Raspirus docs

A simple hash-based virus scanner

Benjamin Demetz

Table of contents

1. 1. Home

1.1 1.1 Benvenuti a Raspirus Docs

1.2 1.2 Contenuto:

1.3 1.3 Introduzione

Questo repository contiene tutta la documentazione del progetto Raspirus. Attualmente è in fase di sviluppo e quindi non è troppo affidabile.

Ultimo aggiornamento: 12 aprile 2023
Creata: 12 aprile 2023

2. 2. FAQ

2.1 2.1 Come posso generare la documentazione per questo repository?

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. Ecco alcuni parametri che potresti voler utilizzare con esso: - `--no-deps`: Ignora le dipendenze, documenta solo il codice stesso - `--release`: Generalmente è meglio di un 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 2.2 In VSCode, come posso impostare l'analizzatore di Rust per lavorare nella struttura di directory non standard?

Il plugin dell'analizzatore Rust in Visual Studio Code cerca di cercare un file Cargo.toml nella directory corrente o nella directory principale. 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. Per risolvere questo problema, è possibile aggiungere un'opzione al plugin e specificare la posizione del file Cargo.toml. As stated in this comment, you need to add the following lines to the end of your plugin settings' JSON. In seguito, dovrai anche riavviare l'analizzatore per rendere effettiva la modifica.

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

2.3 2.3 Più presto in arrivo

...

Ultimo aggiornamento: 12 aprile 2023
Creata: 12 aprile 2023

2.4 2.4 Commenti

3. 3. Guide

3.1 3.1 COMANDO SUONO

3.2 3.2 Note

3.2.1 3.2.1 Esporta in PDF:

- Segui le istruzioni per il tuo sistema operativo qui: https://github.com/orzih/mkdocs-with-pdf#requirements
- Installa tutte le dipendenze usando pip con `pip install -r requirements.txt`
- Esegue `mkdocs build`
- PDF si trova in sito/pdf/document.pdf

3.2.2 3.2.2 Traduzioni

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

Ultimo aggiornamento: 12 aprile 2023
Creata: 12 aprile 2023

3.3 3.3 Commenti

4. 4. Installazione

Questa guida vi aiuterà a costruire il progetto sulla vostra macchina.

4.1 4.1 Tabella dei contenuti

4.2 4.2 Introduzione

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 4.3 Limitazioni

- Glibc può causare problemi su Linux: 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 4.4 Step-by-step guide

Si prega di leggere l'intera guida una volta prima di iniziare ad eseguire ogni passaggio!

4.4.1 4.4.1 1. Scarica il repository

Questo passaggio è molto semplice, basta scaricare l'intero repository facendo clic sul pulsante verde sulla homepage di questo repository. Facoltativamente, è anche possibile scaricare codice specifico per una Rilascio,

visitando la pagina Rilasciare e scaricare il file `.zip` negli asset. Un'altra cosa che potresti voler fare è clonare questo repository

4.4.2 4.4.2 2. Installa Rust

Uno dei requisiti per compilare il progetto è quello di avere installato Rust. 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 4.4.3 3. Install NPM

NPM è necessario perché il frontend dell'applicazione funziona su JavaScript ed è fondamentalmente un sito web. Per verificare se hai già installato Node.js, prova ad eseguire i comandi: `node -v` e `npm -v`. Se qualcuno di loro fallisce, o si scopre che hai una versione più vecchia, vai sul sito web NPM per installare l'ultima versione per il tuo sistema operativo. Se stai usando un WSL, questa guida potrebbe essere utile per te.

4.4.4 4.4.4 4. Installa Next.js

Il frontend è costruito su JavaScript utilizzando un noto framework chiamato Next.js, rende lo sviluppo del sito Web più veloce ed efficiente. Sarà necessario installare anche questo strumento per essere in grado di costruire l'applicazione. Ma non ti preoccupare, puoi farlo facilmente con NPM: `npm install next@latest react@latest react-dom@latest eslint-config-next@latest`. Questo installerà Next.js, React (che Next.js è basato su) e ESLint. Puoi saperne di più sul processo di installazione qui.

4.4.5 4.4.5 5. Installa Tauri

Tauri è il framework che collega il backend Rust con il frontend Next.js. Si tratta di un progetto open-source realizzato da persone molto cordiale e accogliente. Unfortunately, installing Tauri is not as straightforward as other processes. È molto dipendente dal sistema operativo, e quindi ti assicurerai di soddisfare i Prerequisiti prima di iniziare. In seguito, è possibile installare Tauri suing cargo: `cargo install tauri-cli`. Si potrebbe anche utilizzare NPM per installarlo, ma lavoreremo principalmente con il carico in questa breve guida. Scopri la loro sezione FAQ per sapere perché NPM potrebbe essere migliore per te.

4.4.6 4.4.6 6. Installa dipendenze progetto

In primo luogo, installeremo i moduli del nodo. Per fare questo, vai alla directory che contiene tutto il codice Raspirus. Apri la directory `app` , apri un terminale in questa posizione ed esegui il comando: `npm install`. Questo potrebbe richiedere un po ', ma scaricherà tutti i moduli necessari. **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 4.4.7 7. Genera il progetto

Prima di poter costruire completamente il progetto, c'è ancora una cosa che potresti voler controllare. To make sure that the Rust part of the project works fine, open the folder `app/src-tauri/` and execute the command `cargo build`. Se questo comando ha successo, puoi tornare alla directory `app/`. Se questo comando fallisce, si prega di aprire un problema su questo repository con il maggior numero possibile di informazioni sull'errore. Se tutto è andato bene, ora sei nella directory `app` , ed è possibile eseguire in modo sicuro il comando `cargo tauri build`. Questo comando genererà l'intera applicazione e mostrerà un percorso alla fine del processo mostrandoti dove

si trova l'eseguibile. Per impostazione predefinita, dovresti essere in grado di trovarlo nella cartella `app\src-tauri\target\release`.

4.5 4.5 Conclusione

Questa applicazione è fondamentalmente un sito web collegato ad alcuni codici Rust e confezionato con il framework Tauri. Sarà quindi necessaria una sovrapposizione grafica per avviare e visualizzare il sito web. Questo progetto è in costante sviluppo e quindi, se si trova qualcosa di insolito, avere alcune buone idee o trovare alcuni errori, non abbiate paura di aprire un problema su questo repository e sarò felice di aiutarvi.

Ultimo aggiornamento: 12 aprile 2023
Creata: 12 aprile 2023

4.6 4.6 Commenti

5. 5. Utilizzo

5.1 5.1 COMANDO SUONO

Ultimo aggiornamento: 12 aprile 2023
Creata: 12 aprile 2023

5.2 5.2 Commenti

6. Contributing

6.0.1 6.1 Contributi

Questo progetto è completamente open-source e non sarebbe possibile senza il vostro aiuto. 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 Traduzioni

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. Le traduzioni aiutano a migliorare l'esperienza utente e a rendere il progetto più ampiamente disponibile. You can find out more on the dedicated page.

6.1.2 Coding

L'app è costruita utilizzando principalmente due linguaggi di programmazione: Rust e 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 è un framework open-source scritto in Rust, abbastanza nuovo nel suo campo. In futuro c'è anche un piano per aggiungere un programma di installazione Python. If you have knowledge in any of these fields you are more than welcome to contribute to the project. Ovviamente puoi anche usare questo progetto come esempio per creare il tuo progetto.

6.1.3 Sponsorizzazione

COMANDO SUONO

Ultimo aggiornamento: 12 aprile 2023
Creata: 12 aprile 2023

6.1.4 Commenti

6.0.2 6.2 Codice

6.2.1 COMANDO SUONO

Ultimo aggiornamento: 12 aprile 2023
Creata: 12 aprile 2023

6.2.2 Commenti

6.0.3 6.3 Traduzioni

Le traduzioni sono molto importanti e possono aiutare altre persone a capire l'app più facile e renderla più ampia. 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 Tradurre il codice

Si tratta di un lavoro in corso, ma essenzialmente sarà realizzato in uno dei due modi possibili:

File di traduzione

C'è un file di traduzione in formato JSON per ogni lingua differente. 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.

Servizio di traduzione

Utilizziamo un servizio di traduzioni esterne per tradurre il progetto. 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. Questa sarebbe l'opzione preferita, ma il lato negativo è la configurazione di questa opzione. 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. Probabilmente useremo Crowdin

6.3.2 Tradurre Documenti

The docs you are reading right now also need translations and will require a much bigger effort than translating strings used on the frontend. La documentazione è abbastanza grande e può quindi richiedere un bel po' di tempo per tradurre. Come opzione di cui sopra, potremmo utilizzare il servizio anche qui, ma questo sarà aggiunto in futuro. 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. Dopo aver completato la modifica, salvare i file e caricarli sul tuo fork su GitHub. Fondamentalmente fai un commit di GitHub
5. Ora puoi aprire una pull-request al repository originale e richiedere che le tue modifiche vengano aggiunte al progetto principale.
6. Qualcuno del team esaminerà le traduzioni e, se accettato, le aggiungerà al progetto principale.

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. Probabilmente useremo Crowdin.

Ultimo aggiornamento: 12 aprile 2023
Creata: 12 aprile 2023

6.3.3 Commenti

7. Developers

7.0.1 7.1 Sviluppatori

Il progetto Raspirus utilizza principalmente due linguaggi di programmazione e quadri diversi. Uno per il backend e uno per il backend e uno per il frontend. \ Il frontend è scritto in JavaScript utilizzando il framework Next.JS. Il backend sull'altro lato è scritto in Rust. 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.

Ultimo aggiornamento: 12 aprile 2023
Creata: 12 aprile 2023

7.1.1 Commenti

7.0.2 7.2 Backend

7.2.1 COMANDO SUONO

Ultimo aggiornamento: 12 aprile 2023
Creata: 12 aprile 2023

7.2.2 Commenti

7.0.3 7.3 Frontend

7.3.1 COMANDO SUONO

Ultimo aggiornamento: 12 aprile 2023
Creata: 12 aprile 2023

7.3.2 Commenti