

Applicatie platform advies

Thijmen Brand (480490)

08-12-2022

# Inhoudsopgave

# Introductie

Het ThijmenOS project heeft tot op heden de visie om in een browser omgeving in enkel html, css en typescript te draaien. Echter ben ik tot de conclusie gekomen dat dit met de huidige requirement welke zegt: “Een gebruiker moet een kopie van het operating system kunnen installeren en gebruiken op zijn systeem.” Niet erg handig is. Daarom ben ik gaan kijken naar hoe we deze requirement zo goed mogelijk kunnen implementeren en waar maken.

In dit document ligt ik enkele opties toe welke zouden kunnen werken en ik vergelijk ze met elkaar.

# Probleem stelling

Op het moment van schrijven heeft ThijmenOS twee dingen nodig. Een Client en een server. Deze server dient ervoor om data te kunnen bewaren op het systeem van de gebruiker. Dit werkt uitstekend als er maar een instantie zal zijn van ThijmenOS. Echter is dit niet het doel wanneer we naar de requirements lijst kijken. In deze lijst staat een requirement welke, zoals aangegeven, de volgend requirement: “Een gebruiker moet een kopie van ThijmenOS kunnen installeren en gebruiken op zijn systeem.”

Om deze requirement met de huidige architectuur van ThijmenOS tevreden te stellen dient een gebruiker extra stappen te ondernemen om het correct werkend te krijgen. Momenteel dient de gebruiker nodejs te hebben geïnstalleerd, een kopie van de server code te draaien en daarnaast nog de client code. Om dit installatie en start proces gemakkelijker te maken ga ik enkele dingen onderzoeken

* Hoe kan ik een gemakkelijk te installeren applicatie met html, css en javascript maken?
* Is de browser nog steeds de beste optie of is een desktop applicatie voor de huidige requirements beter?
* Hoe kan ik er voor zorgen dat de applicatie zonder extra benodigdheden kan draaien?

# Web applicatie vs. Desktop applicatie

Wanneer we naar het probleem kijken zien we dat de kern van het probleem bij het opslaan van bestanden en data zit. Om dit in de browser te doen word nu gebruik gemaakt van een nodejs backend wat extra complexiteit met zich mee brengt. Wat we dus idealiter zouden willen is één enkele applicatie.

Maar dan lopen we weer tegen het probleem aan van bestanden en data bewaren. Echter zijn hier in de browser verschillende oplossingen voor. Zo is er bijvoorbeeld een file system api welke een virtuele file filesystem in de browser maakt. Je kan deze gebruiken zoals je echte file system maar dan compleet virtueel in de gebruiker zijn browser.

Een andere api die gebruikt kan worden en potentieel veel belovend is de file system access api. Deze api bied daadwerkelijk toegang tot de file system van de gebruiker waarbij bestanden gemanipuleerd kunnen worden en gebruik gemaakt kan worden van de mappen structuur. Echter moet hier wel iedere sessie de gebruiker expliciet toestemming voor geven waardoor het niet weer niet heel gebruiksvriendelijk is.

Wanneer we gebruik maken van een Desktop applicatie aan de andere kant hebben we dit probleem compleet geëlimineerd. Deze applicatie draait namelijk direct op het onderliggende operating systeem en kan daarom ook direct gebruik maken van kernel features waardoor bestanden opslaan en manipuleren geen probleem is.

Daarnaast kan deze applicatie gemakkelijker volledig scherm gemaakt worden waardoor het gevoel van een volwaardige OS beter tot zijn recht komt.

# Een javascript desktop applicatie?

Eerder was het niet mogelijk om javascript native op een machine te draaien. Echter zijn er nu alternatieven zoals NodeJs of Deno. Dit zijn runtime environments die serverside draaien.

En met de komst van deze serverside javascript engines is ook de mogelijkheid ontstaan om native javascript desktop applicaties te schrijven. Er zijn een aantal opties waarmee dit gedaan kan worden. Dit zijn voornamelijk NW.js en Electron.js.

Deze twee projecten zijn beide gebaseerd op dezelfde princiepen. Je kan desktop applicaties schrijven met NodeJS op systeem level en web technologieën op UI level

## Electron vs NW.js

Deze twee frameworks werken beide heel erg hetzelfde. Er zijn onderling kleine verschillende zoals de browser runtime of chrome app support. Het grootste verschil de manier waarop het beginpunt van de applicatie werkt. Bij Electron kan je enkel een javascript bestand gebruiken welke dan een html renderd waarbij NW.js ook toestaat dat je direct een html bestand als index gebruikt.

Een bijkomend voordeel bij het gebruik van een desktop applicatie en deze frameworks is dat code word gecompiled naar bytecode waardoor NodeJS ook niet meer geïnstalleerd hoeft te zijn op de gebruiker zijn systeem.

# Conclusie

Na gekeken te hebben naar een aantal opties om het te kunnen laten werken in de browser en deze opties af gewogen te hebben tegen het maken van een desktop applicatie lijkt het mij het beste om ThijmenOS om te schrijven naar een desktop applicatie. Het interacteren met het file system en het gemak voor het installeren van ThijmenOS op het systeem van de gebruiker wegen af tegen het extra werk wat het oplevert om de code om te schrijven.

Daarnaast is het van belang om te weten welk javascript desktop applicatie framework ik zal gebruiken. Ik heb de twee meest populaire en grootste opties tegen elkaar afgewogen. Er bestaat niet veel verschil tussen de twee opties. Een voordeel bij NW.js is dat ik daar gebruik kan maken van een html bestand als index. Dit zou handig kunnen zijn omdat dit nu ook al het geval is. Een groot voordeel van Electron daar in tegen is dat het enorm groot is, er bestaat een grote community met veel literatuur.

Omdat de verschillen zo klein zijn tussen de twee frameworks en omdat Electron een hele grote community heeft raad ik aan voor Electron te kiezen om de applicatie mee te bouwen.