

GUI + CRYPTO IN RUST



Stanko Krtalić Rusendić

 [github.com/Stankec](https://github.com/Stankec)

 [@monorkin](https://twitter.com/monorkin)

01

DONO



`github.com/dono-app`



Panos



Vincent



Stanko



Petar

# FEATURES

- Password derivation
- Secure
- Simple to remember
- No third party
- No servers
- Free & under GPL

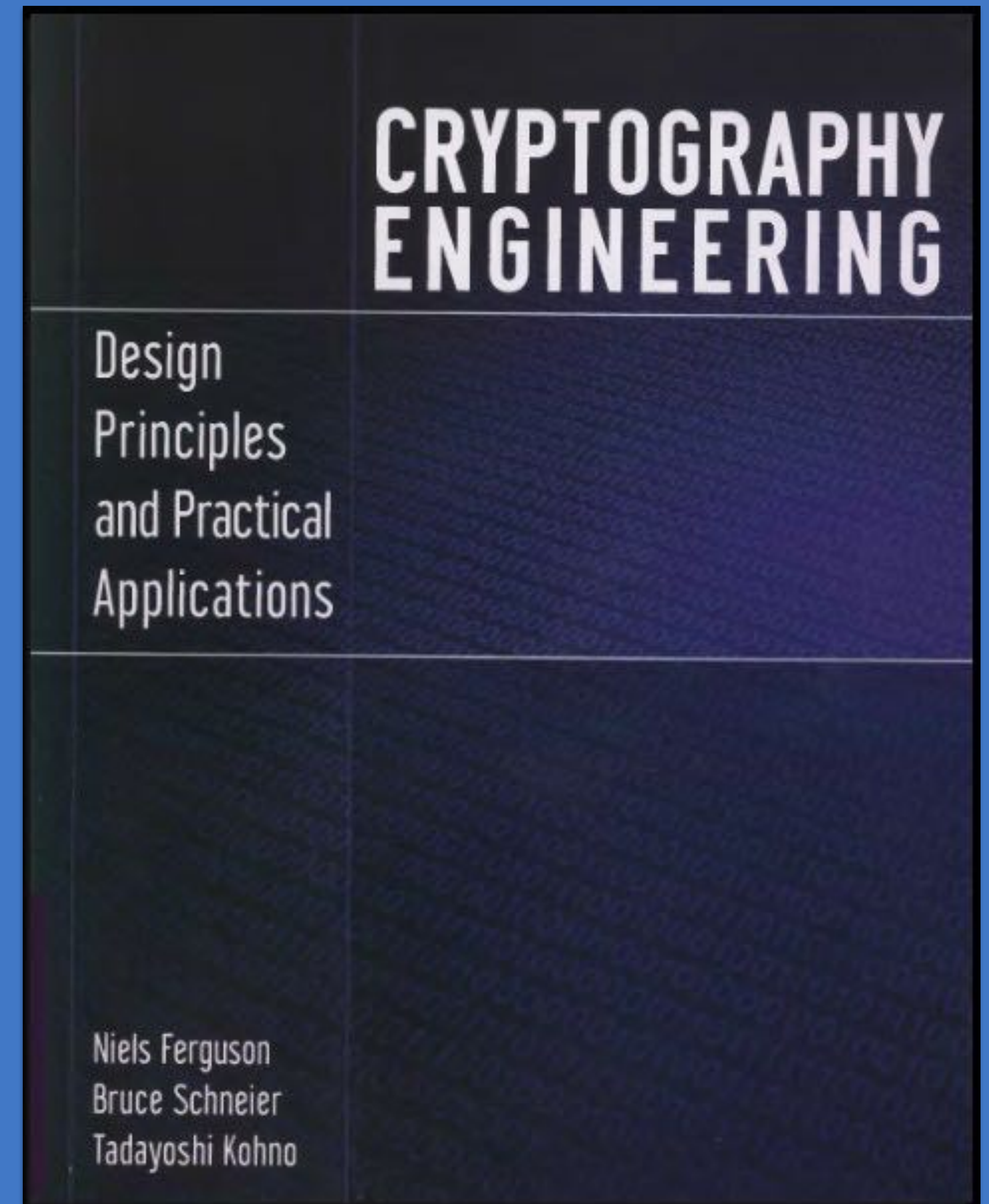
dono.tech

02

CRYPTO



Don't use low-level  
languages

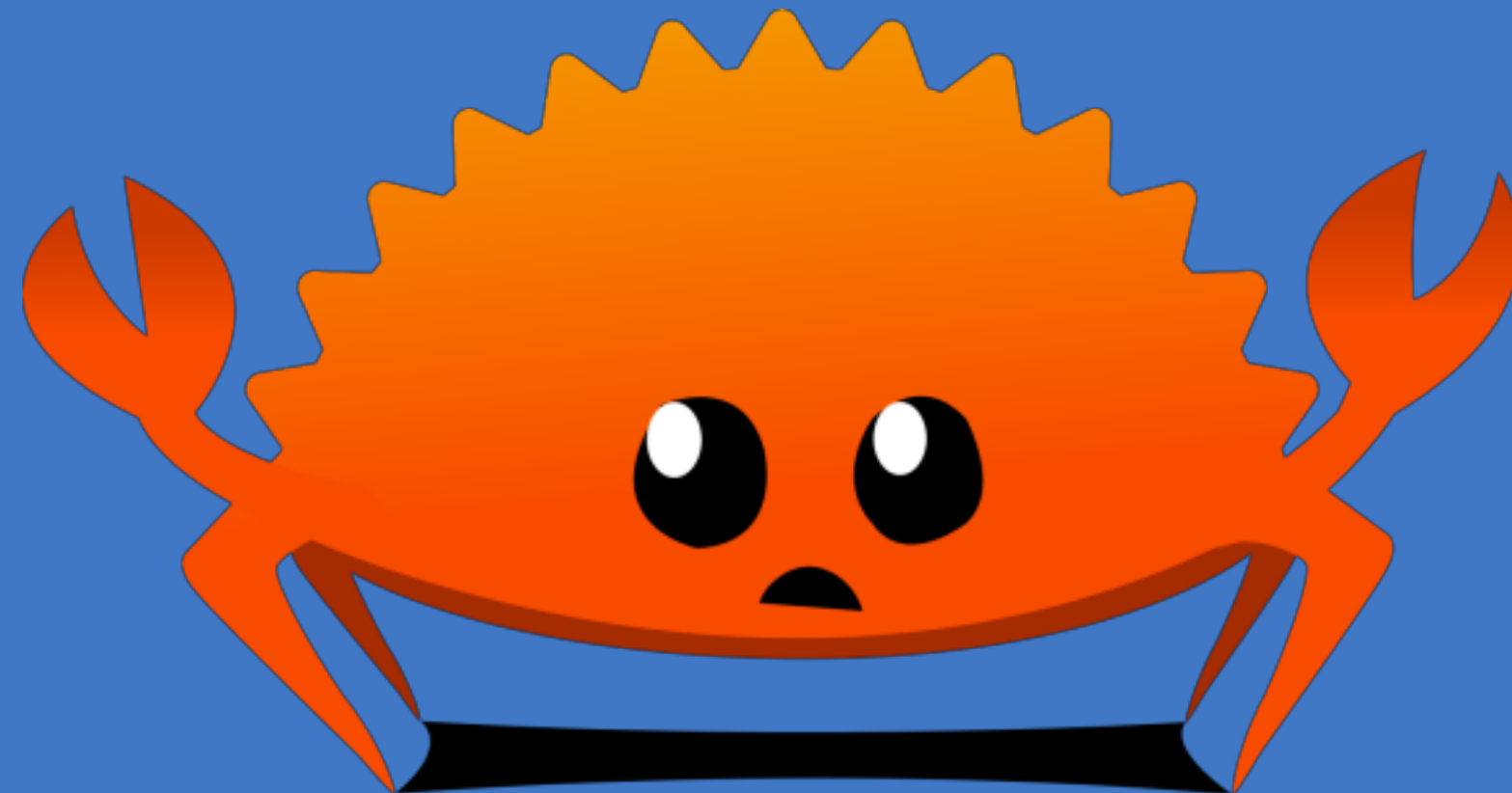


- Buffer overflow
- Pointer management
- Memory leaks
- Data races





- ~~Buffer overflow~~ → Results
- ~~Pointer management~~ → Handled automatically (to a degree)
- ~~Memory leaks~~ → Lifetimes
- ~~Data races~~ → Ownership



# RESULTS

```
int password_char_at_index(int i)  fn password_char_at_index(i: i32)
{                                  {
    // i = 17                      # i = 17
    char password[5] = "pass";     let pwd = "pass";
    return password[i];             pwd.chars().nth(i).unwrap();
}
```



Returns whatever is at memory  
location `password + i`



Returns the character at index `i`  
or raises an error

# OWNERSHIP

```
int password()  
{  
    char password[5] = "pass";  
    // moves the pointer  
    do_something(&password);  
    return password;  
}
```



Runs just fine

```
fn password() -> &str  
{  
    let pwd = "pass";  
    do_something(pwd).unwrap();  
    pwd  
}
```



Won't compile

# OWNERSHIP

```
fn password() -> &str
```

```
{
```

```
    let pwd = "pass";
```

```
    do_something(pwd).unwrap();
```

```
    pwd
```

```
}
```

ownership of `pwd` is passed to `do\_something`

pwd is referenced by a scope that doesn't own it



# OWNERSHIP

```
fn password() -> &str
{
    let pwd = "pass";
    let pwd = do_something(pwd).unwrap();
    pwd
}
```



Runs just fine

# LIFETIMES

```
node *build_chain(int d, node *p)
{
    if (d == 0) return p;
    struct node *n =
        malloc(sizeof(struct node));
    p->n = n;
    return build_chain(d - 1, p);
}
```

```
struct node *a =
    malloc(sizeof(struct node));
free(build_chain(10, a));
```

```
fn build_chain<'a>(d: i32, p: 'a
Node)
{
    if d == 0 {
        return p;
    }
    mut let n = Node::new();
    n.p = p;
    build_chain(d - 1, n)
}
mut let a = Node::new();
build_chain(10, a);
```

[github.com/dono-app/dono-crate](https://github.com/dono-app/dono-crate)  
[github.com/DaGenix/rust-crypto](https://github.com/DaGenix/rust-crypto)

OS


GUI

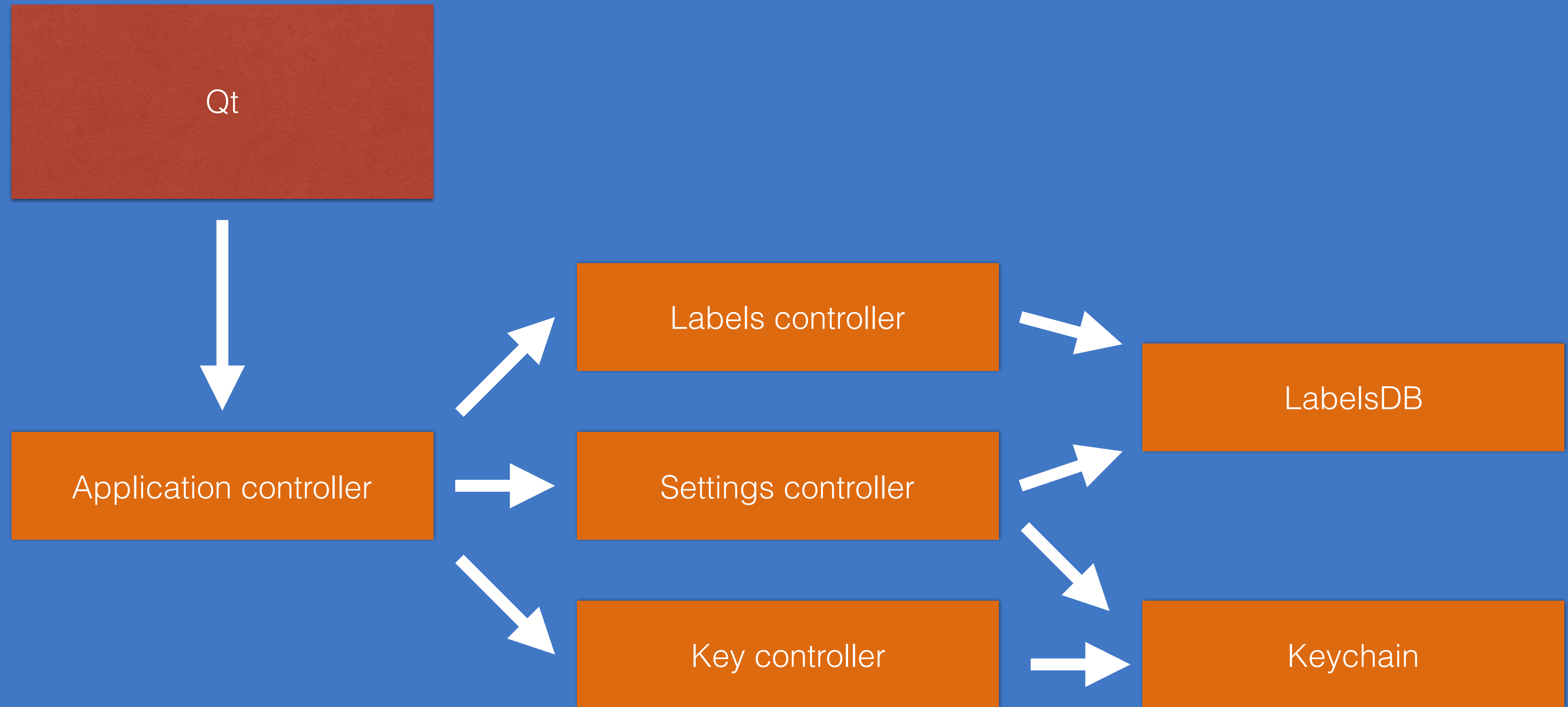


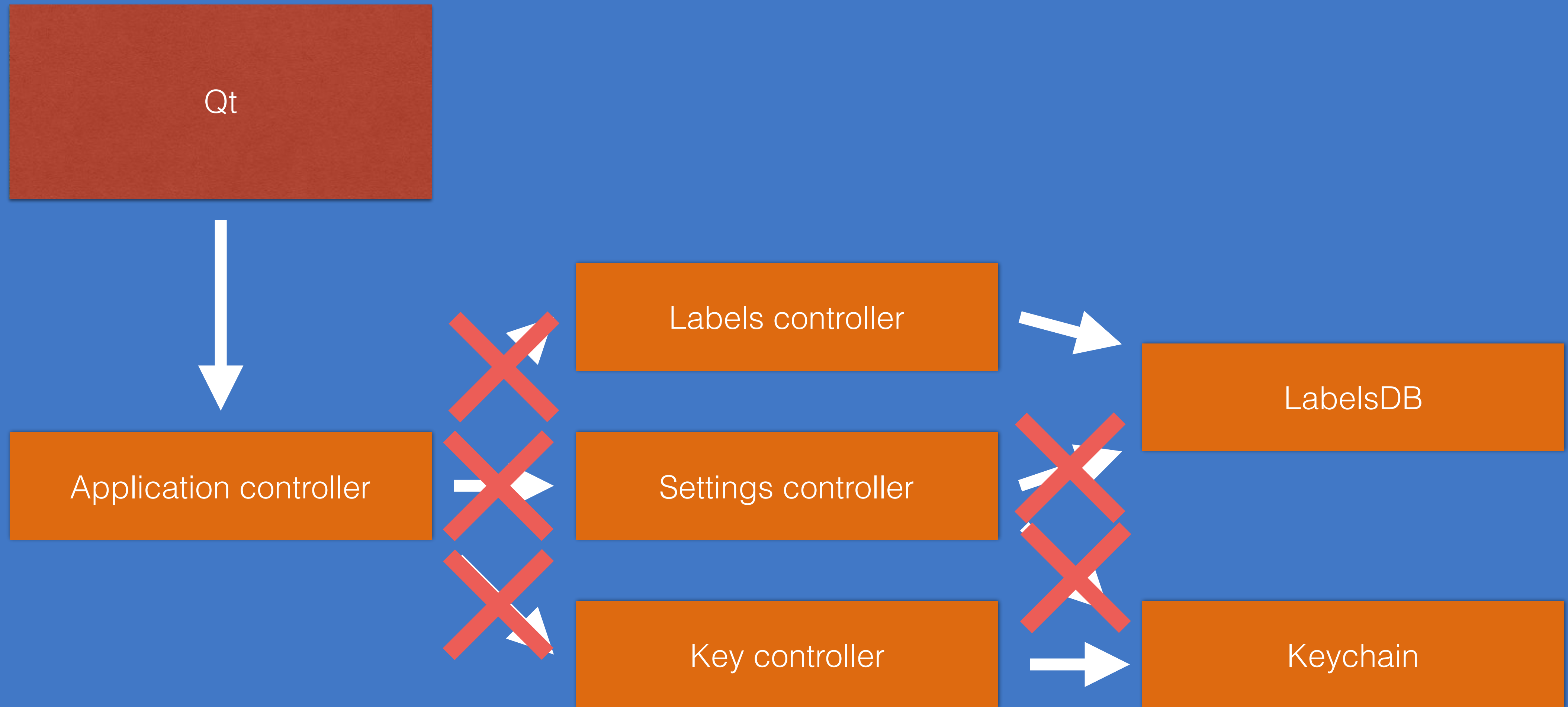
# github.com/kud1ing/awesome-rust

## GUI

### [gui]

- [PistonDevelopers/conrod](#) — An easy-to-use, immediate-mode, 2D GUI library written entirely in Rust build failing
- Cocoa
  - [kylewlacy/sorbet-cocoa](#) — build unknown
  - [servo/cocoa-rs](#)
- IUP
  - [dcampbell24/iup-rust](#) — IUP bindings build passing
  - [Kiss-ui](#) — a simple UI framework built on IUP build passing
- GTK+ [gtk]
  - [gtk-rs/gtk](#) — GTK+ bindings 
- ImGui
  - [imgui-rs](#) - Rust bindings for ImGui build passing
- libui
  - [pcwalton/libui-rs](#) — libui bindings build unknown
- ncurses [ncurses]
  - [jeaye/ncurses-rs](#) — ncurses bindings build passing
- [saurvs/nfd-rs](#) — Open native UI file dialogs in Linux, OS X and Windows
- Qt
  - [cyndis/qmlrs](#) — QtQuick bindings build passing
  - [kitech/qt.rs](#) — Qt5 bindings build unknown
  - [rust-qt](#) —
  - [White-Oak/qml-rust](#) - QML bindings





GUI is by definition loosely coupled

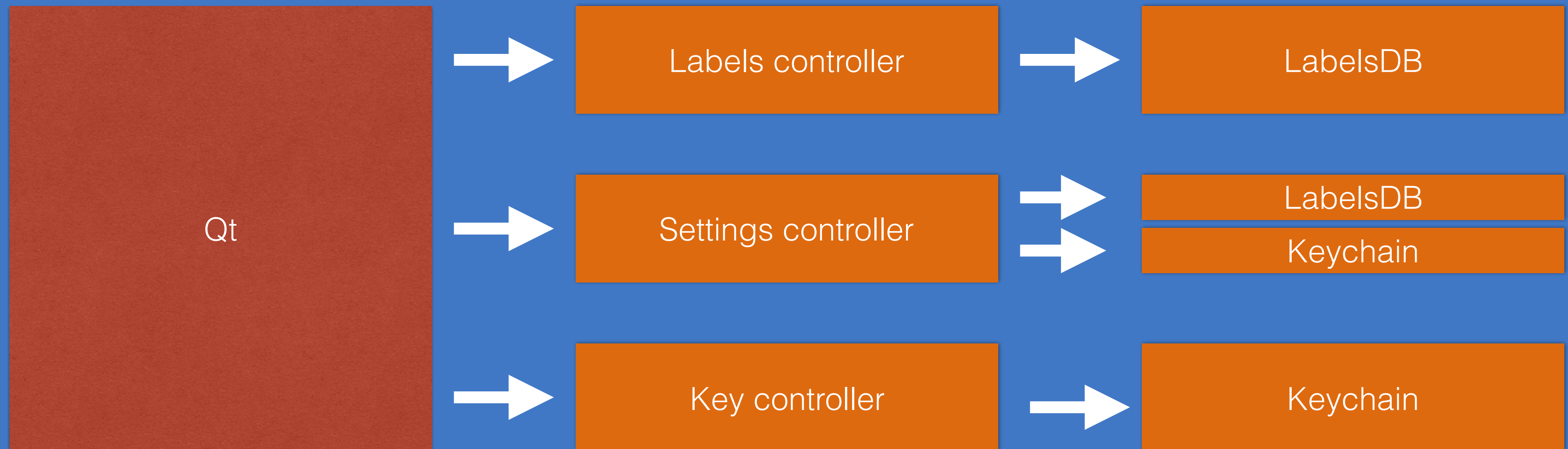


```
155
156 function createLabel() {
157     if (!labelInput.acceptableInput) {
158         return;
159     }
160
161     var serialized_object = newLabelController.save(labelInput.text);
162     var object = JSON.parse(serialized_object);
163
164     if (object === true) {
165         goBack();
166     } else {
167         labelInputValidationMessages.text = object.description;
168     }
169
170     createLabelActionButton.enabled = false;
171 }
172
```



We hope this exists

No `unsafe` blocks

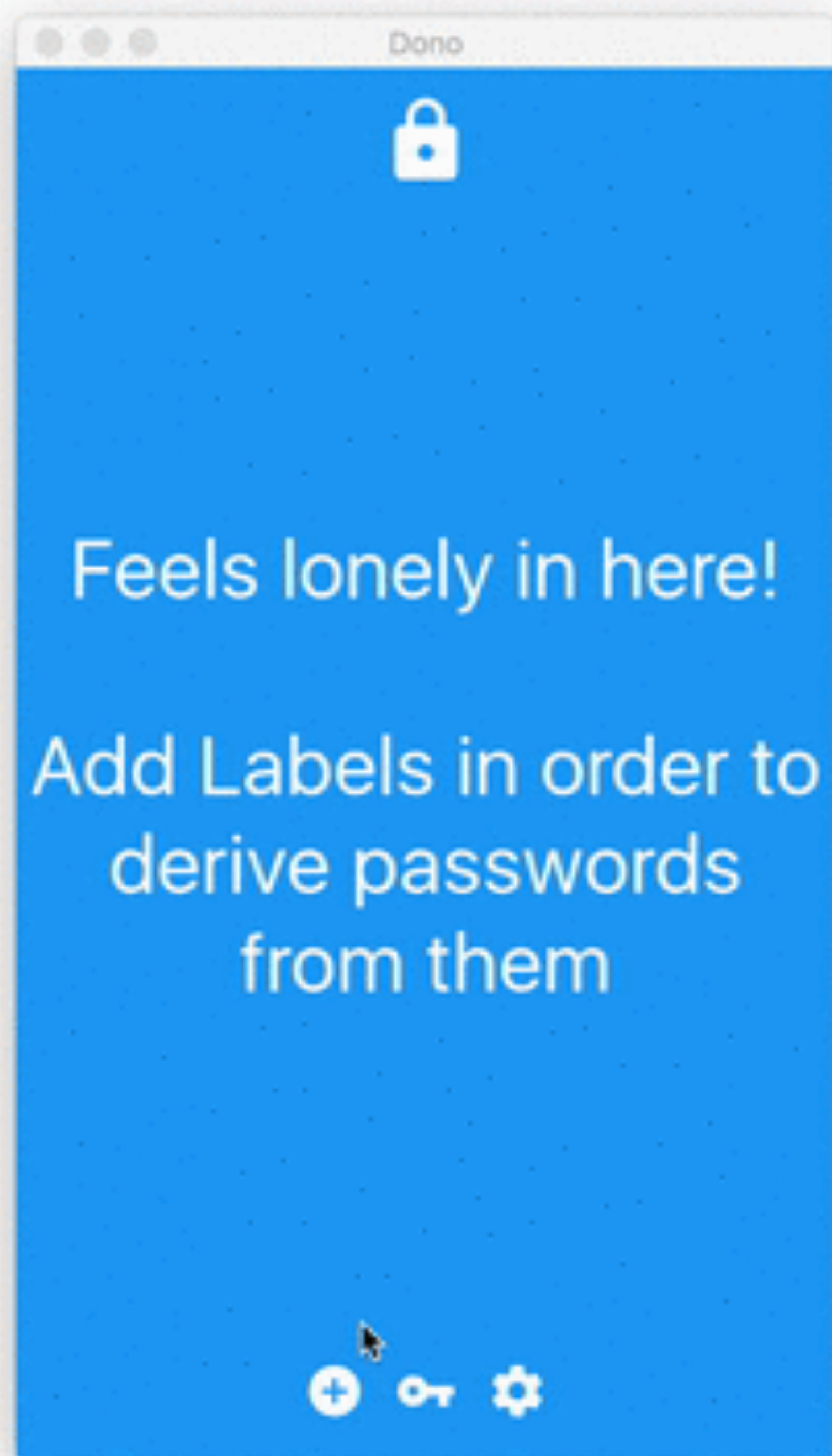


```
fn main() {  
    let mut engine = qmlrs::Engine::new();  
    let mut dono_gui = dono_gui::DonoGui::new();  
  
    engine.set_property("settingsController", dono_gui.settings_controller);  
    engine.set_property("newLabelController", dono_gui.new_label_controller);  
    engine.set_property("labelsController", dono_gui.labels_controller);  
  
    engine.load_url("qrc:LabelsIndexView.qml");  
    engine.load_url("qrc:qml/NewLabelView.qml");  
    engine.load_url("qrc:/qml/SetKeyView.qml");  
    engine.load_url("qrc:/src/qml/SettingsView.qml");  
    engine.load_url("qrc://src/qml/Screen.qml");  
    engine.load_url("qrc://src/qml/Application.qml");  
  
    engine.exec();  
}
```



# PROBLEMS

- Object duplication
- Convoluted architecture
- Prototypal inheritance
- Dynamic compilation
- Asset bundling



[github.com/dono-app/dono-linux/tree/develop](https://github.com/dono-app/dono-linux/tree/develop)

CONCLUSION

GUI



Crypto





# CRYPTOGRAPHY ENGINEERING

Design  
Principles  
and Practical  
Applications

Niels Ferguson  
Bruce Schneier  
Tadayoshi Kohno

