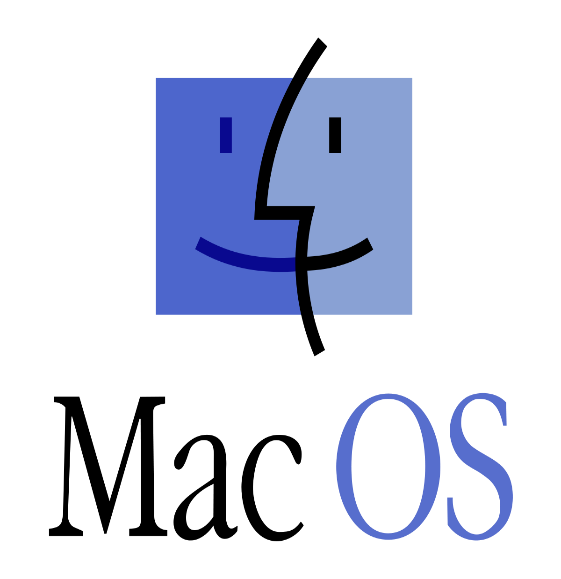
**Dokumentation PicFlip! nach IPERKA**

Autoren: Damian Sommer, Maksim Momcilovic

**Ein Bildkonvertierer für MacOS.**





Inhalt

[Informieren 3](#_Toc133939942)

[Idee 3](#_Toc133939943)

[Technologien 3](#_Toc133939944)

[Planen 4](#_Toc133939945)

[GUI-Skizze 4](#_Toc133939946)

[Architektur 4](#_Toc133939947)

[Frontend 4](#_Toc133939948)

[Backend 5](#_Toc133939949)

[Zeitplanung 5](#_Toc133939950)

[Arbeitsblock Damian 5](#_Toc133939951)

[Arbeitsblock Maksim 6](#_Toc133939952)

[Entscheiden 7](#_Toc133939953)

[Anforderungsanalyse 7](#_Toc133939954)

[Funktional 7](#_Toc133939955)

[Randbedingungen 8](#_Toc133939956)

[Realisieren 9](#_Toc133939957)

[Kontrollieren 10](#_Toc133939958)

[Testfallspezifikation 11](#_Toc133939959)

[Testprotokoll 15](#_Toc133939960)

[Testbericht: 16](#_Toc133939961)

[Auswerten 17](#_Toc133939962)

[Fazit 17](#_Toc133939963)

# Informieren

## Idee

Wir wollen einen Image-Konverter bauen für Desktop der per Drag & Drop funktioniert. Wir haben bisher die Idee so nicht gefunden bei jemand anderem.

Plattform:

Wir werden die App für MacOS bauen, da **MacOS** das ideale Drag & Drop System ist. Es hat ein ziemlich intuitives UX, was sich gut mit der Idee der App vereinen lässt.

## Technologien

Aus vorheriger Erfahrung wissen wir bereits, dass sich das Framework **Tauri** (**Rust**) bei uns als beste Option für Desktopapps herausgestellt hat. Für Pros gegenüber Electron.js, [siehe Dokumentation](https://tauri.app/). Ebenfalls UseCase für Rustapps.

Als **Frontend**-Part von Tauri verwenden wir **React** mit **Typescript**, da Typescript sicherer als Javascript ist und wir bereits solide Vorkentnisse mit React haben.

# Planen

## GUI-Skizze

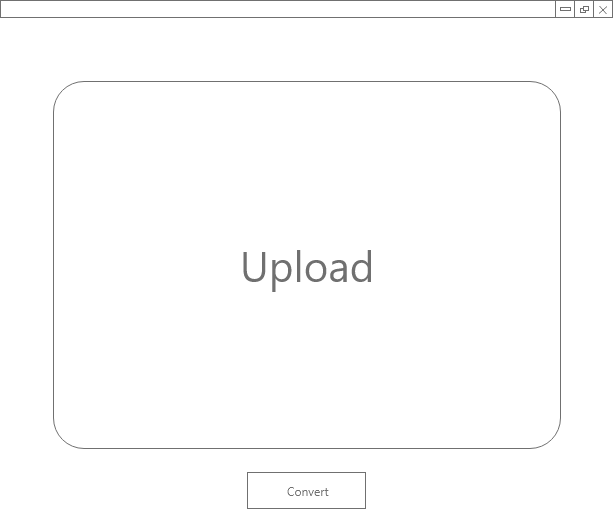


Abbildung - Entrypoint

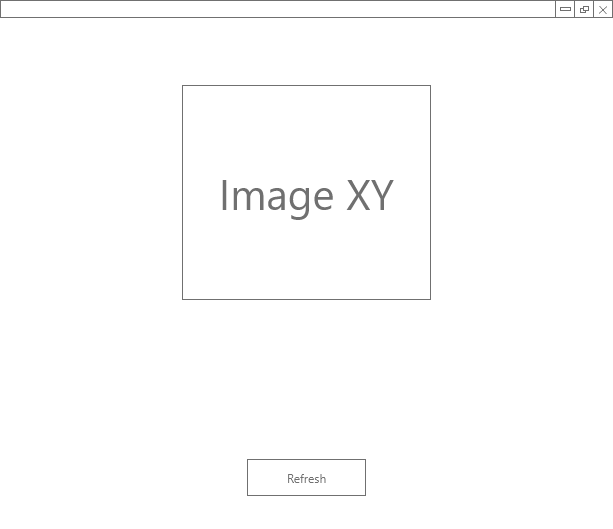


Abbildung - Nach Prozess

## Architektur

### Frontend

Das Frontend hat wie bei Tauri üblich mehrere Eventhandler, die ein Event an das Backend abfeuern und nach spezifischen «Promises» die erwartenden Daten (Bilder in neuem Format) abfängt. Dazu ist logischerweise eine angenehme Interaktion für den Nutzer das Hauptziel.

### Backend

Im Backend soll das gesamte Image-Processing stattfinden. Durch Recherche auf Crates.io konnten wir einige Library’s finden die zu unseren Use-Cases passt.

Rust selber hat leider keine eigenen Librarys, um Imageprocessing vollständig vollziehen zu können. Dank der [ImageMagicK](https://imagemagick.org/index.php) API kann Rust auf sie zugreifen um mit C/C++ teile der Konvertierung übernehmen zu können.

Um auf lokalen Systemen mit ImageMagicK in Programmen interagieren zu können braucht unser OS einen Packageinstaller wie «[Brew](https://brew.sh/index_de)» auf MacOS um die Systemvariablen zu platzieren.

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

Zeitplanung:

In unserem 3 Wöchigen Urlaub vom 10.04.2023 – 16.04.2023 für dieses Projekt arbeiten.

Dabei teilen wir wie folgt auf:

## Arbeitsblock Damian

|  |  |
| --- | --- |
| Task | Deadline |
| Frontend Mainscreen | 10.04.2023 |
| Frontend Components | 11.04.2023 |
| Frontend Eventhandlers | 12.04.2023 |

## Arbeitsblock Maksim

|  |  |
| --- | --- |
| Task | Deadline |
| Backend Architektur | 12.04.2023 |
| Backend Conversion | 13.04-2023 |
| Backend Fehlerbehebung | 14.04.2023 |

# Entscheiden

Da das meiste bereits aus der Planung in Stein gemeisselt ist, müssen wir uns nur die Frage stellen, welche Formate wir umwandeln werden bzw. anbieten können. Da wir das ImageMagicK API nutzen, sind wir auf seine Funktionalität beschränkt. D.h eigentlich fast alle Formate ausser SVG.

Konkret wären das:

* BMP
* JPG
* PNG
* GrayRaw
* WEBP
* ICO
* GIF
* PGM

Siehe: [image-convert 0.14.0 - Docs.rs](https://docs.rs/crate/image-convert/latest)

Da wir jetzt auch die Anforderungen in diesem Sinne festegelegt haben folgt nun die Anforderungsanalyse ausnahmsweise in der Entscheidungsphase.

## Anforderungsanalyse

### Funktional

|  |  |
| --- | --- |
| Nr. | Anforderung |
| 1.1 | Das Programm kann BMP in einer aller oben genannten (ausser sich selbst) kovertieren. |
| 1.2 | Das Programm kann JPG in einer aller oben genannten (ausser sich selbst) kovertieren. |
| 1.3 | Das Programm kann PNG in einer aller oben genannten (ausser sich selbst) kovertieren. |
| 1.4 | Das Programm kann GrayRaw in einer aller oben genannten (ausser sich selbst) kovertieren. |
| 1.5 | Das Programm kann WEBP in einer aller oben genannten (ausser sich selbst) kovertieren. |
| 1.6 | Das Programm kann ICO in einer aller oben genannten (ausser sich selbst) kovertieren. |
| 1.7 | Das Programm kann GIF in einer aller oben genannten (ausser sich selbst) kovertieren. |
| 1.8 | Das Programm kann PGM in einer aller oben genannten (ausser sich selbst) kovertieren. |
| 1.9 | Der Benutzer kann per Drag & Drop ein Bild hochladen. |
| 1.10 | Der Benutzer kann per Drag & Drop wenn ein Bild bereits hochgeladen wurde, mit einem erneuten Drag & Drop das Wunschbild überschreiben. |
| 1.11 | Der Benutzer kann Knopfdruck auf einen Button das Konvertieren starten |
| 1.12 | Der Benutzer kann per Knopfdruck das Programm neustarten |
| 1.13 | Der Benutzer kann per Knopfdruck das Programm beenden. |

### Randbedingungen

|  |  |
| --- | --- |
| Nr. | Anforderung |
| 3.1 | Tauri verwenden |
| 3.2 | Rust verwenden |
| 3.3 | ImageMagicK verwenden für Imageprocessing |
| 3.4 | Typescript verwenden |
| 3.5 | React verwenden |
| 3.6 | Für MacOS builden |
| 3.7 | Programm nach Abschluss builden und auf GitHub laden |

# Realisieren

/

# Kontrollieren

Testkonzept:

Software: PicFlip!

EntwicklerInnen: Maksim Momcilovic, Damian Sommer

Testdatum: 16.04.2023

Testumgebung: MacBook Air 2021 MacOS

## Testfallspezifikation

\*Sämtliche Konvertierungstests pro Testfall wurden hier pro Fileextension für alle weiteren getestet. Bsp:

Bild. Jpg 🡪 Bild.png, Bild.gray, Bild.webp, Bild.ico, Bild.gif, Bild.pgm usw.

|  |  |
| --- | --- |
| * Testfallnummer: 1.1.1 |  |
| * Getestete Anforderung: | 1.1 |
| * Voraussetzung: | Programm geöffnet, Bild (BMP) Bild.bmp vorhanden auf Schreibtisch (vergleichbar mit Windows Desktop Folder) |
| * Eingabe: | 1) Drag & Drop Bild in Upload Feld  2) Selektion beliebige Fileextension (ausser aktuelle)  4) Klick auf Convert Button |
| * Ausgabe: | 3) Convert Button erscheint  5) Erscheinung Bild.png |

|  |  |
| --- | --- |
| * Testfallnummer: 1.1.2 |  |
| * Getestete Anforderung: | 1.2 |
| * Voraussetzung: | Programm geöffnet, Bild (JPG) Bild.jpg vorhanden |
| * Eingabe: | 1) Drag & Drop Bild in Upload Feld  2) Selektion beliebige Fileextension (ausser aktuelle)  4) Klick auf Convert Button |
| * Ausgabe: | 3) Convert Button erscheint  5) Erscheinung Bild.png |

|  |  |
| --- | --- |
| * Testfallnummer: 1.1.3 |  |
| * Getestete Anforderung: | 1.3 |
| * Voraussetzung: | Programm geöffnet, Bild (PNG) Bild.png vorhanden |
| * Eingabe: | 1) Drag & Drop Bild in Upload Feld  2) Selektion beliebige Fileextension (ausser aktuelle)  4) Klick auf Convert Button |
| * Ausgabe: | 3) Convert Button erscheint  5) Erscheinung Bild.jpg |

|  |  |
| --- | --- |
| * Testfallnummer: 1.1.4 |  |
| * Getestete Anforderung: | 1.4 |
| * Voraussetzung: | Programm geöffnet, Bild (GrayRaw) Bild.gray vorhanden |
| * Eingabe: | 1) Drag & Drop Bild in Upload Feld  2) Selektion beliebige Fileextension (ausser aktuelle)  4) Klick auf Convert Button |
| * Ausgabe: | 3) Convert Button erscheint  5) Erscheinung Bild.bmp |

|  |  |
| --- | --- |
| * Testfallnummer: 1.1.5 |  |
| * Getestete Anforderung: | 1.5 |
| * Voraussetzung: | Programm geöffnet, Bild (WEBP) Bild.webp vorhanden |
| * Eingabe: | 1) Drag & Drop Bild in Upload Feld  2) Selektion beliebige Fileextension (ausser aktuelle)  4) Klick auf Convert Button |
| * Ausgabe: | 3) Convert Button erscheint  5) Erscheinung Bild.gif |

|  |  |
| --- | --- |
| * Testfallnummer: 1.1.6 |  |
| * Getestete Anforderung: | 1.6 |
| * Voraussetzung: | Programm geöffnet, Bild (ICO) Bild.ico vorhanden |
| * Eingabe: | 1) Drag & Drop Bild in Upload Feld  2) Selektion beliebige Fileextension (ausser aktuelle)  4) Klick auf Convert Button |
| * Ausgabe: | 3) Convert Button erscheint  5) Erscheinung Bild.png |

|  |  |
| --- | --- |
| * Testfallnummer: 1.1.7 |  |
| * Getestete Anforderung: | 1.7 |
| * Voraussetzung: | Programm geöffnet, Bild (GIF) Bild.gif vorhanden |
| * Eingabe: | 1) Drag & Drop Bild in Upload Feld  2) Selektion beliebige Fileextension (ausser aktuelle)  4) Klick auf Convert Button |
| * Ausgabe: | 3) Convert Button erscheint  5) Erscheinung Bild.webp |

|  |  |
| --- | --- |
| * Testfallnummer: 1.1.8 |  |
| * Getestete Anforderung: | 1.8 |
| * Voraussetzung: | Programm geöffnet, Bild (PGM) Bild.pgm vorhanden |
| * Eingabe: | 1) Drag & Drop Bild in Upload Feld  2) Selektion beliebige Fileextension (ausser aktuelle)  4) Klick auf Convert Button |
| * Ausgabe: | 3) Convert Button erscheint  5) Erscheinung Bild.png |

|  |  |
| --- | --- |
| * Testfallnummer: 1.1.9 |  |
| * Getestete Anforderung: | 1.9 |
| * Voraussetzung: | Programm geöffnet, Bild vorhanden |
| * Eingabe: | 1) Drag & Drop Bild in Upload Feld |
| * Ausgabe: | 2) Convert Button erscheint |

|  |  |
| --- | --- |
| * Testfallnummer: 1.2.1 |  |
| * Getestete Anforderung: | 1.10 |
| * Voraussetzung: | Programm geöffnet, Bild vorhanden, Bild bereits hochgeladen (nicht das gleiche) |
| * Eingabe: | 1) Drag & Drop Bild in Upload Feld |
| * Ausgabe: | 2) Convert Button erscheint  3) Reingezogenes Bild erscheint (überschrieb) |

|  |  |
| --- | --- |
| * Testfallnummer: 1.2.2 |  |
| * Getestete Anforderung: | 1.11 |
| * Voraussetzung: | Programm geöffnet, Bild bereits hochgeladen |
| * Eingabe: | 1) Klick auf Konvert |
| * Ausgabe: | 2) Reingezogenes Bild erscheint konvertiert auf Schreibtisch (bsp: von PNG 🡪 JPG) |

|  |  |
| --- | --- |
| * Testfallnummer: 1.2.3 |  |
| * Getestete Anforderung: | 1.12 |
| * Voraussetzung: | Programm geöffnet |
| * Eingabe: | 1) Klick auf Shut Down button |
| * Ausgabe: | 2) Programm schliesst |

|  |  |
| --- | --- |
| * Testfallnummer: 1.2.4 |  |
| * Getestete Anforderung: | 1.13 |
| * Voraussetzung: | Programm geöffnet |
| * Eingabe: | 1) Klick auf restart button |
| * Ausgabe: | 2) Programm schliesst  3) Programm startet mit leerem main-screen (Ohne Bild) |

## Testprotokoll

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Testfall | Resultat | Datum | Tester | Bemerkung | Unterschrift |
| 1.1.1 | OK | 16.04.2023 | Maksim Momcilovic | / | M.Momcilovic |
| 1.1.2 | OK | 16.04.2023 | Maksim Momcilovic | / | M.Momcilovic |
| 1.1.3 | OK | 16.04.2023 | Maksim Momcilovic | / | M.Momcilovic |
| 1.1.4 | OK | 16.04.2023 | Maksim Momcilovic | / | M.Momcilovic |
| 1.1.5 | OK | 16.04.2023 | Maksim Momcilovic | / | M.Momcilovic |
| 1.1.6 | OK | 16.04.2023 | Maksim Momcilovic | / | M.Momcilovic |
| 1.1.7 | OK | 16.04.2023 | Damian Sommer | / | D.Sommer |
| 1.1.8 | OK | 16.04.2023 | Damian Sommer | / | D.Sommer |
| 1.1.9 | OK | 16.04.2023 | Damian Sommer | / | D.Sommer |
| 1.2.1 | OK | 16.04.2023 | Damian Sommer | / | D.Sommer |
| 1.2.2 | OK | 16.04.2023 | Damian Sommer | / | D.Sommer |
| 1.2.3 | OK | 16.04.2023 | Damian Sommer | / | D.Sommer |
| 1.2.4 | OK | 16.04.2023 | Damian Sommer | / | D.Sommer |

## Testbericht:

Die Imageconverter App wurde einem umfangreichen Test unterzogen und hat alle Tests mit 100% Erfolg bestanden.

Bezüglich Funktionalität wurde die App auf ihre Fähigkeit geprüft, verschiedene Bildformate in andere Formate zu konvertieren. Alle Tests wurden erfolgreich durchgeführt. Die App zeigte eine schnelle Konvertierung von Bildern ohne lange Wartezeiten. Auch bei der Konvertierung von größeren Dateien war die App schnell und effizient.

Zusammenfassend lässt sich sagen, dass die Imageconverter App alle Tests mit 100% Erfolg bestanden hat und eine schnelle, zuverlässige und benutzerfreundliche Lösung zum Konvertieren von Bildern bietet. Die App ist kompatibel mit MacOS und arbeitet auch bei der Konvertierung größerer Dateien effizient.

# Auswerten

## Fazit

Das Projekt zur Erstellung einer Imageconverter App mit Typescript, React und dem Framework Tauri (Rust) war ein voller Erfolg. Die Verwendung der ImageMagick API zur Konvertierung hat sich als äußerst effektiv und zuverlässig erwiesen.

Die Organisation in der Gruppe war hervorragend und wir waren in der Lage, das Projekt schnell abzuschließen. Wir konnten viel lernen, insbesondere bezüglich des Ablaufs der Bildverarbeitung. Maksim Momcilovic konnte seine Rust-Programmierfähigkeiten verbessern, während Damian Sommer seine Typescript-Fähigkeiten ausbauen konnte.

Die Kombination aus Typescript, React und Tauri war eine großartige Wahl, da sie eine schnelle und effiziente Entwicklung ermöglichte. Wir waren auch beeindruckt von der Leistungsfähigkeit des Tauri-Frameworks, das eine nahtlose Integration zwischen Rust und Javascript ermöglichte.

Insgesamt sind wir mit dem Ergebnis unseres Projekts sehr zufrieden und sind stolz auf das, was wir als Team erreicht haben. Wir haben eine benutzerfreundliche und zuverlässige Imageconverter App entwickelt, die auf MacOS abgestimmt ist. Wir sind zuversichtlich, dass unser Projekt eine wertvolle Ressource für Menschen sein wird, die häufig mit Bildkonvertierungen arbeiten. Daher haben wir das Programm auch kostenlos nach dem Build auf GitHub als Binary zur Verfügung gestellt und eine Anleitung zum Download verfasst.