L’evento inizierà alle 18:05

develer 20

Vita in Develer

L’evento inizierà alle 18:05

develer 20

Vita da develeriani

Equilibrio tra vita lavorativa e privata

- *Sede aperta 24h su 24*
- *Flessibilità oraria*
- *Banca ore*

L’evento inizierà alle 18:05

develer 20

Vita da develeriani

Al passo con i tempi

Incoraggiamento alla formazione annuale:

- *Budget 800€*
- *Budget 80 ore*

L'evento inizierà alle 18:05

develer 20

Lavora in Develer

- Web Full Stack Developer
- C++ Developer
- Python Developer

Claudia Cimino
Responsabile risorse umane

jobs@develer.com

LUCA OTTAVIANO
Tech lead

Introduzione a Slint UI toolkit

Webinar
12/04/2022

develer 20

Cos’è Slint

The fast and easy UI toolkit

Cos’è Slint

Libreria UI dichiarativa, type safe, ottimizzata per dispositivi a basse prestazioni

Target: sistemi Linux embedded o MCU

Perché Slint

Attenzione all'esperienza sviluppatore

C'è un'azienda commerciale dietro

Ispirato a QtQuick, migliorati i lati negativi

Perché è un toolkit usabile con Rust! :)

Piattaforme supportate

Linux embedded

Desktop

MCU

WebAssembly (sperimentale)

Licenze

Modello a doppia licenza:

- GPL v3
- Commerciale: prezzo fisso per prodotto + royalties

Temporaneamente:

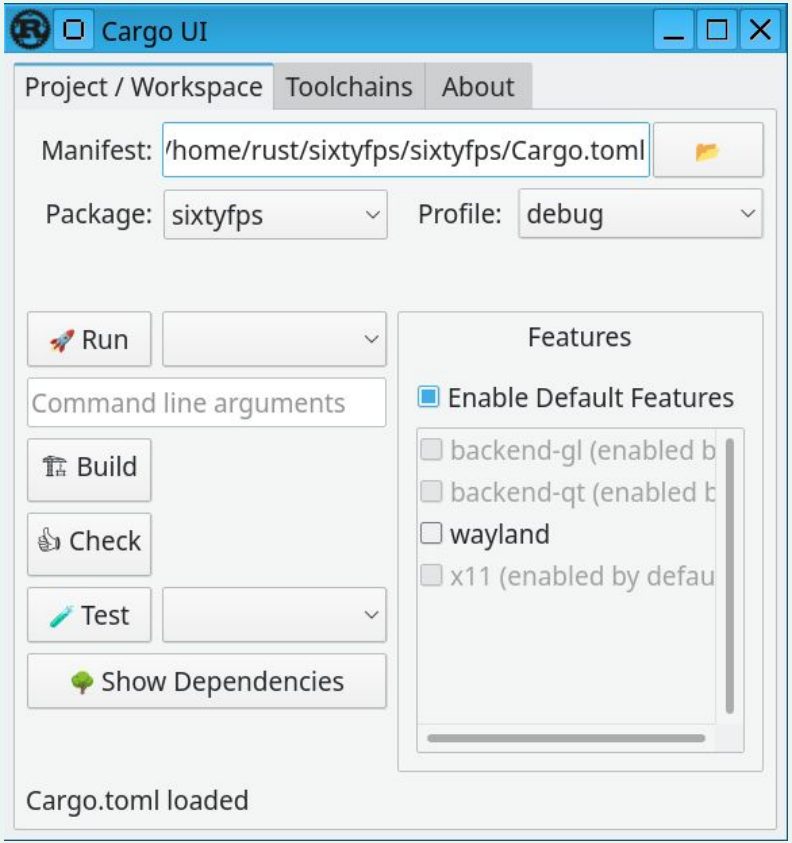
- Ambassador: gratuita, codice proprietario, marketing

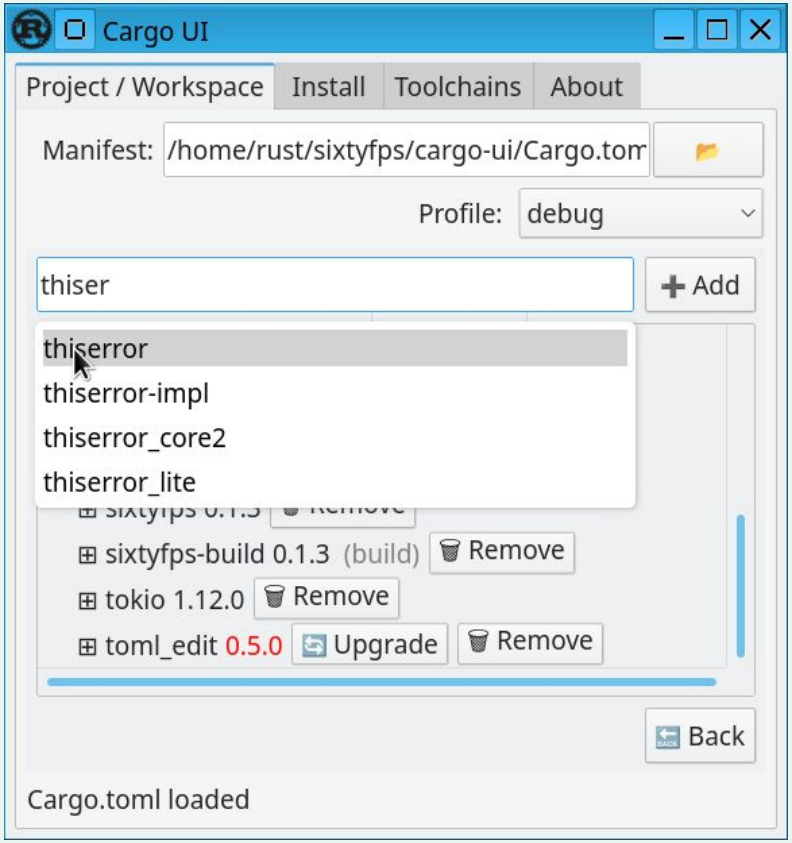
Warning

È un toolkit molto giovane, in cui le modifiche avvengono ogni giorno

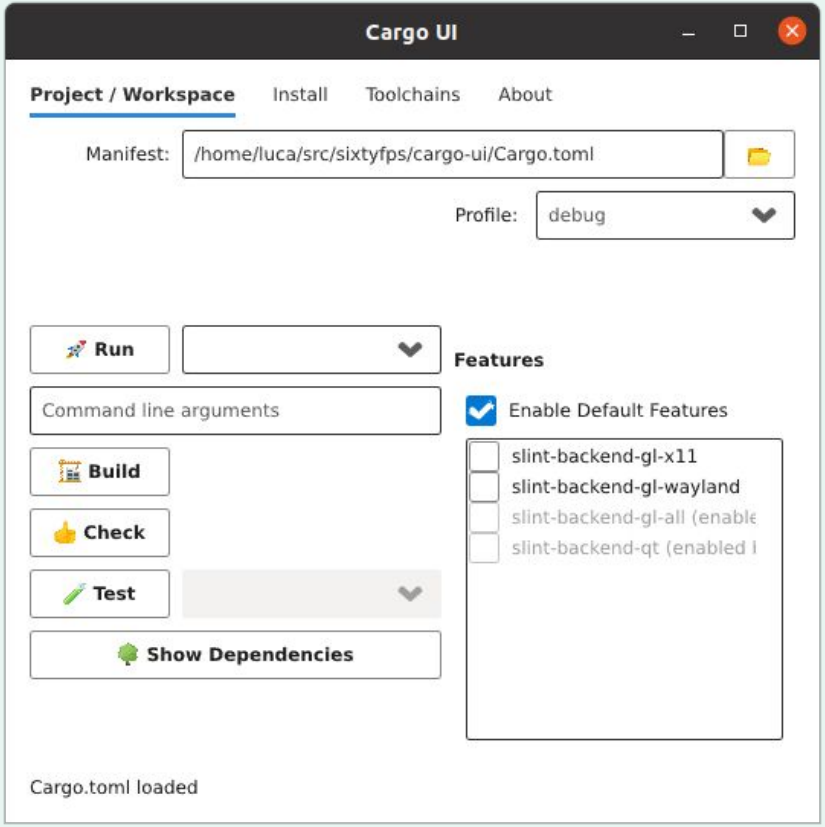
È ragionevole fare un pull ogni settimana

Screenshot (backend Qt)





Screenshot (backend nativo)



Esempio

```
Rectangle {
    width: 60px;
    height: 60px;
    background: aqua;
    Rectangle {
        width: parent.width / 2;
        height: parent.height / 2;
        background: black;
    }
}
```

Realizziamo un pulsante

```
YellowButton := Rectangle {
    callback clicked;
    property<string> text <=> innerText.text;
    width: 190px;
    height: 64px;
    background: ta.has-hover ? #FCB647 : transparent;
    border-color: #FCB647;
    border-width: 1px;
    border-radius: 32px;

    innerText := Text {
        text: "TIMER";
        color:ta.has-hover ? #151B2E : #FCB647;
        x: (parent.width / 2) - (self.width / 2);
        y: (parent.height / 2) - (self.height / 2);
    }

    ta := TouchArea {
        clicked => { root.clicked(); }
    }
}
```

Punti salienti

Si possono avere più componenti nello stesso file

L'elemento top level di un componente ha id `root` implicito

Generazione delle liste tramite `for`

La comunicazione con il backend in C++/Rust avviene tramite singleton

Alcuni elementi grafici

Text: renderizza del testo, anche multilinea

Image: visualizza un'immagine. Le immagini sono integrate all'interno del binario finale

Rectangle: visualizza un rettangolo, anche stondato, con bordo e gradiente di riempimento

Stati e animazioni

Animazioni: keyword “animate” per definire le property da animare

Stati: un cambiamento di stato permette di cambiare property in gruppo

Transizioni: animazioni legate al cambio di stato

Elementi logici e di input

TextInput: consente di inserire del testo tramite tastiera

Layout: Horizontal, Vertical, Grid. Servono per posizionare in modo automatico gli elementi

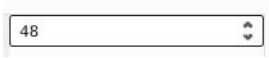
TouchArea: gestisce l'input touch o mouse su una determinata area

Widget standard

Button

Checkbox

Spinbox



Slider

ComboBox

Widget standard (cont.)

LineEdit, TextEdit

ScrollView, ListView

TabWidget

Diagramma di architettura

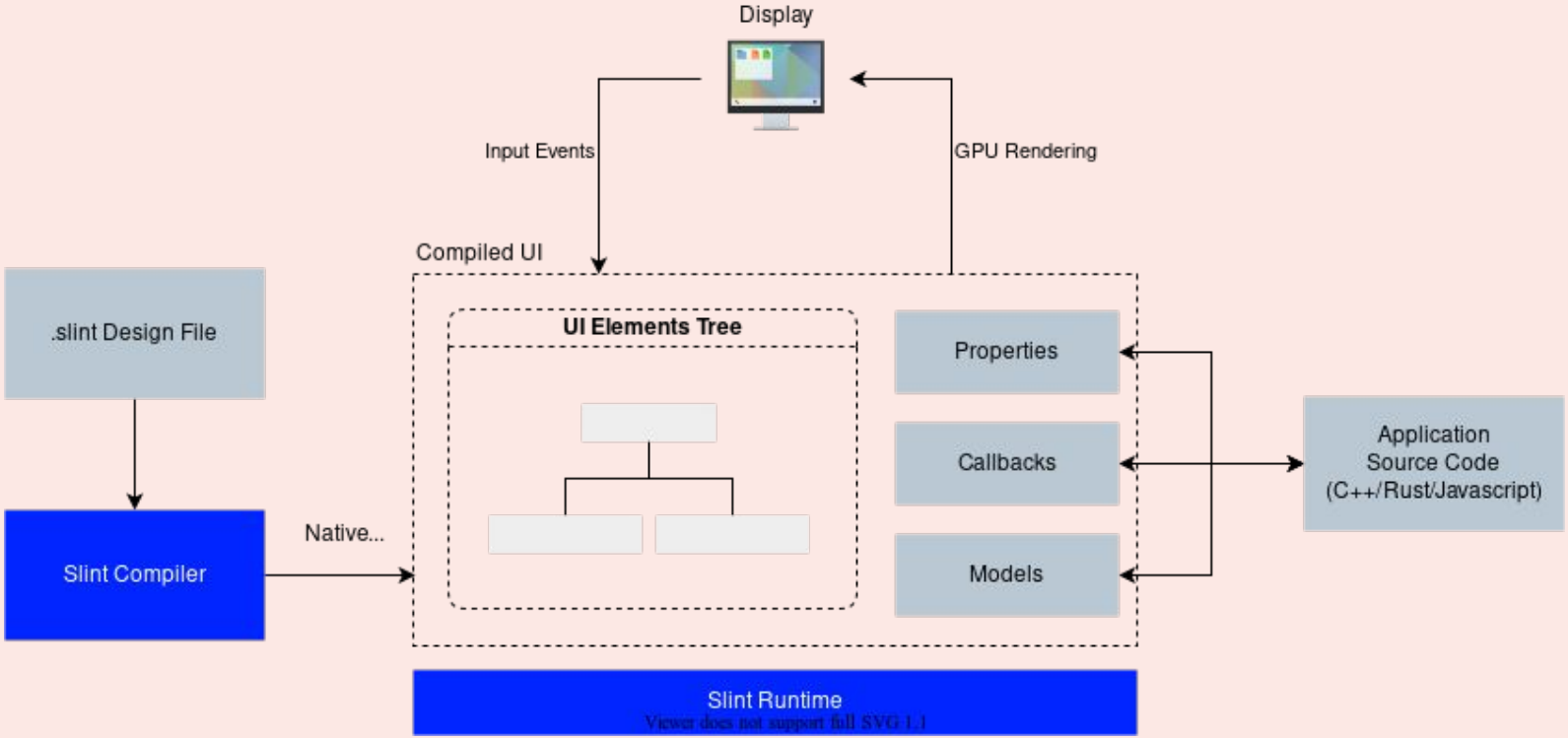


Diagramma di architettura

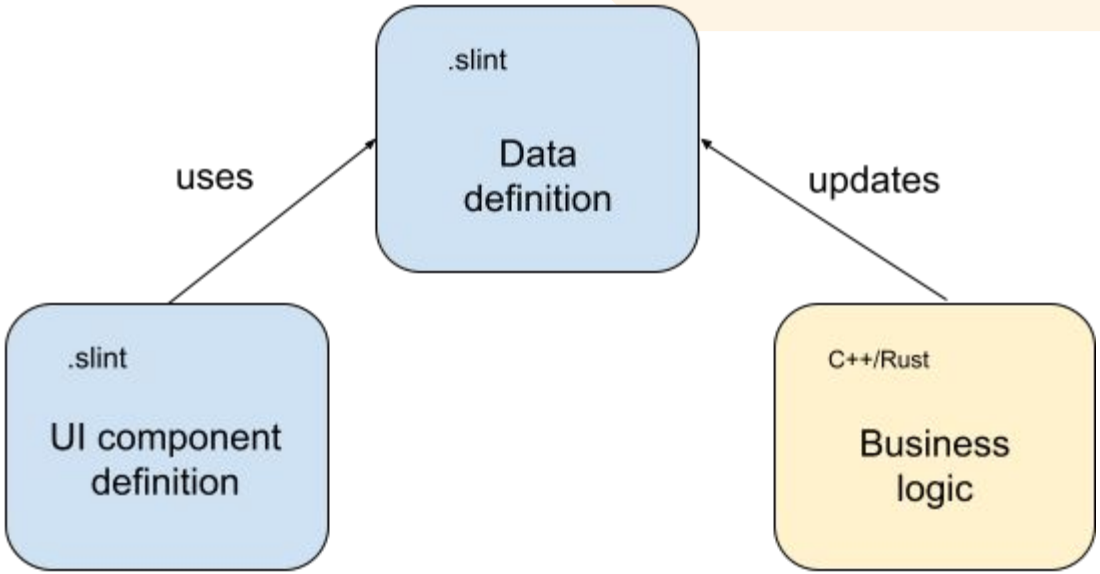
.slint design files: file di UI. Compilati ahead-of-time. Le espressioni sono funzioni pure che il compilatore può ottimizzare

UI elements tree: oggetti grafici organizzati ad albero. Sono gli oggetti renderizzati

Slint runtime: libreria di supporto al linguaggio *.slint*. Comprende un loop degli eventi, gestione input, timer… Deve girare nel main thread.

Application code: business logic dell'applicazione. Implementa modelli, callbacks e aggiorna le property degli oggetti grafici

Collegamento tra UI e logica applicativa



Collegamento tra UI e logica applicativa (cont.)

La comunicazione tra slint runtime e business logic avviene tramite singletons

Le strutture dati necessarie alla UI sono definite in file .slint

La business logic accede ai dati solo dal loop principale

Si passano (weak) handle a tutti gli oggetti che aggiornano la UI

Gli handle hanno funzioni per schedulare update nel loop principale

Punti di forza

Si può scrivere il codice del backend in C++, NodeJS o Rust

Linguaggio dichiarativo type-safe per la UI

Hot reload dei file .slint

Punti di forza

Import di immagini e font

Attenzione all'esperienza sviluppatore (es. immagine Docker per cross compilazione, plugin VSCode)

Debolezze

Mancanza di un tool di traduzioni

Nessun supporto accessibilità

Look and feel della piattaforma solo con il backend Qt

Testo non selezionabile su desktop

Debolezze

No deploy su mobile

Compilatore C++ 20 richiesto

Ancora sotto attivo sviluppo

Alcuni numeri

Cargo UI (v. 91bee62) su Ubuntu desktop 20.04, rustc 1.59.0

Size (debug, stripped): 40 Mb
Size (release, stripped): 14 Mb

Occupazione di RAM: 45 Mb

Target per MCU: Raspberry Pi Pico (RP2040 with 264KB RAM, using a 320x240 pixel screen, pixels transferred via SPI)

Conclusioni

Toolkit molto giovane

Tecnologia interessante

Permette di scrivere backend in Rust

Interessante per microcontrollori

Luca Ottaviano

lottaviano@develer.com

Vuoi rimanere aggiornato sugli eventi Develer?
Seguici nei nostri canali social:

develer | 20

www.develer.com