ROSEdu Summer of School

FinTP Configuration GUI

Participant: Macavei Andrei Gabriel

Mentor: Gabriel Stanciu (Allevo)

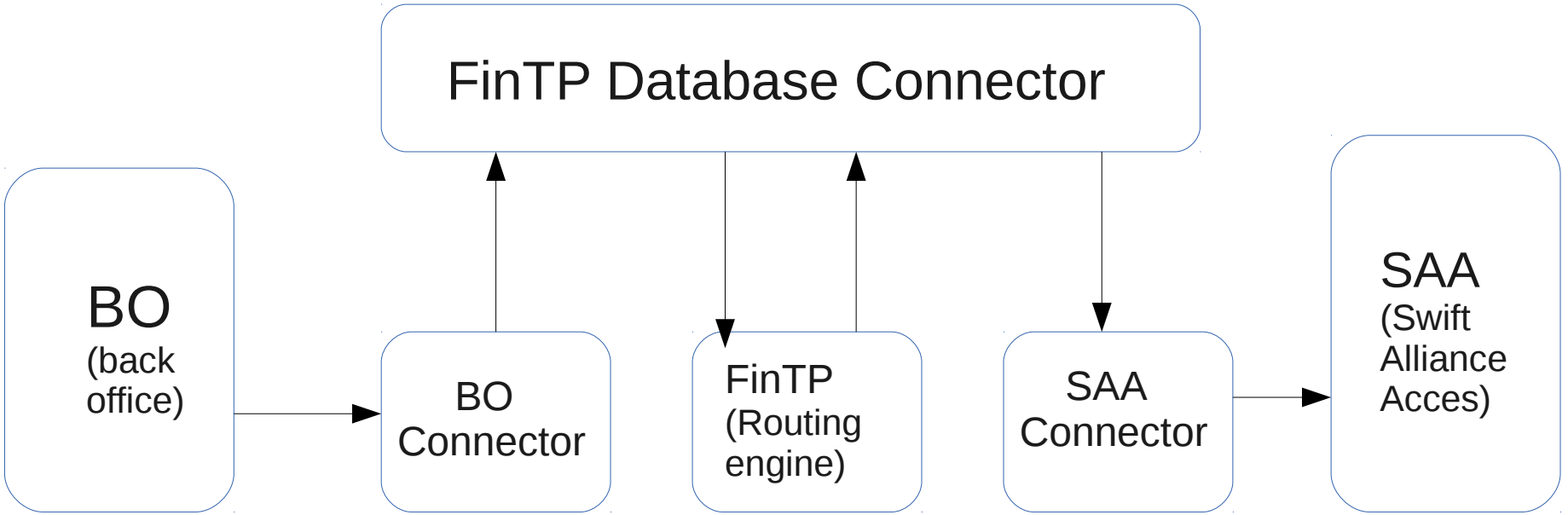
Ce este FinTP?

-o alternativa open-source pentru tranzactiile financiare-
(Made by Allevo)

Ce este FinTP Config GUI?

-o unealta grafica pentru configurarea FinTP-ului-
(Made by me)

Schema generala pentru aplicatia bancara FinTP (in particular ea este mult mai complicata)



Configurarea elementelor de tip “Connector” ale aplicatiei:

- se folosesc fisiere XML.
- aceste fisiere au devenit in timp destul de complicate.
- utilizatorul aplicatiei trebuie sa cunoasca toate “cuvintele cheie” folosite de developer in scrierea lor

Solutie: Crearea unei interfata grafice (GUI) ce se va configura dinamic si va transpune elemente din XML(tag-uri) in elemente grafice (widgets)

Maparea tag-urilor din XML in widgets-uri din GUI:

- tag pentru tip principal de filtre (XML) → Tab widget (GUI)

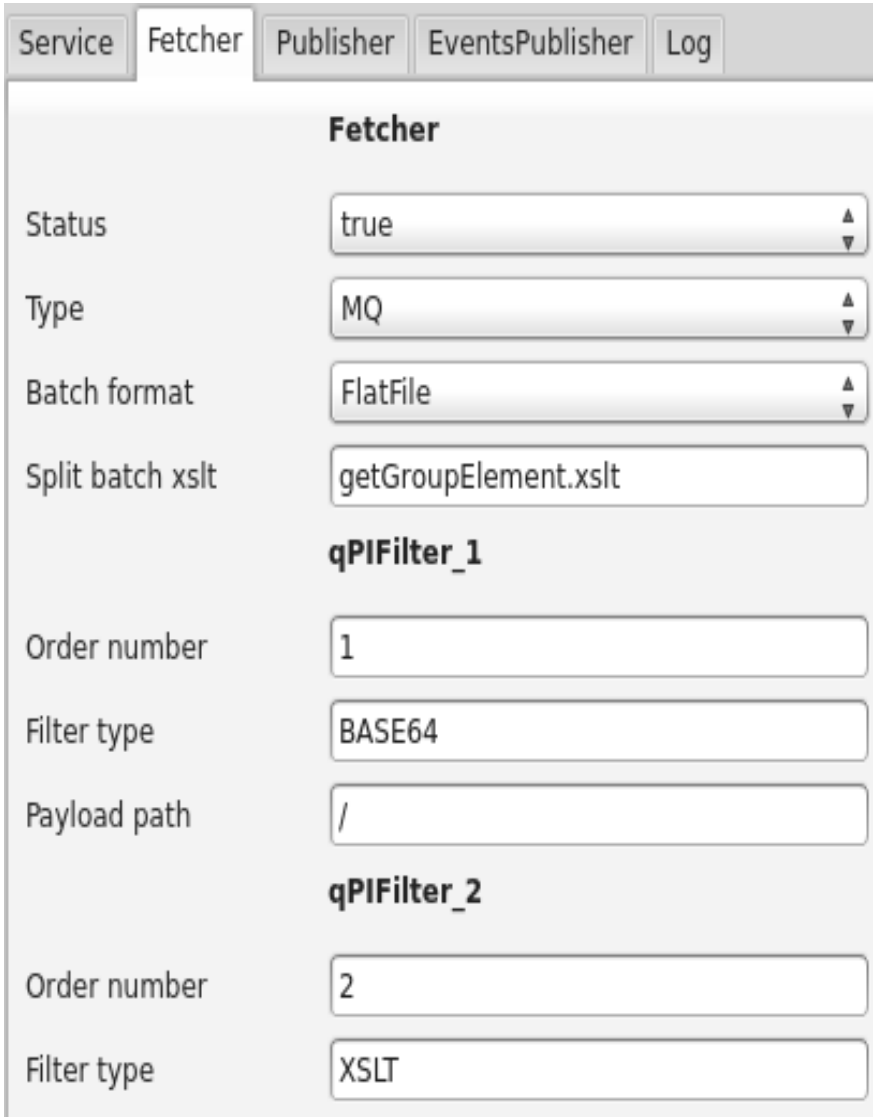
- copiii directi ai tipului de filtre (XML) → GroupBox (sectiune de widgets in GUI)

- mostenitori ai tag-ului de filtre (XML) → diferite widgets-uri in GUI : Label, TextBox, ComboBox

Un exemplu de mapare:

Tag XML (tip de filtre) → TAB Widget in GUI

```
▼<Fetcher>
  ▼<Fetcher>
     <key name="on" list="true,false"
     <key name="type" list="MQ,File,DB
     <key name="AppToMQSeries.BatchMar
     <key name="AppToMQSeries.BatchXsl
   </Fetcher>
  ▼<qPIFilter_1 visible="false">
     <key name="order" list="" alias="
     <key name="type" list="" alias="F
     <key name="xPath" list="" alias="
     <key name="keepOriginal" list=""
   </qPIFilter_1>
  ▼<qPIFilter_2 visible="false">
     <key name="order" list="" alias="
     <key name="type" list="" alias="F
     <key name="xsltFile" list="" alia
   </qPIFilter_2>
  ▼<qPIFilter_3 visible="false">
     <key name="order" list="" alias="
     <key name="type" list="" alias="F
     <key name="templateFile" list=""
   </qPIFilter_3>
  ▼<qPIFilter_4 visible="true">
     <key name="order" list="" alias="
```



Pe langa aceste lucruri, aplicatia trebuie sa poata fi folosita pe orice platforma.

Tehnologii utilizate:

- C++
- Bibliotecile Qt (cross-platform)
- Qt Creator (IDE scris in Qt)

Important Fact:
Qt-Creator vine cu un tool prin care poti creea interfata vizual (drag and drop), la fel ca in Visual Studio.
Mai greu de folosit atunci cand ai o interfata dinamica !

Conclusion: Programatically Designed UI (hardcore)

Componente principale ale aplicatiei:

- Functia de parsare a XML-ului: ConfigUI::parseXML()
  Parseaza XML-ul si creeaza dinamic interfata grafica.

- Functia de salvare a XML-ului: ConfigUI::saveXML()
  Parcurge GUI-ul si il salveaza intr-un fisier XML cu aceeasi structura.

- Functia de deschidere a unui fisier nou: ConfigUI::openFile()
  - reseteaza interfata grafica
  - populeaza interfata cu elemente citite din noul XML

- Functie ce adauga noi filtre in interfata:
  - gaseste filtrele ce lipsesc si le adauga din XML
  - creeaza o fereastra(Frame) in care adauga filtrele gasite intr-un ListView.

Dificultati intampinate , mai ales la inceput:

- Cum transpun un fisier XML intr-o interfata?

R: Folosind un XML parser. Qt are clase specializate pentru acest lucru: QDomDocument (DOM parser) si QxmlStreamReader/Writer (SAX parser).

- Cum actualizez acel fisier XML?

R: Cea mai simpla si eficienta solutie gasita a fost sa implementez Save, ca find un SaveAs pe fisierul curent.
O alta solutie ar fi fost sa retin structura XML-ului si sa o updatez, iar apoi sa efectuez operatii complexe de stergere/inserare in fisier.

- Interactiunea utilizatorului cu interfata

R: Pentru tratarea evenimentelor, Qt foloseste mecanismul: SIGNALS and SLOTS.

Signals and Slots mechanism

SIGNAL → Semnalul declansat de o actiune a user-ului asupra unui widget din interfata.

→ Poate primi doar un parametru aditional predefinit, iar tipul lui trebuie sa corespunda cu cel al parametrului primit de SLOT

SLOT→ Functie ce preia un semnal si il prelucreaza

→ La fel ca la SIGNAL, daca vrei sa transmiti un parametru aditional functiei receptoare, trebuie sa aiba acelasi tip(predefinit) cu cel al semnalului.

```
QPushButton *m_acceptButton = new QPushButton("Ok");
connect(m_acceptButton, SIGNAL(clicked()), this, SLOT(addFilterToGui()));
QPushButton *m_cancelButton = new QPushButton("Cancel");
connect(m_cancelButton, SIGNAL(clicked()), this, SLOT(hideFrameBox()));
```

Problema intampinata:

- Ce faci cand ai mai multe widget-uri de acelasi tip si vrei sa foloseasca acelasi SLOT, dar fara sa umplii codul de functii duplicate.

R: Se foloseste o “mapare” semnale → slot

De exemplu, pentru a folosi aceeasi functie atunci cand utilizatorul selecteaza din menu “Save” sau “SaveAs” , singura diferenta fiind fisierul in care se salveaza.

```
QSignalMapper *m_sigMapper = new QSignalMapper(this);
connect(actionSave, SIGNAL(triggered()), m_sigMapper, SLOT(map()));
connect(actionSaveAs, SIGNAL(triggered()), m_sigMapper, SLOT(map()));
m_sigMapper->setMapping(actionSave, "save");
m_sigMapper->setMapping(actionSaveAs, "saveAs");
connect(m_sigMapper, SIGNAL(mapped(QString)), this, SLOT(saveXML(QString)));
```

Elemente adaugate recent la aplicatie

- Un widget (button) ce deschide o fereastra noua (FrameBox) din care utilizatorul poate adauga noi tipuri de filtre.

  Mecanismul implementat:
  - citeste din XML tipurile existente de filtre ce nu apar in GUI
  - afiseaza aceste filtre ca optiuni intr-o lista
  - in momentul in care userul selecteaza una sau mai multe filtre, acestea sunt adaugate dinamic in UI.

- (In lucru) Un alt buton ce permite stergerea acestor filtre din interfata.

Partea de demonstratie - Live