Raspirus docs

A simple hash-based virus scanner

Benjamin Demetz

Table of contents

1. Zuhause

1.1 Willkommen bei Raspirus Docs

1.2 Inhalte:

1.3 Einführung

Dieses Projektarchiv enthält die gesamte Dokumentation des Raspirus Projekts. Sie befindet sich derzeit in der Entwicklung und daher nicht zu zuverlässig.

Letztes Update: 7. April 2023
Erstellt: 3. April 2023

2. FAQ

2.1 Wie generiere ich die Dokumentation für dieses Projektarchiv?

You can find the generated documentation in the rust folder and in case you want to generate your own, you can do so by using the `cargo doc` command. Hier sind einige Parameter, die Sie mit ihm verwenden können: - `--no-deps`: Ignoriert Abhängigkeiten, dokumentiert nur den Code selbst - `--release`: Es ist im Allgemeinen besser als ein Debug - `--target-dir`: Where to output the docs All together, the command might look something like this: \ `cargo doc --no-deps --release --target-dir=/docs/generated/`

2.2 Wie kann ich in VSCode einen Rust Analyzer einrichten, der in einer nicht-standardmäßigen Verzeichnisstruktur arbeitet?

Das Rust Analyzer Plugin in Visual Studio Code versucht nach einer Cargo.toml Datei im aktuellen Verzeichnis oder übergeordnetem Verzeichnis zu suchen. But since we packed the entire application in the `app` directory, it's unable to find the file and therefore might not work. This is a big lost, as it doesn't tell you if your Rust files have correct syntax or not. Um dieses Problem zu lösen, können Sie dem Plugin eine Option hinzufügen und den Speicherort Ihrer Cargo.toml Datei angeben. As stated in this comment, you need to add the following lines to the end of your plugin settings' JSON. Danach müssen Sie auch den Analyzer neu starten, damit die Änderung wirksam wird.

```
{
    "rust-analyzer.linkedProjects": [
        "/home/matklad/tmp/hello/Cargo.toml"
    ]
}
```

2.3 Weitere demnächst

...

Letztes Update: 7. April 2023
Erstellt: 5. April 2023

2.4 Kommentare

3. Anleitungen

3.1 KOMMT BALD

3.2 Notizen

3.2.1 Export in PDF:

- Folgen Sie den Anweisungen für Ihr Betriebssystem hier: https://github.com/orzih/mkdocs-with-pdf#requirements
- Installieren Sie alle Abhängigkeiten mit Pip mit `pip install -r requirements.txt`
- `mkdocs build` ausführen
- PDF befindet sich in site/pdf/document.pdf

3.2.2 Übersetzungen

- Link: https://github.com/ultrabug/mkdocs-static-i18n

Letztes Update: 7. April 2023
Erstellt: 3. April 2023

3.3 Kommentare

4. Installation

Diese Anleitung hilft Ihnen beim Aufbau des Projekts auf Ihrem eigenen Rechner.

4.1 Inhaltsverzeichnis

4.2 Einführung

For people that just want a working app, they can just head over to the Release page and download the executable for the correct platform. But if you are on a different Linux distribution, unsupported OS, or just want to compile the project on your own, this step-by-step guide will guide you.

4.3 Einschränkungen

- Glibc kann unter Linux Probleme verursachen: https://tauri.app/v1/guides/building/linux#limitations
- You need to use 64-bit systems, else the app might crash because it's using memory improvements that only work there
- The app is meant to be run as a "I'm the only app running on this system" app. This is important regarding RAM usage, because if you have much RAM, it will use much RAM. And if you, for some reason, try to limit the initially available RAM, the app might crash because it doesn't have the promised amount of RAM. (A future version might have a toggle for this)

4.4 Step-by-step guide

Bitte lesen Sie die ganze Anleitung einmal, bevor Sie jeden Schritt ausführen!

4.4.1 1. Repository herunterladen

Dieser Schritt ist sehr einfach, einfach das ganze Projektarchiv herunterladen, indem Sie auf die grüne Schaltfläche auf der Homepage dieses Projektarchivs klicken. Optional können Sie auch den für eine

Veröffentlichung spezifischen Code herunterladen, indem Sie die Release-Seite besuchen und die `.zip` Datei in den Assets herunterladen. Eine weitere Sache, die Sie vielleicht tun möchten, ist dieses Projektarchiv klonen

4.4.2 2. Rust installieren

Eine der Voraussetzungen, um das Projekt zu kompilieren, ist die Installation von Rst. Also make sure that your Rust installation is up-to-date with the command `rustup update`. You can check if you have Rust installed on your machine with the command `rustc --version`, if this command fails, head over to the Rust website and follow the instructions for your OS.

4.4.3 3. Install NPM

NPM wird benötigt, da das Frontend der App auf JavaScript funktioniert und im Grunde eine Webseite ist. Um zu überprüfen, ob Node.js bereits installiert ist, führen Sie die Befehle `node -v` und `npm -v` aus. Wenn einer von ihnen fehlschlägt, oder Sie feststellen, dass Sie eine ältere Version haben auf der NPM-Website installieren, um die neueste Version für Ihr Betriebssystem zu installieren. Wenn du ein WSL verwendest, könnte diese Anleitung für dich nützlich sein.

4.4.4 4. Next.js installieren

Das Frontend basiert auf JavaScript mit dem bekannten Framework Next.js, es macht die Website-Entwicklung schneller und effizienter. Sie müssen auch dieses Tool installieren, um die Anwendung erstellen zu können. Aber keine Sorge, Sie können dies einfach mit NPM: `npm installieren next@latest react@latest react-dom@latest eslint-config-next@latest`. Dies wird Next.js, React (die Next.js basiert darauf) und ESLink installieren. Mehr über den Installationsprozess erfahren Sie hier.

4.4.5 5. Tauri installieren

Tauri ist das Framework, das das Ruster Backend mit dem Next.js Frontend verbindet. Es ist ein Open-Source-Projekt von sehr freundlichen und gastfreundlichen Menschen. Unfortunately, installing Tauri is not as straightforward as other processes. Es ist sehr OS abhängig und Sie werden daher sicherstellen, dass Sie die Voraussetzungen vor dem Start erfüllen. Danach können Sie Tauri-cli installieren: `Fracht installiert Tauri-cli`. Sie könnten auch NPM verwenden, um es zu installieren, aber wir werden hauptsächlich mit Fracht in dieser kurzen Anleitung arbeiten. Schau dir in seinem FAQ-Bereich an, um zu erfahren, warum NPM für dich vielleicht besser ist.

4.4.6 6. Projektabhängigkeiten installieren

Zunächst werden wir die Knotenmodule installieren. Um dies zu tun, gehen Sie in das Verzeichnis, das den gesamten Raspirus-Code enthält. Öffnen Sie das `App` Verzeichnis, öffnen Sie ein Terminal an diesem Ort und führen Sie den Befehl `npm install` aus. Dies kann eine Weile dauern, aber es wird alle notwendigen Module herunterladen. **Warning!:** On WSL you might get an `OpenSSL is missing` error, to fix this you need to edit the file `app/src-tauri/Cargo.toml` and add the following line: `openssl-sys = {version = "0.9.66", features = ["vendored"]}` in the `[dependencies]` section.

4.4.7 7. Projekt erstellen

Bevor Sie das Projekt komplett erstellen können, gibt es noch eine Sache, die Sie vielleicht überprüfen möchten. To make sure that the Rust part of the project works fine, open the folder `app/src-tauri/` and execute the command `cargo build`. Wenn dieser Befehl erfolgreich ist, können Sie zum `app/` Verzeichnis zurückkehren.

Wenn dieser Befehl fehlschlägt, öffnen Sie bitte ein Problem in diesem Projektarchiv mit so vielen Informationen wie möglich. Wenn alles gut gelaufen ist, befinden Sie sich jetzt im `App` Verzeichnis, und Sie können sicher den Befehl `cargo tauri build` ausführen. Dieser Befehl erstellt die gesamte Anwendung und zeigt am Ende des Prozesses einen Pfad an, der Ihnen zeigt, wo sich die ausführbare Datei befindet. Standardmäßig sollten Sie es im `app\src-tauri\target\release` Ordner finden können.

4.5 Schlussfolgerung

Diese Anwendung ist im Grunde eine Website, die mit einigen Rust Code verbunden und mit dem Tauri Framework verpackt ist. Es wird daher ein grafisches Overlay benötigen, um die Website zu starten und anzuzeigen. Dieses Projekt befindet sich in ständiger Entwicklung und wenn Sie etwas Ungewöhnliches finden, haben Sie einige gute Ideen oder finden Sie einige Fehler, haben Sie keine Angst, ein Problem auf diesem Repository zu öffnen und ich werde Ihnen gerne helfen.

Letztes Update: 7. April 2023
Erstellt: 7. Februar 2023

4.6 Kommentare

5. Auslastung

5.1 KOMMT BALD

Letztes Update: 7. April 2023
Erstellt: 3. April 2023

5.2 Kommentare

6. Contributing

6.1 Beiträge

Dieses Projekt ist komplett Open-Source und wäre ohne Ihre Hilfe nicht möglich. Every contribution even if just a little can help a lot. Opening an issue if you find a bug, fixing the bug by opening a pull-request, adding translations, adding features and much more are all types of contributions that help the project go forward.

6.1.1 Übersetzungen

The app itself is made with user-friendliness in mind and therefore doesn't have much text to translate as we want the experience to be fast without slowing down the user by forcing him/her to read lots of stuff. Nonetheless, this project still needs a lot of translators. Foremost we need translators for the docs that you are reading right now, but the app also needs translators. Übersetzungen helfen die Benutzererfahrung zu verbessern und das Projekt breiter zugänglich zu machen. You can find out more on the dedicated page.

6.1.2 Coding

Die App wird hauptsächlich aus zwei Programmiersprachen gebaut: Rust und JavaScript. JavaScript is used for the frontend to build a website using the Next.JS framework and make the app look nicer and be cross-platform. The backend on the other side is written in Rust because we value speed a lot when scanning files and therefore need a fast language. To make the two communicate we use the Tauri framework. Tauri ist ein in Rust geschriebenes Open-Source-Framework, ganz neu in seinem Bereich. In Zukunft gibt es auch einen Plan, einen Python-Installer hinzuzufügen. If you have knowledge in any of these fields you are more than welcome to contribute to the project. Sie können dieses Projekt natürlich auch als Beispiel verwenden, um Ihr eigenes Projekt zu erstellen.

6.1.3 Sponsoring

KOMMT BALD

Letztes Update: 7. April 2023
Erstellt: 5. April 2023

6.1.4 Kommentare

6.2 Code

6.2.1 KOMMT BALD

Letztes Update: 7. April 2023
Erstellt: 3. April 2023

6.2.2 Kommentare

6.3 Übersetzungen

Übersetzungen sind sehr wichtig und können anderen Menschen helfen, die App einfacher zu verstehen und sie umfassender zur Verfügung zu stellen. If you know a language outside of English and would like to add your translations you can do so in two ways: The first one is to translate the documentation, the second one is to translate the frontend strings.

6.3.1 Code wird übersetzt

Es handelt sich um eine laufende Arbeit, die jedoch im Wesentlichen auf eine von zwei möglichen Wegen erfolgen wird:

Übersetzungsdatei

Es gibt eine Übersetzungsdatei im JSON-Format für jede einzelne Sprache. So the app would load the strings it needs from that file. This is a bit easier to implement for the developers, but not very efficient or great for translators, as it involves forking the repository.

Übersetzungsdienst

Wir verwenden einen externen Übersetzungsdienst, um das Projekt zu übersetzen. This option would allow translators to just translate the given strings and have an overview of how much has been translated and what still needs to be translated. Dies wäre die bevorzugte Option, aber die Kehrseite ist das Setup für diese Option. Since we don't know which service to use yet, this option will be added in the future when we have more concrete ideas and hopefully a bigger team. Wir werden wahrscheinlich Crowdin verwenden

6.3.2 Übersetzungen von Docs

The docs you are reading right now also need translations and will require a much bigger effort than translating strings used on the frontend. Die Dokumentation ist ziemlich groß und kann daher einige Zeit in Anspruch nehmen, um zu übersetzen. Als obige Option könnten wir den Service auch hier nutzen, aber das wird in Zukunft hinzugefügt werden. If you want to start adding translations right away, I would suggest to do the following:

1. Fork this repository: You can follow a guide online on how to do it properly.
2. Enter the `docs` folder and, if the language you want to translate doesn't exist yet, add a new directory whose name is the name of the language you want to write translations for. For example, it would be `it` for italian, `de` for german and so on.
3. Then change to that directory and edit the Markdown files inside that directory. Make sure you keep the original folder structure.
4. Nachdem Sie die Bearbeitung abgeschlossen haben, speichern Sie Ihre Dateien und laden sie auf GitHub in Ihren Fork hoch. Mache einen GitHub Commit
5. Jetzt können Sie eine Pull-Request in das ursprüngliche Projektarchiv öffnen und verlangen, dass Ihre Änderungen dem Hauptprojekt hinzugefügt werden.
6. Jemand des Teams wird die Übersetzungen prüfen und wenn akzeptiert, diese dem Hauptprojekt hinzufügen.

In the future we hopefully have an external service that handles all this for you, so that you can focus on your translations without distractions. Wir werden wahrscheinlich Crowdin verwenden.

Letztes Update: 7. April 2023
Erstellt: 5. April 2023

6.3.3 Kommentare

7. Developers

7.1 Entwickler

Das Raspirus-Projekt verwendet hauptsächlich zwei verschiedene Programmiersprachen und Frameworks. Eine für das Backend und eine für das Backend und eine für das Frontend. \ Das Frontend wird unter Verwendung des Next.JS Frameworks in JavaScript geschrieben. Das Backend auf der anderen Seite ist in Rust geschrieben. The communication between these two programming languages is made using the Tauri framework. Using this framework one can write functions in Rust and call them from the frontend easily.

Letztes Update: 7. April 2023
Erstellt: 5. April 2023

7.1.1 Kommentare

7.2 Backend

7.2.1 KOMMT BALD

Letztes Update: 7. April 2023
Erstellt: 3. April 2023

7.2.2 Kommentare

7.3 Frontend

7.3.1 KOMMT BALD

Letztes Update: 7. April 2023
Erstellt: 3. April 2023

7.3.2 Kommentare