

Inhoud

[Intro 3](#_Toc182668537)

[Ionic 3](#_Toc182668538)

[Test case 8](#_Toc182668539)

[Conclusie 12](#_Toc182668540)

[Link theorie 13](#_Toc182668541)

[Vergelijking 17](#_Toc182668542)

[Changes made 17](#_Toc182668543)

# Intro

Ik ga een to do list app maken. Hierbij kan je een taak toevoegen, verwijderen of aanduiden dat deze gemaakt is. Naast jouw eigen taken kan je ook een smiley zien welke aangeeft hoe het met je progressie is van de te maken taken die je in het totaal moest doen.

Zo geeft de smiley 3 stemmingen weer:

1. Hij is blij en groen als je meer dan 60% van je taken al gedaan hebt.
2. Hij is neutraal en oranje als je tussen de 40% en de 60% zit.
3. Hij is sip en rood als je minder dan 40% van je te maken taken maar gedaan hebt.

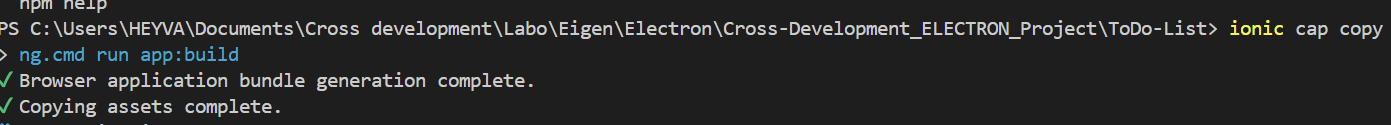
Ook kan je de taken in een file gaan wegschrijven. Zo kan je deze altijd terug kijken, of opnieuw gaan inlezen.

# Electron

Als eerste heb ik electron op mijn apparaat geinstaleerd doormiddel van het commando “npm install -g electron” uit te voeren.

Vervolgens ben ik begonnen met een electron app te maken vanaf de ionic app omdat, electron ook ionic ondersteunt. Ionic is eerder op de grafische laag van een app gericht, het is de bedoeling om de native feel and look van het platform waarop je de app gebruikt na te bootsen. Electron is meer voor de achterliggende processen (interactie met het OS) en heeft heen grapische components.

Als je electron wilt gebruiken moet je eerst buiden doormiddel van “ionic cap copy”



Afbeelding met tekst, Lettertype, schermopname

Automatisch gegenereerde beschrijving

Of doormiddel van de build knop bij de ionic plugin

Afbeelding met tekst, schermopname, software

Automatisch gegenereerde beschrijving

En dan na het builden kan je gaan runnen wat kan gedaan worden door het commande “electron electronmain” te gebruiken.

Afbeelding met tekst, Lettertype, schermopname, ontwerp

Automatisch gegenereerde beschrijving

Het tweede deel van het commando (electronmain) is de naam van de folder waar de javascript files staan die nodig zijn voor electron te kunnen gebruiken.

De run knop van de ionic plugin mag/kan je NIET gebruiken. Omdat je hier door de electron functies/api (index.js, preload.js…) niet kan gevonden/gebruikt worden.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijvingJe kan dit ook altijd in 1 keer laten doen doormiddel van een nieuw commando (script) te maken. Wat kan gedaan worden in de package.json file van je project. Je kan dit zien de afbeelding hiernaast. Als je onder script kijkt naar de run naam zie je de 2 commando’s in 1 string staan waardoor je gewoon het commando “npm run run” gebruiken om je project te builden en te runnen met electron.

In de source folder van je ionic project heb je een <base> element staan met href=”/”. Je moet dit veranderen naar href=”./” wat aan je programma zegt dat alle relative urls in je programma beginnen vanaf het huidige folder. In electron is dit zeker belangrijk omdat je hierdoor weet dat alle documenten (stylsheets, scripts…) correct worden ingeladen.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

We wetten al hoe we het project kunnen starten en wat ervoor nodig is.  
Maar welk heb je dan extra nodig om het om te zetten van ionic naar electron op vlak van functionaliteit?

Het eerste nieuwe bestand wat ik ga uitleggen wat ik extra heb gedaan om electron in mijn project te laten werken is Window.d.ts

Hier gaan we in een Window interface de functies zetten die we in de electron javascript files zetten om te kunnen gebruiken. Een soort van function declaratie om de angular compiler slimmer te maken. Want als we dit niet doen dan krijgen we van de typescript compiler compile errors. Maar zelf is dit nog niet de functie declaratie. Je kan het meer zien als een blueprint voor de angualr compiler.

Deze soort van declaratie houdt in dat we de naam van de functie zetten gevolgd door een dubbel punt (:) met ronde haakjes waarin we de variabelen kunnen meegeven die we in de functie kunnen gebruiken. Met een pijltje (=>) waarna de return type van de functie volgt.

Afbeelding met tekst, schermopname, software, scherm

Automatisch gegenereerde beschrijving

Deze interface is de brug van de electron api tussen de renderer en het main process.

Afbeelding met tekst, schermopname, Lettertype, software

Automatisch gegenereerde beschrijving  
Het volgende bestand dat we gaan bekijken is de preload.js

Hierin gaan we alle functies zetten van het main process die we kunnen gebruiken in de renderer. Een beetje zoals de window.d.ts file. Het enige verschil is dat dit de daadwerkelijke function decalartie is. Dit is dan ook hetzelfde opgebouwd als de window interface. Beginnend met de naam van de functie gevolgd door dubbelpunt met tussen de ronde haakjes de parameters die je gaat meegeven aan de functie. Vervolgens een pijl die het oproepen van de functie in index.js (het main process) gaat volbrengen.

Ik heb hier 2 manieren gebruikt om de functie op te roepen:

1. ipcRenderer.send
2. ipcRenderer.invoke

Het grote verschil tussen send en invoke is dat bij send je geen return waarde hebt/krijgt van het main process. Terwijl bij invoke krijgje een promise terug als return waarde.

Achter de ronde haakjes van de invoke of send zien we dan als eerste de naam van de functie in het main process staan (de naam waar je in index.js naar luistert). Met de parameters die je kan meegeven.

De volgende file die er nieuw is in bijgekomen is index.js wat alle handelingen gaat maken van het main process. Wat staat er allemaal in deze file?

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijvingOm te beginnen zien we dat de window wordt aangemaakt. Met de grote (breedte en hoogte waarmee de aap wordt gestart, de minimum grote dat de window moet zijn. Ook kunnen we zien door middel van de contextIsolation = true dat het wordt afgeschermt van de Node.js api’s (de os functionaliteit). Het pad naar het preload script kan je ook zien staan. Bij mainWindow.loadFile ga je de web app van ionic aan de electron app koppelen.

Het volgende deel van de file geeft aan hoe de window wordt afgehandeld. Zoals wanneer de app klaar is om te starten (het is klaar met initialiseren) dat de window wordt gemaakt. (de functie createWindow wordt opgeroepen)

App.on(‘active’) is voornamelijk voor macOS handelingen. Wanneer je app active is maar je geen window hebt, dan moet er een window aangemaakt worden.

De laatste functie app.on('window-all-closed') gaat ervoor zorgen dat het sluiten van de app goed kan worden afgehandelt.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijvingNa deze standaard functies hebben we onze eigen functies staan. Zo zie je de eerste functie om de todo items naar een file te schrijven. Hierbij hebben we een listener staan die luistert naar write-files (die kan worden aangeroepen vanuit de preload.js en de parameters die worden meegegeven. Het event wat we niet gebruiken, de naam die de file moet hebben waarin we willen in gaan opslaan en de content wat er naar de file moet worden geschreven. Dan hebben we 2 variabalen die we aanmaken om te gaan gebruiken in de functie. In de eerste var todoListPath gaan we de 2 paden toe voegen, het eerste pad gaan we zoeken naar de documents folder waaraan we ToDo\_List aan gaan toe voegen (zo gaan we alle files gaan plaatsen in de documents folder in de map ToDo\_List). De 2de var is om de volledige path in te zetten (de folder documents/ToDo\_List/filename te zetten). Zo gaan vervolgens proberen met een try catch voor errors op te vangen eerst zien of de filename eindigt met de extensie .json als dit niet zo is gaan we deze extensie hier aan toevoegen. Dit heb ik gedaan zodat ik makkelijk met 1 functie naar een nieuwe file kan schrijven of een file kan gaan overschrijven (als je gaat overschrijven dan staat er al een extensie .json). Vervolgens gaan we zien of deze directory al bestaat zodat als we de app voor de eerstekeer gebruiken (een file gaan opslaan) we de folder ToDo\_List moeten gaan aanmaken. Maar als we de app al eerder hebben gebruikt om op te slaan dat we de folder niet altijd moeten gaan aanmaken. De reden dat ik dit in een aparte folder zet is om het gemakkelijker en overzichtelijker te houden om alle files op 1 plek op te slaan. Dus als de file al bestaat gaat de functie die kijkt of de file al bestaat true teruggeven en dus in de if gegaan wordt. In de if wordt dan de file toegevoegd aan het pad (zodat we documents/ToDo\_List/filename.json). Als de directory nog niet bestaat en dus naar de else wordt gestuurd. Ga we eerst de folder maken en dan ook toevoegen pad. Waarna we tot slot in de content in een json formaat gaan wegschrijven in de file.

De volgende functie is om alle files op te halen zodat we weten welke files we kunnen gaan inladen of overschrijven. We gaan hierbij eerst alle inhoud van de ToDo\_List folder inlezen en alle ingelezen onderwerpen die eindigen met een json extensie gaan door geven aan de renderer (via de preload).

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

De fs functies die in deze file komt (vanuit de get-files, read-file en write -files functies) komen van de node.js api call die alle file system handelingen gaat doen.

Afbeelding met tekst, schermopname, Lettertype, software

Automatisch gegenereerde beschrijvingDe laatste functie is om alles todo items in te lezen van een file. Hierbij krijgen we ook de filename van de renderer (via de preload) we gaan vervolgens de filename toevoegen aan het pad en dan de inhoud van de file gaan inlezen. Vervolgens gaan we de inhoud omzetten van een json formaat naar een javascript object en dit terug sturen naar de renderer (via de preload) met een extra variabelen om door te geven dat het gelukt is. Als het niet gelukt is gaat de catch blok een lege inhoud terugsturen met een error boodschap en dat het niet gelukt is.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijvingDe laatste file die er is bij gekomen door electron te gebruiken is package.json

Deze file is automatisch gegenereerd na het commando npm init in te geven met de nodige instellingen. Hierbij zie je de naam van je electron project (zelf gekozen tijdens de npm init), de versie, een beschrijving (zelf gekozen tijdens de npm init), main (zelf gekozen tijdens de npm init) wat het entry point van je electron app is en dus het main process is. Script kan je ook nog zien, author (zelf gekozen tijdens de npm init) en tot slot license.

We hebben nu al alles gezien wat er nieuwe is bijgekomen door electron aan je project toe te voegen en hoe we het project kunnen runnen. Maar nog niet hoe we in het project de electron functies kunnen gebruiken van het main process.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijvingDit kan gedaan worden door middel van de window interface te gebruiken. Wat dan window.api.Nodige functie wordt. Zoals bv window.api.writeFiles of window.api.readFiles met de nodige parameters die moeten worden mee gegeven.

Op het einde gaan we nog een installer maken met squirl. Om dit te doen gaan we electron deployen via een tool genaamd electron-forge (electron raadt deze package aan maar dit is niet van electron). Electron-forge vindt het niet leuk dat je meerdere package.json files zijn hierdoor moeten we onze structuur aanpassen.  
De package.json file die er is bij gekomen door electron toe te voegen aan ons project moeten we gaan verwijderen. Maar voor we dit gaan doen moeten we eerst in de package.json file in de root van ons project nog een paar aanpassingen maken.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijvingZo moeten het main process hier zetten door middel van “main”: “index.js” en het run script moeten we ook aanpassen van "run": "ionic cap copy && electron electronmain" naar "run": "ionic cap copy && electron ."

Het pad naar index.html moet ook aangepast worden vanaf de current dir

Afbeelding met tekst, Lettertype, schermopname, Graphics

Automatisch gegenereerde beschrijving

# Test case

Als je de app start heb ik gekeken of je de 3 tabs ziet en de smiley geel is als je nog niets hebt ingevuld, ook naar de grote van de window. Dat deze klopt aan de opstart grote.

Afbeelding met tekst, schermopname, Besturingssysteem, software

Automatisch gegenereerde beschrijving

Maar ook klopt aan de minimum grote

Afbeelding met tekst, schermopname, emoticon, smiley

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname, software, nummer

Automatisch gegenereerde beschrijving Vervolgens heb ik op de Add task tab geklikt om een taak te kunnen toevoegen. Als je direct op de add knop klikt zou er niets mogen gebeuren. Want je moet de required velden (To do (titel)) invullen.

Afbeelding met tekst, schermopname, lijn, Lettertype

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, lijn, Lettertype

Automatisch gegenereerde beschrijvingAls je dan een titel en beschrijving zet, zou het er als volgt moeten uitzien. Maar als je dan daarna de titel zou weg halen zou deze rood moeten komen te staan met nog een extra boodschap dat het veld verplicht moet ingevuld worden.

Als je een lange beschrijving wilt zetten kan dit ook gedaan worden. Om ervoor te zorgen dat je nog altijd snel een taak kan toevoegen wordt er een scroll functie aan de beschrijving input toegevoegd.

Afbeelding met tekst, schermopname, software, nummer

Automatisch gegenereerde beschrijving

Duw je dan op de add knop als je een titel hebt ingegeven, zou je naar de overview page gestuurd moeten worden. En de smiley zou ook rood moeten worden.

Afbeelding met tekst, schermopname, software, Computerpictogram

Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname, Besturingssysteem, software

Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname, Besturingssysteem, software

Automatisch gegenereerde beschrijving

Zo kan je de smiley ook van kleur laten veranderen. Je krijgt een rode smiley als minder dan 40 procent van de taken volbracht zijn, oranje als het aantal uitgevoerde taken tussen de 40 en de 60 procent ligt en groen is voor alles meer dan 60%.

Afbeelding met tekst, schermopname, Besturingssysteem, software

Automatisch gegenereerde beschrijving

Wanneer er meer taken zijn als die op het scherm passen, kan je gaan scrollen door de taken. Maar de smiley blijft altijd bovenaan je scherm vaststaan op de overview page.

Afbeelding met tekst, schermopname, software, Besturingssysteem

Automatisch gegenereerde beschrijving

De laatste tab is options.

Waarbij je jouw taken kan gaan opslaan in een txt file of gaan lezen. Deze worden opgeslagen in het document onder een ToDoList map. De map wordt aangemaakt als deze nog niet bestaat. In de txt file staat het als een json formaat in opgeslagen.

Klik door op “save file” en je ziet het volgende scherm om te kunnen gaan opslaan.

Afbeelding met tekst, schermopname, software, scherm

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname, Lettertype, Elektrisch blauw

Automatisch gegenereerde beschrijvingAls je nog geen enkel keer hebt opgeslagen in een file en dus de map nog niet bestaat zie je enkel het input veld.

Maar als je al reeds in files hebt opgeslagen en deze dus in de map ToDoList staan komen deze hier ook te staan zoals de test.txt. Hierdoor kan je deze file gaan overschrijven met je huidige taken.

Wat nu als je verkeerd geklikt hebt op save file maar je wou op load file klikken. Dan kan je teruggaan door de back button van je gsm of je kan op de cancel knop rechtsboven klikken, zo word je terug naar de options pagina gestuurd. Dit wordt tevens ook automatisch gedaan als je een file gaat saven (word je ook naar de options page geleid).

Stel je klikt door op de “load file” knop op de options page, dan zie je het volgende staan:

Afbeelding met tekst, schermopname, software, Besturingssysteem

Automatisch gegenereerde beschrijving

Ook zoals bij save is er de cancel knop. Dit doet hetzelfde als bij save en ook hier als je een file load en het is gelukt dan word je ook terug naar de options page geleid.

Als er files bestaan waaruit je zou kunnen gaan inladen zie je ze staan zoals de test.txt file. Waarbij je door op de knop te duwen je de taken kan gaan inladen. Je wordt ook terug naar de options page gestuurd en zie je ze staan op de overview page.

# Conclusie

Afbeelding met tekst, schermopname, Lettertype, diagram

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype, Rechthoek

Automatisch gegenereerde beschrijvingElectron is net zoals ionic een hybride app. Wat betekent dat het niet zoals de qt + mxe cross build is maar eerder native is. Waarbij ik niet bedoel dat dit een native app is maar het zich wel zo zou kunnen gedragen. Wat doen we dan wel?

We gaan een app maken met behulp van web technologie die gebruik maakt van html, css, js. Maar toch nog aan de native api kan om deze te gaan gebruiken.

Ondanks het gebruik van web technologie is het geen website maar wel op de achtergrond. Je kan het uitleggen als een browser die full screen loopt die je niet kan zien als gebruiker. Je kan de app gebruiken als een desktop app op alle verschillende platformen zoals linux, windows en macOS. Helaas ondersteund electron niet meer de mobile devices.  
Door het feit dat je werkt met een web technologie is de app snel ontwikkeld omdat je maar 1 code moet maken om het op meerdere platformen te kunnen gaan gebruiken.

Ook is het en goed alternatief om te gebruiken in plaats van c++, QT, .Net …

Afbeelding met tekst, diagram, schermopname, Lettertype

Automatisch gegenereerde beschrijvingElectron gebruikt chromium om de webpagina in te laden en node.js om de native api functies te kunnen gebruiken.

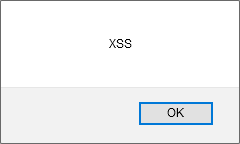
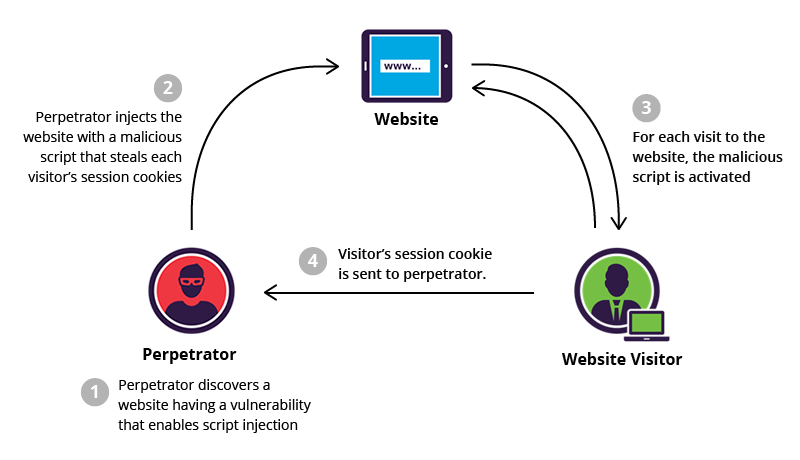
Omdat het niet veilig is om vanuit de renderer de api calls te doen gaan we er een preload script tussen zetten om alle communicatie tussen de api en de app te laten verlopen.  
Afbeelding met tekst, diagram, schermopname, Lettertype

Automatisch gegenereerde beschrijving  
zo kunnen we het preload script ook context bridge noemen omdat we de api calls gaan afschermen vanaf de renderer.  
De communicatie van het renderer process en main process gebeurd via IPC (inter-Process Communication) over het preload script dus.

# Link theorie

Dit wordt ook in de andere punten uitgelegd.

Normaal gezien wordt elk renderer procss in een sandbox gestoken waardoor je in een veilige omgevong zit. Door deze veilige omgeving kan je niet aan files, os functies dus moeten we deze sandbox afzetten om nog wel aan de files en os functies te kunnen. Wat op zijn beurt grote gevaren met zich meebrengt, zoals een cross site script attack.



Electron weet dat dit een probleem is (vroeger was dit een groter probleem) en laat daarom weten aan developers dat ze er rekening mee houden. Door zoveel mogelijk af te zetten en enkel het minimum aanzetten. Als je het dan toch niet vertrouwt ga je het best zo veel mogelijk gaan isoleren zoals bv als je een file gaat downloaden dit te laten doen door een browser omdat hier meer controle op zit tijdens het downloaden. Een andere goede manier van werken is door vanaf het begin van het maken van de app al te denken aan security. Zo heb je een checklist met 17 elementen om ervoor te zorgen dat je app veilig blijft.

|  |  |  |
| --- | --- | --- |
| 1 | Only load secure content | Ik laad geen content in buiten mijn app. |
| 2 | Disable the Node.js integration in all renderers that display remote content | Dit wordt in mijn app gedaan door het preload script |
| 3 | Enable context isolation in all renderers | Ik heb contextIsolation toe gevoegd aan de webPreferences |
| 4 | Use sesion.setPermissionRequestHandler() in all sessions that load remote content | Dit heb ik nieuw moeten toevoegen om hieraan te voldoen |
| 5 | Do not disable webSecurity | Dit is van toepassing omdat ik webSecurity niet heb disabled en zelfs nog voor de zekerheid heb enabled. |
| 6 | Define a Content-Security-Policy and use restrictive rules (i.e. script-src 'self') | Ik heb dit toe gevoegd aan de index.html file om hieraan te voldoen. |
| 7 | Do not set allowRunningInsecureContent to true | Ik heb dit op false moeten zetten |
| 8 | Do not enable experimental features | Ik heb dit ook explisiet op false gezet |
| 9 | Do not use enableBlinkFeatures | Dit gebruik ik niet. |
| 10 | < webview>: Do not use allowpopups | Dit gebruik ik niet. |
| 11 | < webview>: Verify options and params | Ik gebruik geen webview dus heb ik dit ook niet nodig. |
| 12 | Disable or limit navigation | Ik heb de index.js volgend er moeten bij zetten om er voor te zorgen dat enkel mijn files kunnen worden naar geroute.  en in de routes heb ik een extra stap toe gevoegd dat deze standaard redirect naar overview page als ik route. |
| 13 | Disable or limit creation of new windows | Ik heb dit moeten toe voegen aan index.js |
| 14 | Do not use openExternal with untrusted content | Dit gebruik ik niet. |
| 15 | Disable the remote module | Ik heb remote op false gezet |
| 16 | Filter the remote module | Ik heb remote gedisable dus moet ik niet fileten |
| 17 | Use a current version of Electron | Ik gebruik de laatste nieuwe stabel version van electron. Wat versie 33.0.0 is maar er is al een beta versie van versie 34.0.0 meer deze is pas stabel vanaf 14 januari 2025 |

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijvingOm er voor te zorgen dat je app minder op een browser lijkt en meer op een app heb ik via de global css gezegt dat je het hele body niet kan geselecteerd worden maar de input Velden of textarea wel kunnen geselecteerd worden.

# Vergelijking

## Changes made

|  |  |  |
| --- | --- | --- |
|  | Qt + MXE Cross compiled | Ionic |
| Smiley plaatsing |  |  |

In de qt app hadden we een groter scherm waardoor we een mogelijkheid hadden om de smiley naast de taken te zetten maar deze ruimte hebben we niet in de ionic app (mobile) waardoor ik deze bovenaan heb gezet.

|  |  |  |
| --- | --- | --- |
|  | Qt + MXE Cross compiled | Ionic |
| Menu bar |  |  |
| Menu bar plaatsing |  |  |

Zo is de menubar/navigatie ook anders gedaan. Zoals de plaatsing: in windows is het vanzelfsprekend dat dit bovenaan de window gedaan wordt. Terwijl dit op mobile logischer is dat dit beneden gedaan wordt als je een tab menu gebruikt (anders heb je een hamburger menu nodig maar dit is niet veel duidelijker). Ook heb je in de qt app meer opties in het menu (exit, file…), dit is minder van zelf sprekend op een mobile app.

|  |  |  |
| --- | --- | --- |
|  | Qt + MXE Cross compiled | Ionic |
| File saving en loading location | Afbeelding met tekst, schermopname, Lettertype, nummer  Automatisch gegenereerde beschrijving | Afbeelding met tekst, schermopname, software  Automatisch gegenereerde beschrijving |
| File save structuur | Afbeelding met tekst, schermopname, Lettertype, nummer  Automatisch gegenereerde beschrijving |  |
| File kiezen | Afbeelding met tekst, schermopname, Lettertype, lijn  Automatisch gegenereerde beschrijving |  |

In de Qt app heb ik voor het loaden en het saven van een file geopteerd om dit te doen via de menu bar en in ionic heb ik dit gedaan via een nieuwe page. De inhoud van de file is ook anders gedaan omdat de kans groter is dat je in windows je txt file nog eens gaat bekijken en op mobile wordt dit niet snel gedaan. Ook is het via Qt makkelijker om de inhoud van een file te controleren of deze juist is dan in ionic.

Voor het kiezen waar je de file opslaat is het ook anders. Net zoals in het vorige punt dat het minder snel gedaan wordt is hier ook van toepassing.

|  |  |  |
| --- | --- | --- |
|  | Qt + MXE Cross compiled | Ionic |
| scaling | Afbeelding met tekst, schermopname, scherm, software  Automatisch gegenereerde beschrijving | Afbeelding met tekst, schermopname, ontwerp  Automatisch gegenereerde beschrijving |

Omdat je in windows je venstergrootte kan aanpassen, heb ik in qt een minimum grootte ingesteld van 1 taak. Op mobile kan dit niet dus moest ik hier in ionic ook geen rekening mee houden.