**Desktop app tools onderzoeks document**

Inhoud

[Introductie 3](#_Toc178928959)

[Welke desktop applicatie development tool vult mijn use-case het beste? 4](#_Toc178928960)

[Welke desktop applicatie development tools zijn er? 4](#_Toc178928961)

[What houden deze tools in? 4](#_Toc178928962)

[1. Electron.js 4](#_Toc178928963)

[2. Tauri 5](#_Toc178928964)

[3. WPF of WinForms (C#) 5](#_Toc178928965)

[4. Java met JavaFX 6](#_Toc178928966)

[5. Qt (C++/Python) 6](#_Toc178928967)

[methodologie 7](#_Toc178928968)

[Resultaat 8](#_Toc178928969)

[Conclusie 9](#_Toc178928970)

[Referenties 10](#_Toc178928971)

# Introductie

Mijn naam is Martijn verhappen. Ik studeer hbo ICT bij fontys hogeschool en zit in semester drie. Voor dit semester is het de bedoeling dat ik een individuele opdracht maak die twee front-end applicaties heeft. Hier heb ik de keuze gemaakt om een desktop applicatie te maken voor de manager kant van een web shop. Dit document zal informatie bevatten over welke frameworks het belangrijkste zijn voor de desktop applicatie voor mijn opdracht en welke van de frameworks ik ga gebruiken.

# Welke desktop applicatie development tool vult mijn use-case het beste?

Om te bepalen welke desktop applicatie development tool het beste aansluit bij mijn use-case, moeten eerst enkele populaire opties onderzocht worden. Dit document beantwoordt daarom de volgende vragen:

* **Welke desktop applicatie development tools zijn er beschikbaar?**
* **Wat houden deze tools in en wat zijn de voor- en nadelen?**

## Welke desktop applicatie development tools zijn er?

Dit literatuuronderzoek richt zich op vijf veelgebruikte development tools voor desktopapplicaties: **Electron.js, Tauri, C# met WPF of WinForms, Java met JavaFX, en Qt (C++/Python)**. Elk van deze tools heeft unieke eigenschappen, voordelen en nadelen.

## What houden deze tools in?

Dit onderdeel bespreekt vijf populaire frameworks voor het ontwikkelen van desktopapplicaties. Elk framework ondersteunt meerdere programmeertalen en is geschikt voor specifieke platformen en use-cases. Hierbij worden de belangrijkste kenmerken van **Electron.js, Tauri, WPF/WinForms, JavaFX**, en **Qt** toegelicht. Het doel is om inzicht te geven in de functionaliteiten, gebruiksvriendelijkheid, en beperkingen van deze frameworks.

## 1. Electron.js

**Electron.js** – vaak kortweg Electron genoemd – is een framework voor het ontwikkelen van desktopapplicaties met behulp van **JavaScript**, **HTML**, en **CSS**. Het integreert **Node.js** voor de backend en **Chromium** voor de frontend, waardoor één enkele codebase gebruikt kan worden om cross-platform applicaties te creëren voor **Windows**, **macOS**, en **Linux**. Electron vereist geen ervaring met native development.

**Voordelen:**

* **Cross-platform:** Eén codebase voor meerdere platformen.
* **Grote community:** Wordt gebruikt door bekende bedrijven zoals Discord, WordPress en WhatsApp.
* **Webtechnologieën:** Biedt ondersteuning voor populaire technologieën zoals JavaScript en HTML.

**Nadelen:**

* **Zwaar:** Electron-applicaties zijn relatief zwaar omdat zowel Node.js als Chromium gebundeld worden.
* **Grote bestandsgrootte en resource-gebruik.**

(*Introduction | Electron, n.d.)*

## 2. Tauri

**Tauri** is een lichte toolkit voor het bouwen van desktopapplicaties. Het is gebaseerd op **Rust** voor de backend en ondersteunt vrijwel elke front-end technologie zoals **HTML**, **CSS**, **JavaScript**, en frameworks zoals **Next.js** en **SvelteKit**. Voor het genereren van webinhoud op Windows is **WebView2** vereist.

**Voordelen:**

* **Efficiënt:** Lichter dan Electron, waardoor applicaties minder geheugen en CPU gebruiken.
* **Cross-platform:** Ondersteunt Windows, macOS en Linux.
* **Flexibele front-end ondersteuning:** Werkt met verschillende front-end frameworks.

**Nadelen:**

* **Relatief nieuw:** Bevat minder features en heeft een kleiner ecosysteem dan Electron.
* **Rust-kennis vereist:** Ontwikkelaars moeten ervaring hebben met Rust.

(*What Is Tauri? | Tauri Apps, n.d.)*

## 3. WPF of WinForms (C#)

**WPF (Windows Presentation Foundation)** en **WinForms** zijn frameworks ontwikkeld door Microsoft voor het bouwen van **Windows-only** desktopapplicaties met **C#**. Terwijl WinForms gebruikmaakt van standaard Windows-controls, biedt WPF meer flexibiliteit door de mogelijkheid om complexe, moderne gebruikersinterfaces te bouwen.

**Voordelen:**

* **Diepe integratie met Windows:** Krachtige ondersteuning voor het Windows-ecosysteem.
* **Flexibel:** WPF biedt meer mogelijkheden voor het maken van geavanceerde UI's dan WinForms.
* **Uitgebreide .NET-bibliotheken:** Zowel WinForms als WPF maken gebruik van het rijke .NET-ecosysteem.

**Nadelen:**

* **Windows-only:** Applicaties werken uitsluitend op Windows, wat de cross-platform mogelijkheden beperkt.
* **WinForms is verouderd:** Minder geschikt voor moderne applicaties in vergelijking met WPF.

(*WPF Vs. WinForms - the Complete WPF Tutorial, n.d.)*

## 4. Java met JavaFX

**JavaFX** is een framework voor het ontwikkelen van cross-platform desktopapplicaties met **Java**. Het biedt ondersteuning voor Windows, macOS, en Linux. JavaFX biedt een moderne toolkit voor het bouwen van clientapplicaties, en er is uitgebreide documentatie beschikbaar dankzij de samenwerking van vele ontwikkelaars en bedrijven.

**Voordelen:**

* **Cross-platform:** Ondersteunt Windows, macOS, en Linux.
* **Krachtige UI-mogelijkheden:** Geschikt voor het ontwikkelen van complexe, interactieve interfaces.
* **Grote community:** Java heeft een uitgebreid ecosysteem en veel beschikbare libraries.

**Nadelen:**

* **Leercurve:** Er is aanzienlijke kennis van Java nodig om het framework effectief te gebruiken.
* **Complexere UI-styling:** UI-styling met JavaFX is minder intuïtief dan met webtechnologieën zoals CSS.

(*JavaFX, n.d.)*

## 5. Qt (C++/Python)

**Qt** (uitgesproken als “cute”) is een cross-platform framework voor het ontwikkelen van desktopapplicaties. Het ondersteunt zowel **C++** als **Python** (via PyQt of PySide). Qt biedt veel controle over UI-elementen en wordt gebruikt voor het ontwikkelen van professionele, schaalbare desktopapplicaties.

**Voordelen:**

* **Cross-platform:** Ondersteuning voor Windows, macOS en Linux.
* **Krachtige UI-bibliotheek:** Geeft veel controle over de interface en is geschikt voor veeleisende applicaties.
* **Ondersteunt meerdere talen:** Beschikbaar in zowel C++ als Python.

**Nadelen:**

* **Complex:** Qt heeft een steile leercurve, vooral voor eenvoudige applicaties.
* **Overkill voor kleine projecten:** Qt is vaak te uitgebreid voor simpele desktopapplicaties.

(*Desktop App Development Tools & UI Design | QT, n.d.)*

# Gekozen tool en motivatie

Om de vraag “welke desktop applicatie framework kan ik het beste gebruiken bij het maken van de manager kant van een web shop” heb ik naar vijf frameworks gekeken en waar zij voor dienen. Hiervoor heb ik de voor- en nadelen van elk framework onderzocht zodat ik aan kan geven welk framework het beste werkt voor mijn project daarna, zal ik een klein prototype maken van het literatuuronderzoek uitgekozen framework en als deze bevalt gebruik ik hem in mijn eigen project.

## Literatuur onderzoek resultaat

In dit hoofdstuk noem ik alle frameworks die ik niet ga gebruiken op en leg uit waarom ze niet gekozen zijn voor mijn project. In het hoofdstuk conclusie leg ik uit waarom ik het overgebleven framework zal gebruiken.

Electron.js: Electron is zwaarder dan de andere frameworks en biedt niet veel unieks aan voor mij om te zeggen dat het gebruik ervan waardevol genoeg is.

WFP of WinForms: Ik behouw het liefst de mogelijkheid voor mijn programma om cross-platform te zijn zodat ik hier later mischien nog iets mee kan doen.

Java met JavaFX: omdat ik een andere front-end ga maken in react wat ik nog moet leren lijkt het mij niet handig om een framework te gebruiken waarvan er gezegd wordt dat UI styling complex is.

Qt: Ik ken geen c++ en heb een heel basische kennis over python en samengevoegd met het moeten leren van react lijkt het mij niet verstandig om Qt te gebruiken.

## Prototype resultaat

Ik heb met Tauri een kleine start gemaakt aan een webshop desktopapplicatie. Hierin ben ik erachter gekomen dat er weinig verschil is in het maken van een tauri desktopapplicatie en een website omdat deze allebei gemaakt kunnen worden met react.

Ik ben wel op een paar dingen gekomen die voor wat problemen gezorgd hadden namelijk, de grootte van een tauri development bestand. Op aanmaak en installatie van alle benodigdheden is een tauri applicatie al meer dan 4 gigabyte groot. Dit komt door de robuuste development kit die inbegrepen is bij rust. Omdat ik dit niet wist moest ik een specifieke folder deleten om ervoor te zorgen dat mijn applicatie klein genoeg was om ingeleverd te worden in mijn school portfolio. Buiten dat waren er geen problemen met het opzetten van de applicatie en zal ik verder tauri gebruiken tijdens het maken van mijn project.

Om te compenseren voor de grootte van het development bestand is de daadwerkelijke gebruikers applicatie heel klein en werkt zonder problemen.

# Conclusie

Voor het maken van mijn desktopapplicatie ga ik gebruik maken van Tauri omdat het voor de gebruiker lichter is dan electron waardoor het meer informatie aan zal kunnen wat handig zal zijn wanneer het mogelijk gaat om honderden mischien zelfs duizenden producten waar de status bekend van moet zijn.

# Referenties

*Introduction | Electron*. (n.d.). <https://www.electronjs.org/docs/latest>

*What is Tauri? | Tauri Apps*. (n.d.). <https://tauri.app/about/intro/#:~:text=Tauri%20is%20a%20toolkit%20that,and%20the%20CLI%20leverages%20Node>.

*WPF vs. WinForms - The complete WPF tutorial*. (n.d.). <https://wpf-tutorial.com/nl/2/over-wpf/wpf-vs-winforms/>

*JavaFX*. (n.d.). <https://openjfx.io/>

*Desktop App development Tools & UI Design | QT*. (n.d.). <https://www.qt.io/product/qt-for-desktop>