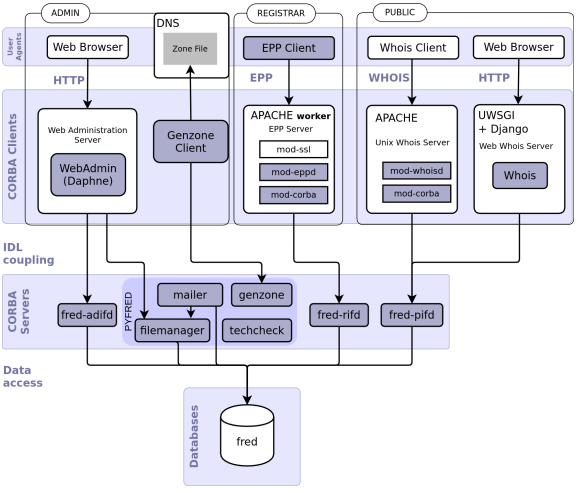
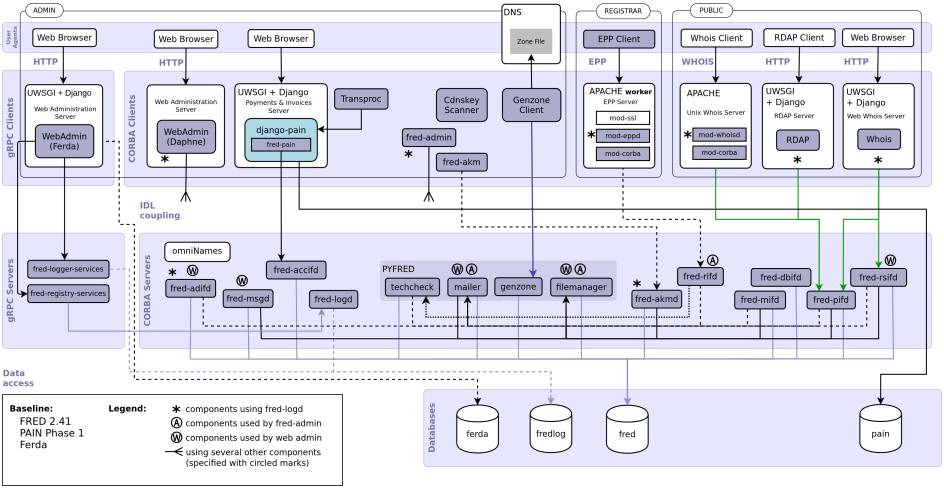
Změny v architektuře FREDa

Vlastimil Zíma • vlastimil.zima@nic.cz • 12. listopadu 2020

FRED 2008


ADMIN
DNS
REGISTRAR
PUBLIC
User Agents
Web Browser
Zone File
EPP Client
Whois Client
Web Browser
HTTP
EPP
WHOIS
HTTP
CORBA Clients
Web Administration Server
WebAdmin (Daphne)
Genzone Client
APACHE worker
EPP Server
mod-ssl
mod-eppd
mod-corba
APACHE
Unix Whois Server
mod-whoisd
mod-corba
UWSGI + Django
Web Whois Server
Whois
IDL coupling
CORBA Servers
PYFRED
mailer
genzone
fred-adifd
filemanager
techcheck
fred-rifd
fred-pifd
Data access
Databases
fred

FRED 2019


ADMIN
DNS
Zone File
REGISTRAR
PUBLIC
User Agents
Web Browser
Web Browser
Web Browser
EPP Client
Whois Client
RDAP Client
Web Browser
HTTP
HTTP
EPP
WHOIS
HTTP
HTTP
gRPC Clients
CORBA Clients
UWSGI + Django
Web Administration Server
WebAdmin (Ferda)
Web Administration Server
WebAdmin (Daphne)
UWSGI + Django
Payments & Invoices Server
django-pain
fred-pain
Transproc
Cdnskey Scanner
Genzone Client
fred-admin
fred-akm
APACHE worker
EPP Server
mod-ssl
mod-eppd
mod-corba
APACHE
Unix Whois Server
mod-whoisd
mod-corba
UWSGI + Django
RDAP Server
RDAP
UWSGI + Django
Web Whois Server
Whois
IDL coupling
gRPC Servers
CORBA Servers
omniNames
fred-logger-services
fred-registry-services
fred-adifd
fred-msgd
fred-accifd
fred-logd
PYFRED
techcheck
mailer
genzone
filemanager
fred-akmd
fred-rifd
fred-dbifd
fred-mifd
fred-pifd
fred-rsifd
Data access
Databases
ferda
fredlog
fred
pain
Baseline:
FRED 2.41
PAIN Phase 1
Ferda
Legend:
components using fred-logd
components used by fred-admin
components used by web admin
using several other components (specified with circled marks)

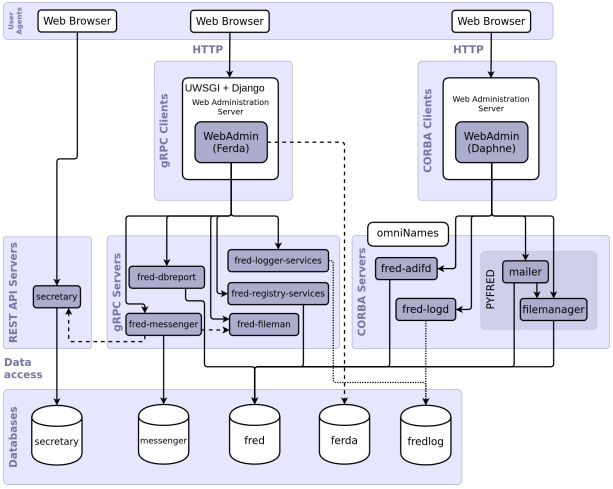
Odesílání zpráv

- Součást registru
  - Databáze registru 100 GB
  - Zprávy 35 GB
- Nejednotné zpracování
- Chybí správa šablon
- Omezená možnost konfigurace odesílačů
- Výkon
- Python 2.7

Refaktoring odesílání zpráv

- Oddělené systémy
  - šablony `secretary`
  - soubory `fileman`
  - zprávy `messenger`
- Sjednocení šablon
- Administrace šablon
- Flexibilní rozhraní
- Vlastní rozšíření
- Archivace

FRED 2020


User Agents
Web Browser
Web Browser
Web Browser
HTTP
HTTP
gRPC Clients
UWSGI + Django
Web Administration Server
WebAdmin (Ferda)
CORBA Clients
Web Administration Server
WebAdmin (Daphne)
omniNames
REST API Servers
secretary
gRPC Servers
fred-dbreport
fred-logger-services
fred-registry-services
fred-messenger
fred-fileman
CORBA Servers
fred-adifd
fred-logd
PYFRED
mailer
filemanager
Data access
Databases
secretary
messenger
fred
ferda
fredlog

Corba

- Zastaralá
- Složitá
  - Objektově orientovaná
- Správa paměti v C++
- Není na PyPI
- Problémy škálování

Náhrada Corby

Požadavky

- Definované rozhraní
- Implementace v Pythonu i C++
- Výkonnost
- Škálování
- Open source

Možnosti

- gRPC
- REST
- XML-RPC
- JSON-RPC
- Vlastní řešení

gRPC

- Protocol Buffers
- Binární
- Chybí výjimky
- Aktivně vyvíjený
- Složitější typy
- Změny rozhraní
- Nešikovné rozhraní v Pythonu

Základní objekty

```
typedef string Uid;

struct Email {
    string recipient;
    string subject;
    string body;
};
```

```
message Uid {
     string value = 1;
}

message Email {
    string recipient = 1;
    string subject = 2;
    string body = 3;
}
```

```
// IDL
email = Email()  # TypeError: __init__() missing 3 required positional
                 # arguments: 'recipient', 'subject', and 'body'
email = Email('kryten@example.org', 'Subject', 'Body')
email.body = 'Other body'
email.body  # 'Other body'

// Protobuf
email = Email()
email = Email(recipient='kryten@example.org', subject='Subject',
              body='Body')
email.body = 'Other body'
email.body  # 'Other body'
```

Složitější objekty

```
typedef string Timestamp;

struct Envelope {
    Timestamp sent;
    Email message;
};
```

```
import "google/protobuf/timestamp.proto";

message Envelope {
    google.protobuf.Timestamp sent = 1;
    Email message = 2;
}
```

```
// IDL
envelope = Envelope(sent=datetime.now().isoformat(), message=email)
envelope.sent  # '2020-10-29T14:33:41.302113'
datetime.strptime(envelope.sent, '%Y-%m-%dT%H:%M:%S.%f')

// Protobuf
sent = Timestamp()
sent.FromDatetime(datetime.now())
envelope = Envelope(sent=sent, message=email)
envelope.sent.FromDatetime(datetime.now())
envelope.sent  # seconds: 1603982021
               # nanos: 302113000
envelope.sent.ToDatetime()
  # datetime.datetime(2020, 10, 29, 14, 41, 58, 302113)
```

```
// IDL
envelope.email = email
envelope.email.subject = 'Test'

// Protobuf
envelope.email = email  # AttributeError: Assignment not allowed to
  # field "message_data" in protocol message object.
envelope.email.subject = 'Test'
envelope.email.CopyFrom(email)
```

Výchozí hodnoty

```
// Protobuf
envelope = Envelope()
envelope.HasField('email')  # False
type(envelope.email)  # Email
envelope.email.subject  # ''
envelope.sent.ToDatetime()  # datetime(1970, 1, 1, 0, 0)
```

Výčty

```
enum Status {
    SUCCESS,
    FAIL
};
```

```
enum Status {
    SUCCESS = 0;
    FAIL = 1;
}
```

```
// IDL
SUCCESS  # SUCCESS
str(SUCCESS)  # 'SUCCESS'
SUCCESS._v  # 'SUCCESS'
getattr(Messenger, 'SUCCESS')  # SUCCESS

// Protobuf
Status.SUCCESS  # 1
Status.Name(Status.SUCCESS)  # 'SUCCESS'
Status.Value('SUCCESS')  # 1 # Status.SUCCESS
```

Základní rozhraní

```
typedef sequence<Uid> UidSequence

interface EmailMessenger {
    Uid send(in Email message);
    UidSequence list(in string recipient,
                     in string subject,
                     out long count)
};
```

```
uids, count = messenger.list(recipient, subject)
```

Základní rozhraní

```
message ListRequest {
    string recipient = 1;
    string subject = 2;
}
message ListReply {
    repeated Uid uids = 1;
    uint64 count = 2;
}

service EmailMessenger {
    rpc send (Email) returns (Uid) {}
    rpc list (ListRequest) returns (ListReply) {}
}
```

Výjimky

```
interface EmailMessenger {
    exception RecipientRequired {};

    Uid send(in Email message) raises (RecipientRequired);
};
```

- https://grpc.github.io/grpc/core/md_doc_statuscodes.html

```
service EmailMessenger {
    // Return INVALID_ARGUMENT if recipient is empty.
    rpc send (Email) returns (Uid) {}
}
```

Vlastní výjimky

```
message SendReply {
    message Data {
        Uid uid = 1;
    }
    message Exception {
        bool recipient_required = 1;
    }
    oneof DataOrException {
        Data data = 1;
        Exception exception = 2;
    }
}
```

Streaming

```
rpc list_emails (ListRequest) returns (stream Email) {}
rpc send_emails (stream Email) returns (SendReply) {}
```

```
for email in messenger.list_emails(request):
    print(email)

messenger.send_emails(email_iterator)
```

Děkuji za pozornost

Vlastimil Zíma • vlastimil.zima@nic.cz