zijn de gebruikers?

Zoals hierboven reeds uitgelegd zijn de gebruikers vooral mensen die het wat moeilijker hebben met het begrijpen van het internet en hoe ze zichzelf beter kunnen beschermen. Het is ook voor de mensen die niet goed weten waarom het zo gevaarlijk is als je gegevens openbaar beschikbaar zijn.

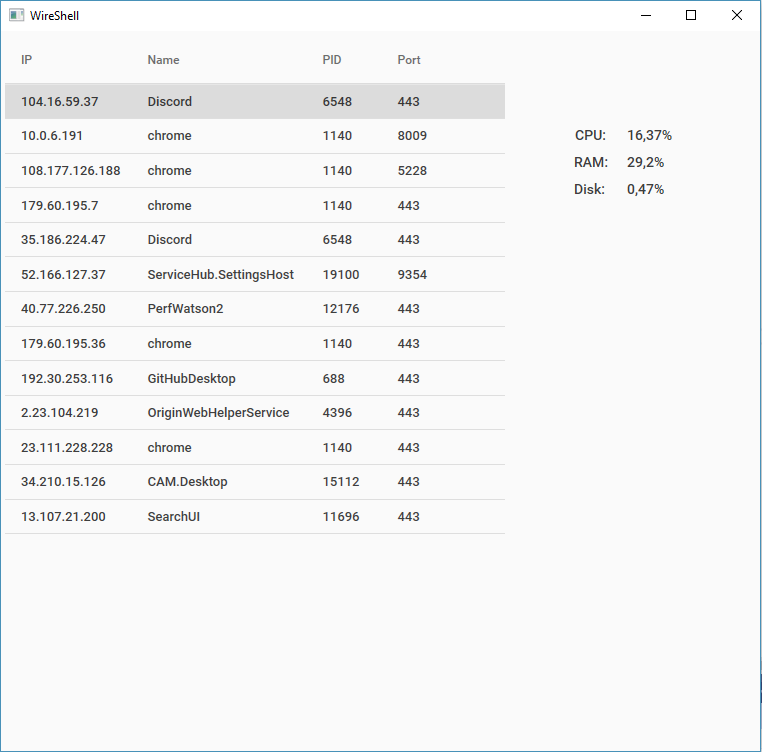
## Wat zijn de trends & ontwikkelingen binnen dit vakgebied/technologie?

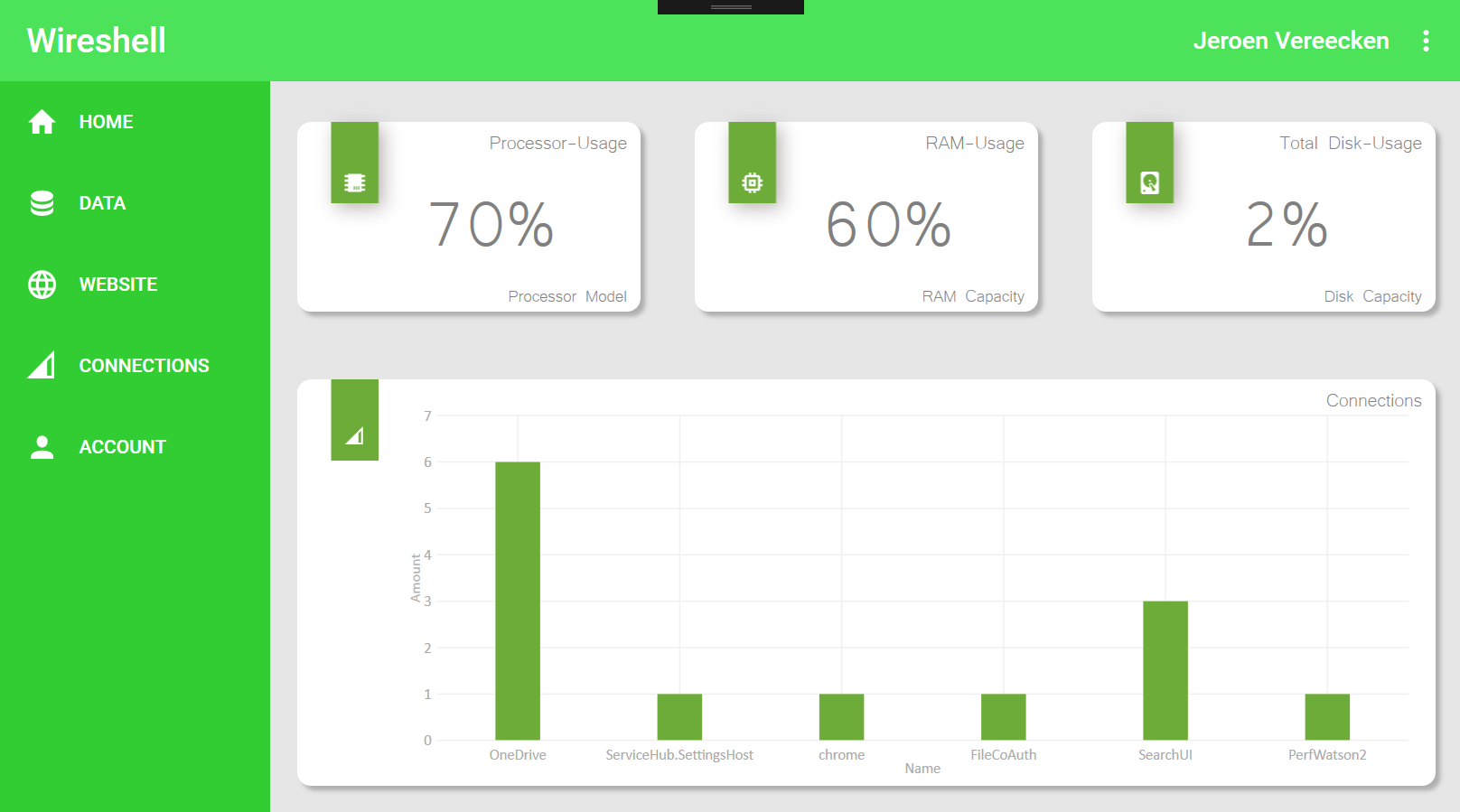
Er bestaat al een website dat eigenlijk laat zien wat er van jezelf op het internet te vinden is. Wij hebben deze bekeken en hebben daaruit de conclusie getrokken dat deze website niet echt doet wat je zou verwachten, bovendien wordt niet volledig weergegeven wat er allemaal over een persoon te vinden is.

Daarnaast is er het laatste jaar heel wat rond privacy gebeurd. Europa heeft onder andere de GDPR “General Data Protection Regulation” ingevoerd. Daarnaast is er ook nog het grote Facebook schandaal geweest waar er heel veel gegevens gelekt zijn.

# Problemen:

We hadden eerst een ander project voorgesteld zoals je kan zien bovenaan het verslag bij projectvoorstel 1. We waren direct begonnen met het uitwerken van dit project maar na een aantal weken eraan te werken zijn we tot de conclusie gekomen dat de API van facebook geblokkeerd was voor de gegevens die we nodig hadden. Om toegang tot deze gegevens te verkrijgen moesten we een aanvraag indienen. Om te voldoen aan de eisen van de aanvraag moest het project volledig klaar zijn in verband met hoe de gebruiker het te zien zou krijgen. En als het dan aan alles voldoet ging facebook beslissen of we toegang kregen of niet. Er was ons verteld om te proberen een email te sturen naar facebook en uit te leggen waarom we toegang nodig hadden tot deze gegevens en dat we studenten zijn van Thomas More campus De Nayer. Maar er was geen email adres te vinden van facebook. Dus hebben we besloten een heel nieuw project te maken. Om te voorkomen dat we opnieuw een project zouden kiezen dat we niet konden realiseren hebben we eerst een goede twee weken nagedacht over een nieuw project en een daarbij horende Proof of concept gemaakt. Zodat we zeker waren dat dit keer het project mogelijk was. Hieronder kan je onze Proof of concept zien en daaronder zie je tot wat onze Proof of concept geëvolueerd is.

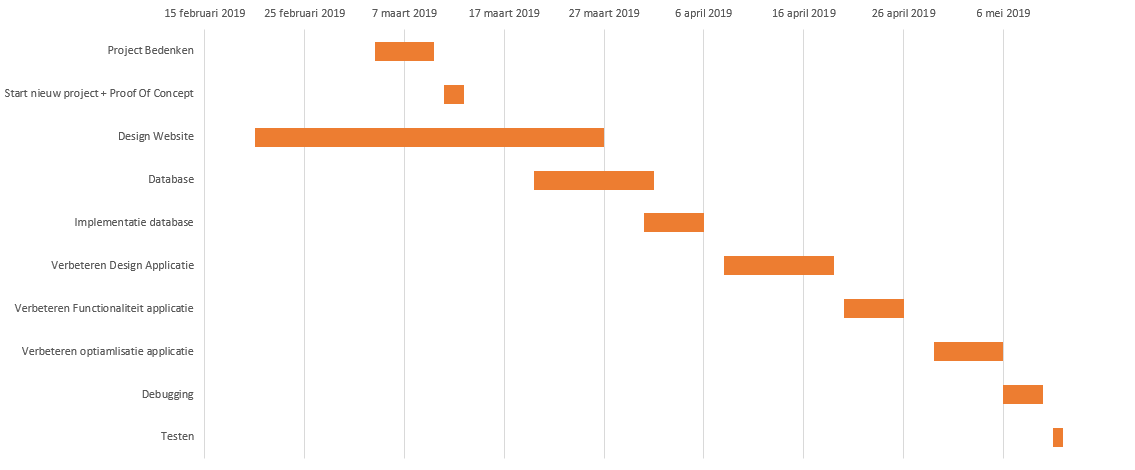




Om zo weinig mogelijk dubbel werk te doen hebben we gedacht aan een project waarbij we het werk van het vorige project konden hergebruiken. We hadden al een stuk van de huisstijl gemaakt voor de website met het vorige project dus hebben we dat stuk hergebruikt in het huidige project inclusief de kleuren.

## Tijdverdeling:

We hebben onze Gantt Chart moeten aanpassen omdat we zoals hierboven uitgelegd opnieuw zijn moeten beginnen. Je kan erop zien dat we het design van het vorige project hergebruikt hebben. Toch heeft het design veel langer geduurd dan verwacht omdat onze bedoeling was om de app en de website er zo gebruiksvriendelijk mogelijk uit te laten zien. Daarom heeft dit zeer veel tijd in beslag genomen. Ook omdat XAML de designtaal dat gebruikt wordt in WPF iets volledig nieuw was voor ons dus heeft het design van de app ongeveer twee weken geduurd om het te ontwikkelen. Met het bedenken van een nieuw project zijn we ook wel wat tijd verloren waardoor we uiteindelijk later klaar zijn geraakt dan verwacht en hebben de laatste weken van het project enorm veel aandacht vereist om het klaar te krijgen.



# Projectvoorstel 2:

## Inleiding:

Ons project is een website en een Windows applicatie die in verbinding staan met elkaar via een MySQL server. De applicatie gaat via het CMD commando Netstat alle huidige TCP verbindingen tonen exclusief de lokale verbindingen en de verbinding met de MySQL server. Met deze applicatie kan de gebruiker zich beter informeren over alle huidige verbindingen. En kan hij/zij zich ook beter beschermen tegen schadelijke verbindingen. Alle informatie wordt gebruiksvriendelijk voorgesteld in een lijst en in een grafiek.

## Welk probleem lost deze innovatie op?

Deze innovatie geeft een duidelijk overzicht in de vorm van een lijst en in de vorm van een grafiek met wat de computer momenteel verbonden is. Er bestaan gelijkaardige applicaties maar uit ervaring word je er meestal overspoeld met overbodige gegevens en te veel informatie waardoor het niet zo gebruiksvriendelijk is voor de gebruiker.

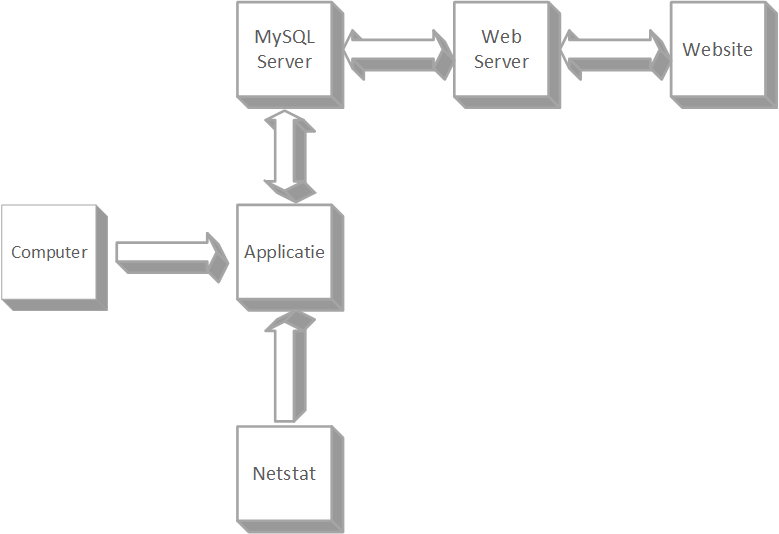
## Wat is de behoefte die erachter schuilgaat?

Veel mensen weten niet met wat de computer allemaal verbonden is. Ze gaan het ook niet uitzoeken omdat de meeste mensen bijvoorbeeld niet weten wat Netstat in CMD is of doet. De meeste applicaties die dit vereenvoudigen voor de gebruiker zijn betalend en geven de gebruiker zoveel informatie dat het voor veel mensen gewoon niet meer duidelijk is.

## Wie zijn de gebruikers?

Zoals hierboven reeds uitgelegd zijn de gebruikers vooral mensen die niet weten hoe ze controleren met wat de computer verbonden is en welke van deze verbindingen mogelijk schade kunnen opleveren. En gebruikers dat vinden dat de betalende applicaties niet goed genoeg werken voor de prijs.

## Blokschema:



# Plan van aanpak:

## Doel:

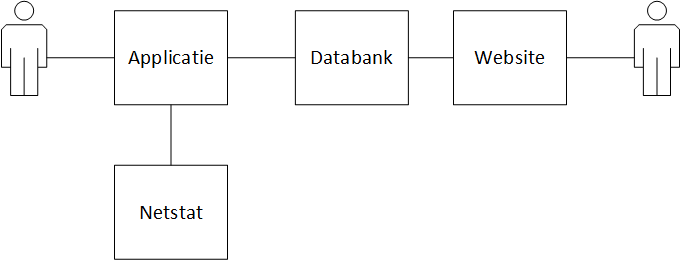
Een gebruiksvriendelijke applicatie maken waarop de gebruiker live kan zien welke verbindingen er allemaal gemaakt zijn met de computer. Hij/zij kan ook het processorverbruik, het RAM-verbruik en het opslagverbruik bekijken. Deze applicatie staat dan in verbinding met een overzichtelijke website via een databank. Op deze website kan de gebruiker een historiek zien van zijn verbindingen van het laatste half uur.

## Ontwerpspecificaties:

We gebruiken als webserver XAML wat volledig gratis is. Hiermee kan de website communiceren met de databank. De databank hebben we op een NAS draaien bij Stijn thuis waardoor we ook geen lokale databank moeten gebruiken.

Kostprijs: 0 euro

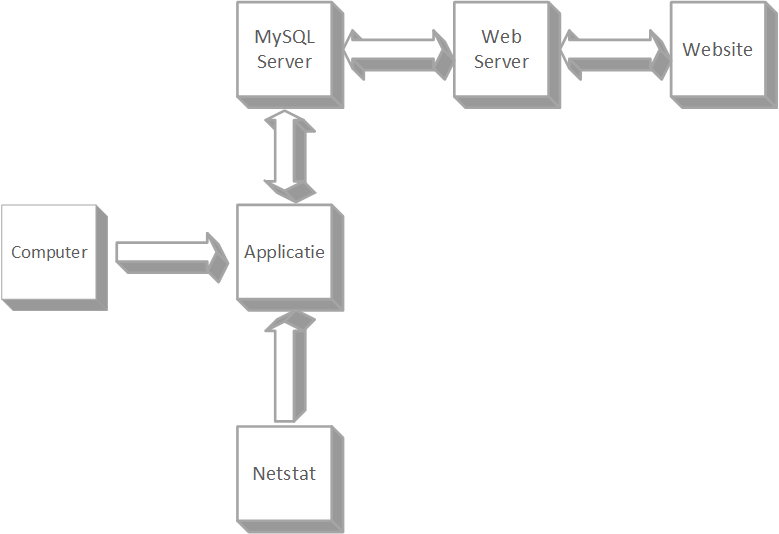
## Functionele analyse:



### Stappen:

1. De gebruiker opent de applicatie, de applicatie zal het Netstat commando starten en de data doorsturen naar de databank.
2. De databank zal de data van de applicatie aankrijgen en verwerken.
   1. De databank is in twee richtingen verbonden met de applicatie en de website voor de login.
3. De website zal de data ophalen van de databank en in het juiste formaat tonen aan de gebruiker.

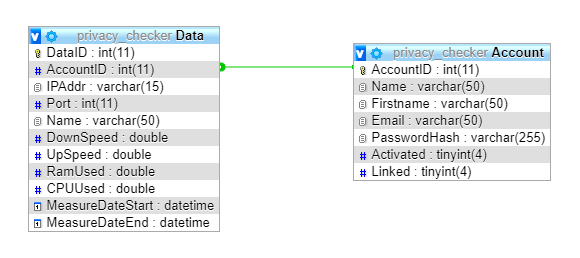
# Blokschema:



De gebruiker opent via de computer de applicatie. Die zal op zijn beurt de commando Netstat starten. Netstat zal de gegevens doorsturen naar de applicatie. De applicatie zal deze gegevens doorsturen naar de databank waarmee hij in verbinding staat. De databank zal deze data verwerken. De website zal dan op zijn beurt de data van de databank halen om deze data op een overzichtelijke manier te tonen aan de gebruiker. De webserver dient om de communicatie tussen de website en de databank mogelijk te maken.

De databank is in twee richtingen verbonden met de web server en met de applicatie. De reden hiervoor is dat je op de applicatie en de website kan inloggen. Verdere uitleg over waarom de databank in twee richtingen verbonden is vind je bij de belangrijke code.

# Databank Diagram:



## Opgeslagen gegevens:

* AccountID
  + Is een uniek getal dat de account aanduidt.
* Name (Account)
  + De naam van de aangemaakte gebruiker. Dit houden we bij zodat we de gebruiker kunnen aanspreken bij zijn naam.
* FirstName
  + De voornaam van de aangemaakte gebruiker.
* Email
  + Dit is het email adres waarop de gebruiker zijn account heeft geregistreerd. Dit wordt bijgehouden om de activatie mail te versturen en om in te kunnen loggen.
* PasswordHash
  + Dit is het wachtwoord van de gebruiker maar geëncrypteerd zodat als er een data lek zou komen dat de wachtwoorden niet bruikbaar zijn.
* Activated
  + Dit wordt bijgehouden om te controleren of de gebruiker zijn account geactiveerd heeft via de bijhorende activatiemail.
* Linked
  + Deze variabele hebben we aangemaakt ter uitbreiding. Stel Linked staat op TRUE dan gaat de applicatie anders werken en voor de proces naam de naam van het apparaat plaatsen.
* DataID
  + Dit is een uniek getal dat elke datalijn aanduidt.
* IPAddr
  + Dit is het IP address van het bijhorende process.
* Port
  + Dit is de Port van het bijhorende process.
* Name (Data)
  + Dit is de Naam van het process bijvoorbeeld (Chrome).
* DownSpeed
  + Dit houden we bij ter uitbreiding om te kunnen tonen hoeveel elk process individueel heeft gedownload.
* UpSpeed
  + Dit houden we bij ter uitbreiding om te kunnen tonen hoeveel elk process individueel heeft verstuurd.
* RamUsed
  + Hiermee tonen we op de website hoeveel elk process van het RAM geheugen verbruikt.
* CPUUsed
  + Hiermee tonen we op de website hoeveel elk process van de CPU verbruikt.
* MeasureDateStart
  + Hiermee kunnen we zien wanneer het process gestart is.
* MeasureDateEnd
  + Hiermee kunnen we zien wanneer het process beëindigd is. Als dit NULL is in de databank betekent dit dat het process in de databank nog actief is.

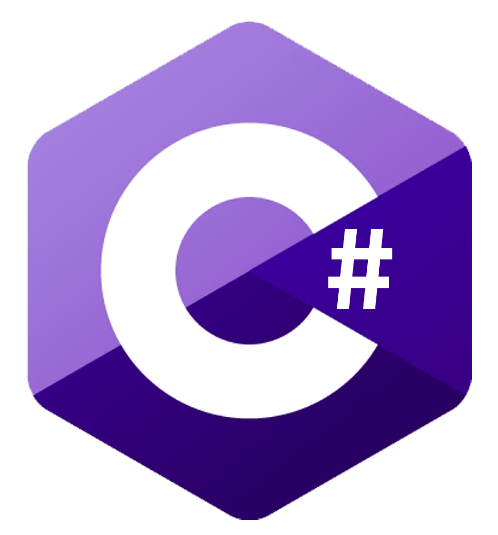
# Talen:

## Website:

* PHP7
* HTML5
* CSS3
* Jquery
* Javascript
* MySQL (MariaDB 10)

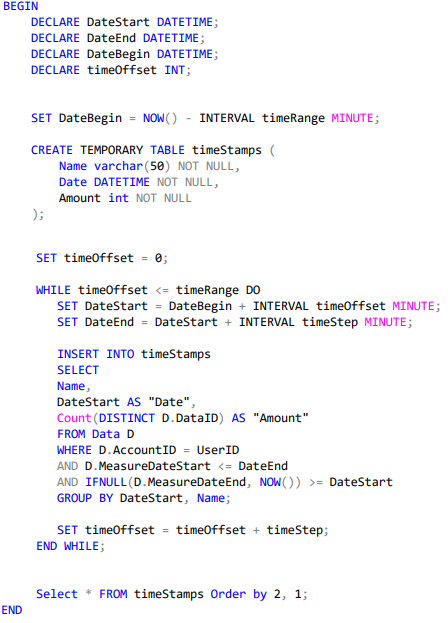
## Applicatie:

* C#
* MySQL (MariaDB 10)
* XAML



# Belangrijke Code:

## MySQL:



## Uitleg SQL-code:

Dit stuk code wordt gebruikt om de data van de databank te tonen in de grafieken op de website. Dit stuk code zal bij het laden van de pagina de verbindingen van het laatste half uur nemen met een step van drie minuten. Na de pagina geladen is zal deze query gestart worden om de minuut met een range van 1 minuut en een step van 1 minuut.

## C#:

1. **private** **static** **void** OutputHandler(**object** sendingProcess, DataReceivedEventArgs outLine)
2. {
3. //Als de nieuwe lijn leeg is zal er niets worden gedaan
4. **if** (String.IsNullOrEmpty(outLine.Data))
5. **return**;
7. **try**
8. {
9. //Controleert wanneer netstat opnieuw controleert naar connecties
10. **if**(outLine.Data.Equals("Active Connections"))
11. {
12. //Console.WriteLine("----- new list -----\n old list length: " + currentDataList.Count);
13. **try**
14. {
15. **for**(**int** i = activeDataList.Count-1; i >= 0; i--)
16. {
17. //Console.Write(i + " " + activeDataList[i].Name + " \t ------\t");
18. //activeDataList[i].WriteData();
20. ProcessData currentData = activeDataList[i];
21. **bool** connected = **false**;
23. **foreach** (ProcessData oldData **in** currentDataList)
24. {
25. //Console.Write("\t" + oldData.Name + " is being checked -----\t");
26. //oldData.WriteData();
27. **if** (currentData.Equals(oldData))
28. {
29. connected = **true**;
30. **break**;
31. }
32. }
34. **if** (!connected)
35. {
36. **if** (loggedIn)
37. {
38. currentData.Connected = **false**;
39. currentData.Uploaded = **false**;
40. currentData.TimeStamp = DateTime.Now;
41. }
42. **else**
43. {
44. activeDataList.RemoveAt(i);
45. }
47. }
49. }
51. **for** (**int** i = uiDataList.Count - 1; i >= 0; i--)
52. {
53. ProcessData currentData = uiDataList[i];
54. **bool** connected = **false**;
56. **foreach** (ProcessData oldData **in** currentDataList)
57. {
58. **if** (currentData.Equals(oldData))
59. {
60. connected = **true**;
61. **break**;
62. }
63. }
65. **if** (!connected)
66. {
67. Application.Current.Dispatcher.Invoke((Action)(() =>
68. {
69. uiDataList.RemoveAt(i);
70. }));

73. Console.WriteLine(currentData.Name + " disconected");
74. }
76. }
78. }
79. **catch** (Exception ex)
80. {
81. Console.WriteLine(ex.ToString());
82. }
83. **finally**
84. {
85. //Ververst de lijst van actieve connecties
86. currentDataList.Clear();
87. }
89. }
90. **else**
91. {
92. **string**[] tokens = Regex.Split(outLine.Data, "\\s+");
94. //Filtert alle onnodige verbindingen
95. //94.227.224.119 is het ip van sql server waarop wireshell draait
96. **if** (tokens[1].Equals("TCP") && !tokens[3].Split(':')[0].Equals("0.0.0.0") && !tokens[3].Split(':')[0].Equals("94.227.224.119") && !tokens[3].Split(':')[0].Equals("127.0.0.1") && !tokens[5].Equals("0"))
97. {
98. Process p;
99. **string** pName = "";
100. **try**
101. {
102. p = Process.GetProcessById(Convert.ToInt32(tokens[5]));
103. pName = p.ProcessName;
104. }
105. **catch** (ArgumentException ex)
106. {
107. Console.WriteLine("Process doesn't exists: " + tokens[5] + "\n" + ex.Message);
108. **return**;
109. }
111. **if** (pName == "svchost")
112. **return**;

115. //Creert nieuwe data adv van de output van netstat
116. ProcessData newData = **new** ProcessData(tokens[3].Split(':')[0], Convert.ToInt64(tokens[3].Split(':')[1]), Convert.ToInt32(tokens[5]), pName, DateTime.Now);
118. //Voegt de nieuwe data toe aan de actieve connecties
119. currentDataList.Add(newData);
121. //Controleert of een gelijkaardige connectie al eerder verbonden is
122. **bool** newCon = **true**;
123. **foreach**(ProcessData data **in** activeDataList)
124. {
125. **if** (data.Equals(newData))
126. {
127. newCon = **false**;
128. **break**;
129. }
131. }
133. //Als het een nieuwe verbinding is zal deze worden teogevoegd aan activeDataList
134. **if** (newCon)
135. {
136. activeDataList.Add(newData);
137. }
139. newCon = **true**;
140. **foreach** (ProcessData data **in** uiDataList)
141. {
142. **if** (data.Equals(newData))
143. {
144. newCon = **false**;
145. **break**;
146. }
147. }
149. **if** (newCon)
150. {
151. Application.Current.Dispatcher.Invoke((Action)(() =>
152. {
153. uiDataList.Add(newData);
154. }));
155. }
156. }
157. }
158. }
159. **catch** (Exception ex)
160. {
161. Console.WriteLine(ex.ToString());
162. }
163. }

## Uitleg C# code:

## PHP:

1. <?php
2. session\_start();
3. **require\_once** '../../../Scripts/PHPScripts.php';
5. $con = connectDataBase();
7. $boxes = **array**();
9. $names = **array**();
10. $timestamps = **array**();
11. $graphData = **array**();
12. $connectionsPerName = **array**();
14. $range = $\_POST['range'];
15. $step = $\_POST['step'];
17. $datatime;
18. $dataPoint = 0;
19. $pName = "";
21. $numberQuery = $con->prepare('call sp\_GetAllActiveConSince(?, ?, ?)');
22. $numberQuery->bind\_param('iii', $\_SESSION['userID'], $range, $step);
24. $numberQuery->execute();
25. $numberQuery->bind\_result($pName, $datatime, $dataPoint);
27. **while**($numberQuery->fetch()){
28. $pNameLow = strtolower($pName);
30. **if**(!in\_array($pNameLow, $names)){
31. array\_push($names, $pNameLow);
33. $timestamps[$pNameLow] = **array**();
34. $graphData[$pNameLow] = **array**();
35. $connectionsPerName[$pNameLow] = 0;
36. }
38. **if**($connectionsPerName[$pNameLow] != $dataPoint){
39. $connectionsPerName[$pNameLow] = $dataPoint;
40. array\_push($graphData[$pNameLow], $dataPoint);
42. $time = date("H:i", strtotime($datatime));
43. array\_push($timestamps[$pNameLow], $time);
44. }
45. }
46. $numberQuery->close();
48. **foreach**($names **as** $name){
49. $box = "<div **class** = 'ProcessBox'>
50. <p **class** = 'pName'>$name</p>
51. <p **class** = 'pRam'>Ram: 0%</p>
52. <p **class** = 'pCPU'>CPU: 0%</p>
53. <p **class** = 'pAmount'>Verbindingen: $connectionsPerName[$name]</p>
54. <div **class** = 'pDetail'><span>Details</span></div>
55. <p **class** = 'pGraph'><input type = 'hidden'/><canvas **class** = 'graphCanvas'> </canvas></p>
56. </div>";
58. $boxes[$name] = $box;
59. }
60. echo json\_encode(**array**("boxes"=>$boxes, "timeHistory"=>$timestamps, "graphData"=>$graphData, "names"=>$names));
61. ?>

## Uitleg PHP Code:

# Poster:

