**ThijmenOS architectuur**

Project: ThijmenOS

Teamleden: Thijmen Brand (480490)

Opdrachtgever: Thijmen Brand

Versie: 0.1

Versiedatum: 09-11-2022

Status: Concept

# Inhoud

Inhoud

[Inhoud 2](#_Toc120028841)

[1. Inleiding 3](#_Toc120028842)

[2. Systeem context 4](#_Toc120028843)

[3. Containers en technologie keuze 5](#_Toc120028844)

[3.1 Heroverweging Containers 6](#_Toc120028845)

[4. Componenten 7](#_Toc120028846)

[Client 7](#_Toc120028847)

[Common 7](#_Toc120028848)

[Utils 7](#_Toc120028849)

[5. Klassen- en sequence diagrammen 8](#_Toc120028850)

[ThijmenOS Client 8](#_Toc120028851)

# 1. Inleiding

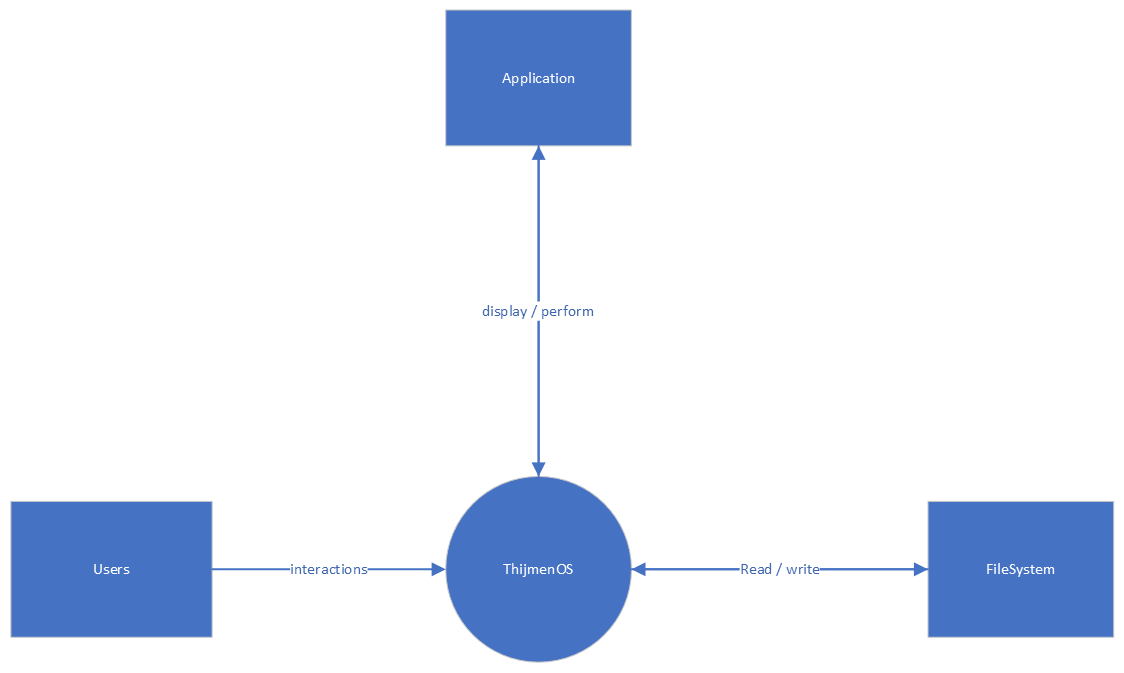
JavascriptOS is een namaak operating system wat draait in de browser. Met dit project probeer ik meer te weten te komen van operating systemen, Maar ook een diepere kennis te ontwikkelen van typescript en javascript.

Het doel is om een compleet werkend operating system te schrijven met features zoals een kernel, applicaties, notificaties etc.

In dit document zal ik het ontwerp proces van deze applicatie toelichten.

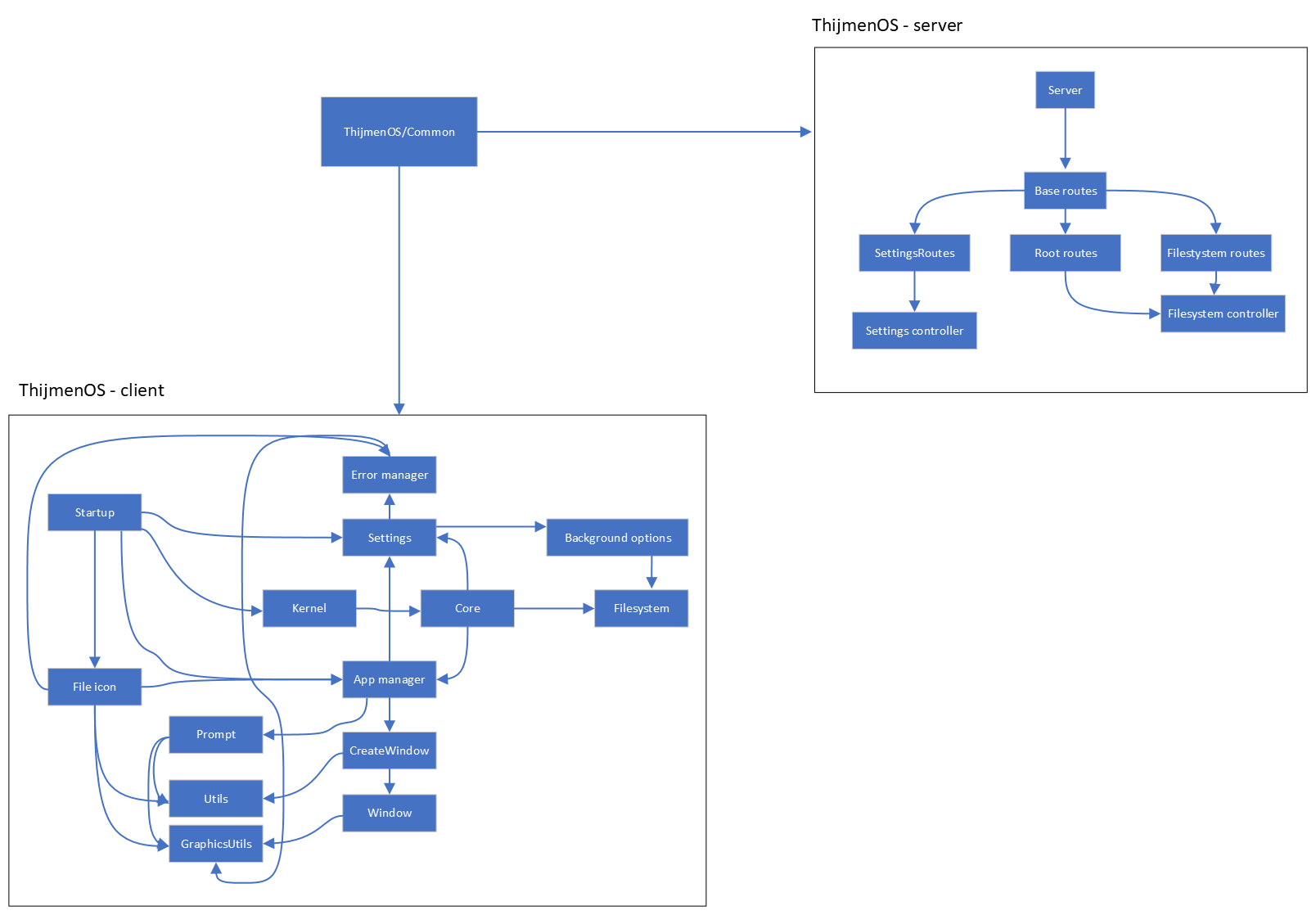
# 2. Systeem context

De systeem context van ThijmenOS is relatief simpel. Er zijn drie grote factoren. Ten eerste natuurlijk de gebruiker welke interactie heeft met de grafische interface van het operating system. Daarnaast zijn er applicaties die getoond kunnen worden. Maar deze applicatie kunnen ook weer acties uitvoeren op het operating system zoals setting veranderen en gebruik maken van de filesystem. En ten slotte dus het filesystem die kan lezen of schrijven naar bestanden.



# 3. Containers en technologie keuze

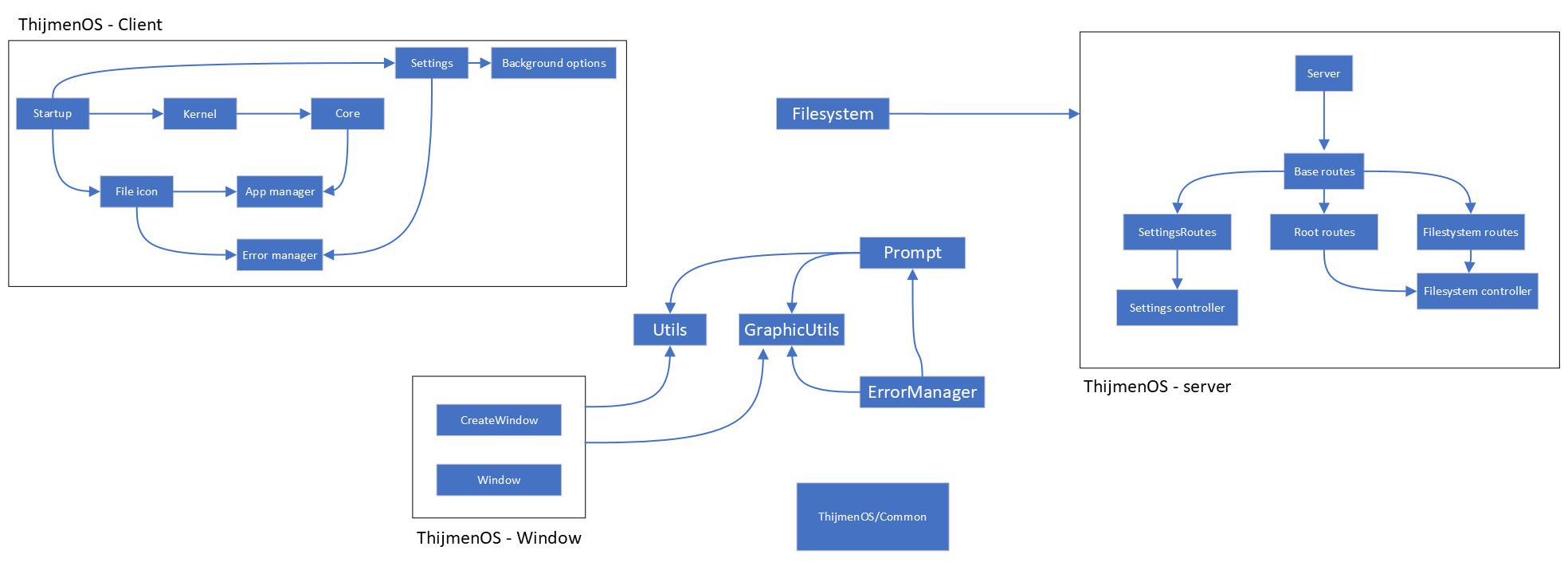
De huidige systeem architectuur is opgebouwd uit drie componenten. De client, de server en common. De client zorgt voor alles omtrent applicaties. Het visualiseren ervan, maar ook de processen daaromheen. De server zorgt vooral voor alles wat te maken heeft met het filesystem. Wanneer er bijvoorbeeld een bestand gelezen moet worden word er een API request naar de server gestuurd welke het bestand dan uitleest en naar de front-end stuurt. En slotte is er een common component welke gedeelde interfaces en variables beschikbaar stelt. Denk dan bijvoorbeeld aan een settings interface of configuratie opties.



Mijn doel is om zoveel mogelijk client side te doen. Omdat je dan het gevoel krijgt dat het echt een operating system is wat op een fysiek stuk hardware draait. Maar omdat je natuurlijk geen bestanden kan gebruiken en opslaan aan de client side maak ik gebruik van een server. De server dient dan ook vooral om data wat langer bewaard moet worden op te slaan. Zoals bestanden of instellingen.

## 3.1 Heroverweging Containers

Na kritisch te hebben gekeken naar de huidige container structuur en het zelf ondervinden van het ontwikkel proces ben ik tot de conclusie gekomen dat er vooral aan de client side erg veel onnodige complexiteit zit. Daarom heb ik ervoor gekozen om verschillende componenten die niet te veel op andere dependancies leunen los te trekken en net als common een eigen package te maken. Dit versimpelt het ontwikkel proces en het maakt de code meer testbaar.



# 4. Componenten

## Client

<<TODO: Vertalen naar nederlands>>

Everything starts with the index.ts file thats is being called from the index.html file. This file calls the startup class which initiates the nesecerry processes like settings gathering, kernel operations and it fetches desktop files to show.

Then the desktop files will be rendered and when you click an icon the app manager is consulted. Based on the fil e type the file will be executed or a default application will be searched for the filetype. If it can't find that either it goes ahead and asks the user with which application the file should be openend.

Then when it is time to open the actual application the window module is called with the request to build a window for the application (for more details on application windows see the @ThijmenOS/window repository).

The last component in this module is the kernel. This is the communication between applications and the operating system. When an application wants to do something, for example read files or talk to other applications, it asks the kernel to do so. Then the kernel will handle everything and callback the application with the correct infomation.

## Common

De @thijmenos/common package bevat interfaces, types en methoden die in meer dan een thijmenos package gebruikt worden.

Common bevat een aantal dingen. De eerste is configuratie met bijvoorbeeld de backend URL en het pad waar de OS bestandsiconen kan vinden. Hier staat dus operationele informatie.

Daarnaast bevat het zoals al genoemd typen die in meer dan een package gebruikt worden. Denk bijvoorbeeld aan de structuur die gebruikt word om instellingen te noteren, of applicatie typen zoals metadata.

## Utils

De @thijmenos/utils package bevat utility methoden die niet per se gebonden zijn aan een functionaliteit. Denk dan bijvoorbeeld aan het verkrijgen van meta data van een applicatie of het genereren van een id.

# 5. Klassen- en sequence diagrammen

## ThijmenOS Client

Afbeelding met tekst, ontvangstbewijs, schermafbeelding

Automatisch gegenereerde beschrijving