Erlang/OTP In the wild: a governmental web application

Erlang Factory Lite Brussels 2016

Pieterjan Montens

// Introduction

- A little about myself
- What the application is and what it does

- How it started
- How it evolved
- What the benefits of using Erlang/OTP were
- In hindsight…

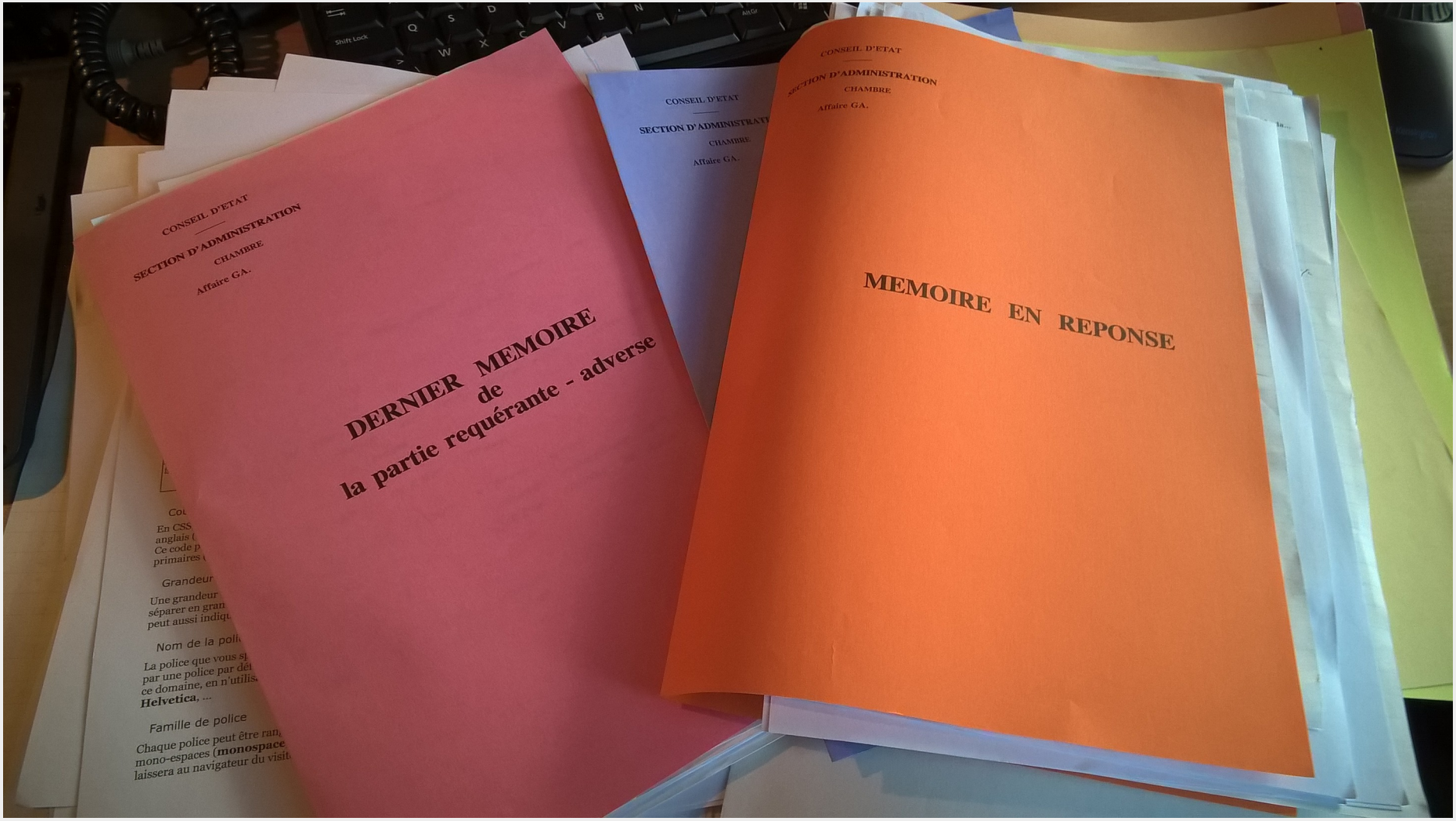
- Conclusion

And a demo somewhere in between

// Intro : A little about me

- Pieterjan Montens
  - born in **Bruges**, grew up in the **East Cantons**, lives in **Liège**, works in **Brussels**
- Jurist & self-taught software & system « engineer »
- Merged interest for **IT**, pleasure of **building stuff** and the desperate state of **justice computerization**
- Using **Erlang/OTP since 2008**, in Belgium & Switzerland
- Works at the **Council of State** in Brussels, occasional freelancer/consultant

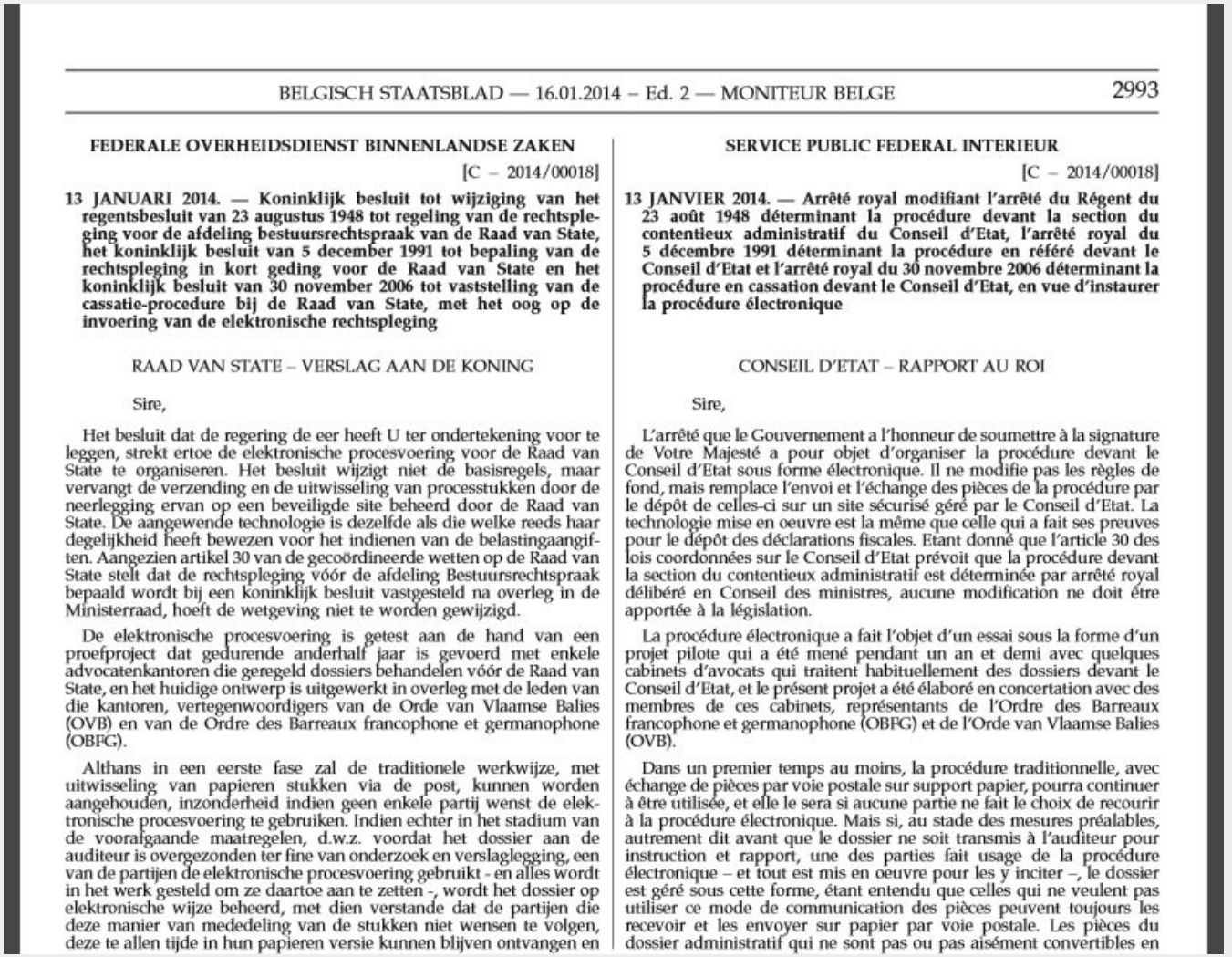
// Intro : the application



// Intro : What the application is & does

- e-proceedings platform, where lawyers and the court exchange documents electronically
- Sends e-mails, notifies urgent procedures via sms, has plenty of services, multiple browser/platform support, no java, prolog, eID authentication and signatures, ...
- May seem easy, and everyone thought it would be, but …
- Erlang delivered, and then some
- Achievement : application concepts transcribed in a Royal Decree

// Intro : What the application is & does : Royal Decree



BELGISCH STAATSBLAD — 16.01.2014 – Ed. 2 — MONITEUR BELGE 2993

FEDERALE OVERHEIDSDIENST BINNENLANDSE ZAKEN

[C − 2014/00018]

13 JANUARI 2014. — Koninklijk besluit tot wijziging van het regentsbesluit van 23 augustus 1948 tot regeling van de rechtspleging voor de afdeling bestuursrechtspraak van de Raad van State, het koninklijk besluit van 5 december 1991 tot bepaling van de rechtspleging in kort geding voor de Raad van State en het koninklijk besluit van 30 november 2006 tot vaststelling van de cassatie-procedure bij de Raad van State, met het oog op de invoering van de elektronische rechtspleging

RAAD VAN STATE – VERSLAG AAN DE KONING

Sire,

Het besluit dat de regering de eer heeft U ter ondertekening voor te leggen, strekt ertoe de elektronische procesvoering voor de Raad van State te organiseren. Het besluit wijzigt niet de basisregels, maar vervangt de verzending en de uitwisseling van processtukken door de neerlegging ervan op een beveiligde site beheerd door de Raad van State. De aangewende technologie is dezelfde als die welke reeds haar degelijkheid heeft bewezen voor het indienen van de belastingaangiften. Aangezien artikel 30 van de gecoördineerde wetten op de Raad van State stelt dat de rechtspleging vóór de afdeling Bestuursrechtspraak bepaald wordt bij een koninklijk besluit vastgesteld na overleg in de Ministerraad, hoeft de wetgeving niet te worden gewijzigd.

De elektronische procesvoering is getest aan de hand van een proefproject dat gedurende anderhalf jaar is gevoerd met enkele advocatenkantoren die geregeld dossiers behandelen vóór de Raad van State, en het huidige ontwerp is uitgewerkt in overleg met de leden van die kantoren, vertegenwoordigers van de Orde van Vlaamse Balies (OVB) en van de Ordre des Barreaux francophone et germanophone (OBFG).

Althans in een eerste fase zal de traditionele werkwijze, met uitwisseling van papieren stukken via de post, kunnen worden aangehouden, inzonderheid indien geen enkele partij wenst de elektronische procesvoering te gebruiken. Indien echter in het stadium van de voorafgaande maatregelen, d.w.z. voordat het dossier aan de auditeur is overgezonden ter fine van onderzoek en verslaglegging, een van de partijen de elektronische procesvoering gebruikt - en alles wordt in het werk gesteld om ze daartoe aan te zetten -, wordt het dossier op elektronische wijze beheerd, met dien verstande dat de partijen die deze manier van mededeling van de stukken niet wensen te volgen, deze te allen tijde in hun papieren versie kunnen blijven ontvangen en

SERVICE PUBLIC FEDERAL INTERIEUR

[C − 2014/00018]

13 JANVIER 2014. — Arrêté royal modifiant l'arrêté du Régent du 23 août 1948 déterminant la procédure devant la section du contentieux administratif du Conseil d'Etat, l'arrêté royal du 5 décembre 1991 déterminant la procédure en référé devant le Conseil d'Etat et l'arrêté royal du 30 novembre 2006 déterminant la procédure en cassation devant le Conseil d'Etat, en vue d'instaurer la procédure électronique

CONSEIL D'ETAT – RAPPORT AU ROI

Sire,

L'arrêté que le Gouvernement a l'honneur de soumettre à la signature de Votre Majesté a pour objet d'organiser la procédure devant le Conseil d'Etat sous forme électronique. Il ne modifie pas les règles de fond, mais remplace l'envoi et l'échange des pièces de la procédure par le dépôt de celles-ci sur un site sécurisé géré par le Conseil d'Etat. La technologie mise en oeuvre est la même que celle qui a fait ses preuves pour le dépôt des déclarations fiscales. Etant donné que l'article 30 des lois coordonnées sur le Conseil d'Etat prévoit que la procédure devant la section du contentieux administratif est déterminée par arrêté royal délibéré en Conseil des ministres, aucune modification ne doit être apportée à la législation.

La procédure électronique a fait l'objet d'un essai sous la forme d'un projet pilote qui a été mené pendant un an et demi avec quelques cabinets d'avocats qui traitent habituellement des dossiers devant le Conseil d'Etat, et le présent projet a été élaboré en concertation avec des membres de ces cabinets, représentants de l'Ordre des Barreaux francophone et germanophone (OBFG) et de l'Orde van Vlaamse Balies (OVB).

Dans un premier temps au moins, la procédure traditionnelle, avec échange de pièces par voie postale sur support papier, pourra continuer à être utilisée, et elle le sera si aucune partie ne fait le choix de recourir à la procédure électronique. Mais si, au stade des mesures préalables, autrement dit avant que le dossier ne soit transmis à l'auditeur pour instruction et rapport, une des parties fait usage de la procédure électronique – et tout est mis en oeuvre pour les y inciter –, le dossier est géré sous cette forme, étant entendu que celles qui ne veulent pas utiliser ce mode de communication des pièces peuvent toujours les recevoir et les envoyer sur papier par voie postale. Les pièces du dossier administratif qui ne sont pas ou pas aisément convertibles en

http://www.etaamb.be/fr/2014000018

// Intro : What the application is & does : the « but »

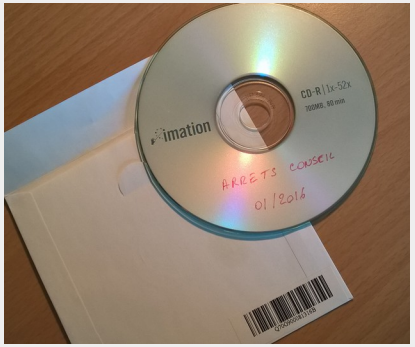
But…

- Complex access hierarchy and logic : some can see and/or sign, others can't or only partially, not always at the same time, not always for the same reasons, « your mileage may vary »
- Judiciary logic (it's logic except when it's not)
- E-mail notifications, SMS notification
- Specific business intelligence, jurisprudential procedure knowledge in the minds of experts who spend their lives honing and complexifying it
- Your users are 90 % lawyers, 10 % common people, with little or no IT background
- Electronic signatures are mandatory (and rightfully so)
- 24/7 operation
- Private data, strictly confidential information (enterprise IP, protected personal data, ...)
- Multiple interfaces, interfacings
- GUI's: at least 2
- Credibility of the institution is at stake
- Reality check : You do Q&A, support, maintenance & future develoments

How it started

// How it started 1 : the context

- State of justice computerization
  - Millions lost in failed attempts (phoenix)
  - Cut and paste : paper & scissors
  - Lost in time
  - Just yesterday, in 2016, ...



- Initial requirement : something simple to exchange files. In a secure way, for jurists.
- Developer gets “irrationally adventurous*” : uses Erlang/OTP

* He wasn't, though many benefits occured after the fact

// How it started 2 : hard

- Why Erlang ? (The good choice)
  - Step up from Php : less scripting, more programming
  - Joe Armstrong's book: mind bender.
    - Joe's anagram example
  - Erlang language expressiveness
- Why an Erlang web framework ? (The not so good choice)
  - Still good : Fast prototyping (3 months)
  - Bad : Limiting, working around it rather than with it
  - Hand-holding : you don't need/want to know OTP !
  - Erlang is not a templating language

Programming Erlang, by Joe Armstrong

```
perms([]) -> [[]];
perms(L)  ->
   [[H|T] || H <- L, T <- perms(L--[H])].
```

```
1> lib_misc:perms("123").
["123","132","213","231","312","321"]
2> lib_misc:perms("cats").
["cats", "cast", "ctas", "ctsa", "csat", "csta", "acts",
"acst","atcs", "atsc", "asct", "astc", "tcas", "tcsa", "tacs",
"tasc","tsca", "tsac", "scat", "scta", "sact", "satc", "stca",
"stac"]
```

> Code Interlude <

HTML templating with Erlang records

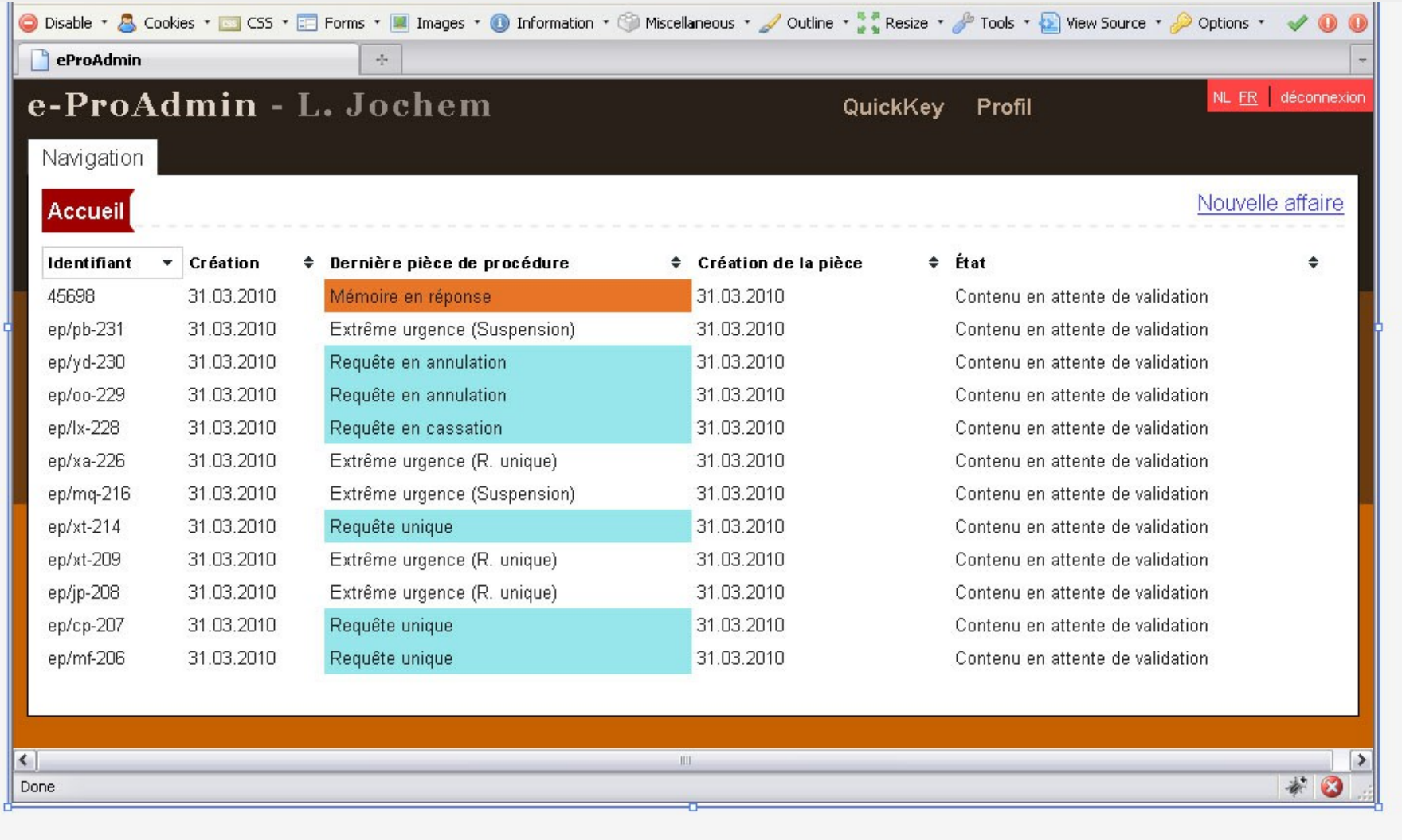
forms.erl + (~\Documents\eproadmin\eproadmin\src\tabs) - GVIM

Fichier Edition Outils Syntaxe Tampons Fenêtre Aide

```
11                                  ", ",
10                                 ?L(category)," ",?L(navigation_manager:get_cat_type(CatI))]),
 9      setFormData(Form_Id,newfiles),
 8      addDataToForm(Form_Id,procid,ProcI),
 7      addDataToForm(Form_Id,catid,CatI),
 6      { Form_Id,
 5      [
 4      #h3 { text= Title},
 3      #panel {class=meta_title, body=[ #literal { text = MetaTitle}]},
 2      #table { class=new_el_table, rows = [
 1         #tablerow { cells = [
 0                 #tablecell { class=header, text=?L(addfiles)},
 1                 #tablecell   { body=[#upload{ tag={proc_file_upload,{formid,Form_Id}}, show_button=false }]}
 2                 ]},
 3          #tablerow { cells = [
 4                 #tablecell { class=buttons, body = [
 5                     #button{ class=[Form_Id, " ", green], text=?L(submit), postback={form_event,{sendformdata,{Form_Id}}}
 6                     #button{ class=[Form_Id," ",red," ",cancel], text=?L(cancel), postback={close_tab,Form_Id}}]},
 7                 #tablecell { body = [
 8                     #table {id=Form_Id, class=file_table, rows=[]} ]}
 9                 ]}
10          ]}
11      ]}.
12
13  make_submitted(Type,ObjectId,FormId) ->
14      ProcI = if
15                 is_record(ObjectId,id) == false -> #id{id=ObjectId};
16                 ObjectId#id.procid == undefined -> ObjectId;
17                 true     -> #id{id=ObjectId#id.procid} end,
18
19      ?PRINTIT(ProcI),
20      case Type of
21          newproc ->
22                 [
23                 #h3 {text=?L(procedurecreated)},
24                 #p{},
25                 #span { text=?L(proctempidis)},
26                 #span { class=proc_id, text=ProcI#id.id},
27                 #p{},
28                 #button{ class=close, text=?L(close), postback={close_tab,FormId}}
29                 ];
```

forms.erl [+] 116,62 44%

// How it started 3 : how it looked

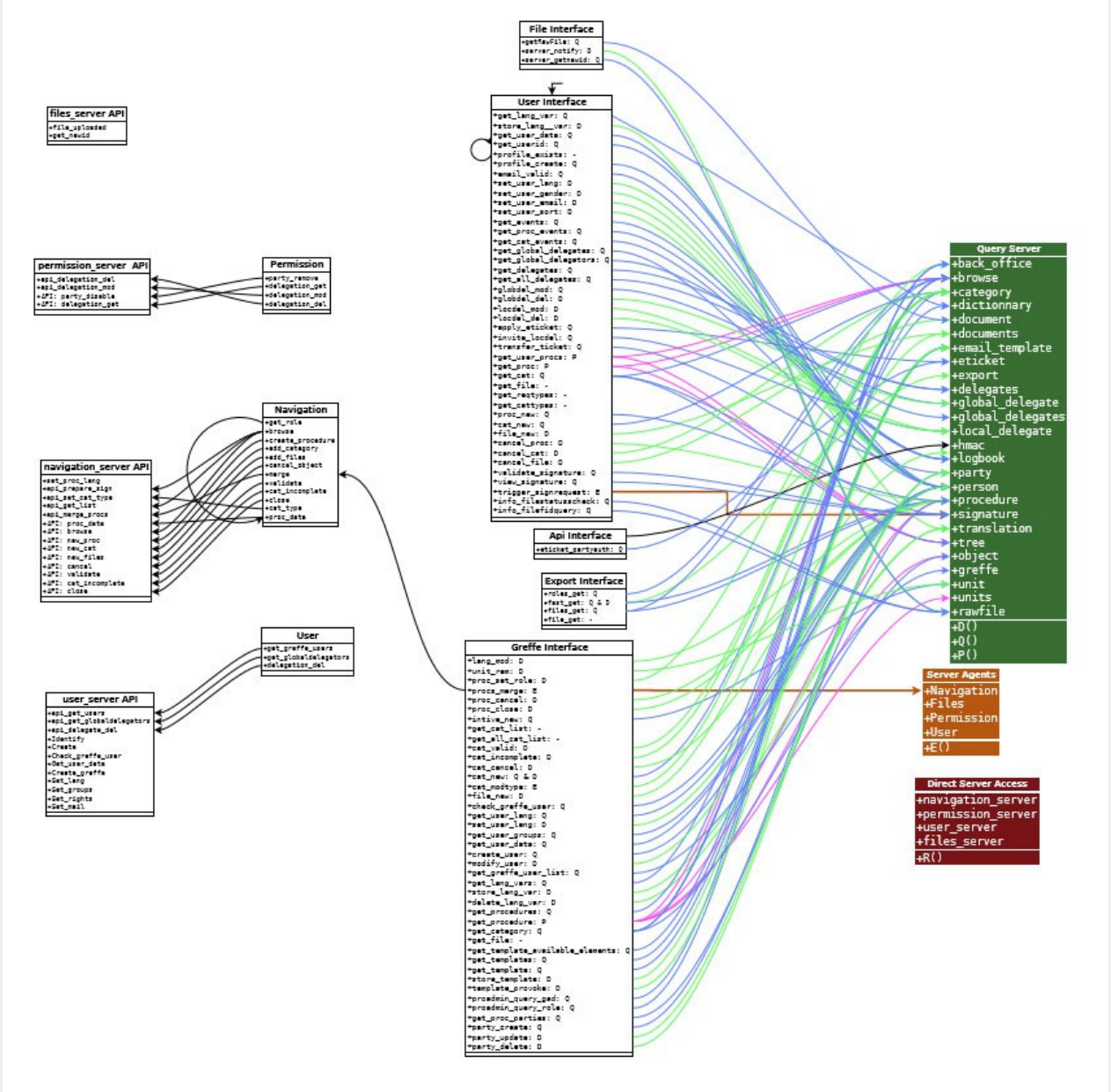


Early benefits :

- Refactoring is fun (pattern matching & atoms)
- Update application code & mnesia data structures without losing data, from day 1
- Migration to separate OTP applications, mochiweb server, true parallelism
- Organic evolution to quasi-DSL, improved code re-use, simplification

// How it started 5 : refactoring web interface calls

- Central column: web API
- On the left: old, monolithic server application calls
- On the right: rpc casts, synchronous and asynchronous calls and yields to a dynamic *simple_one_for_one* OTP gen_server instance



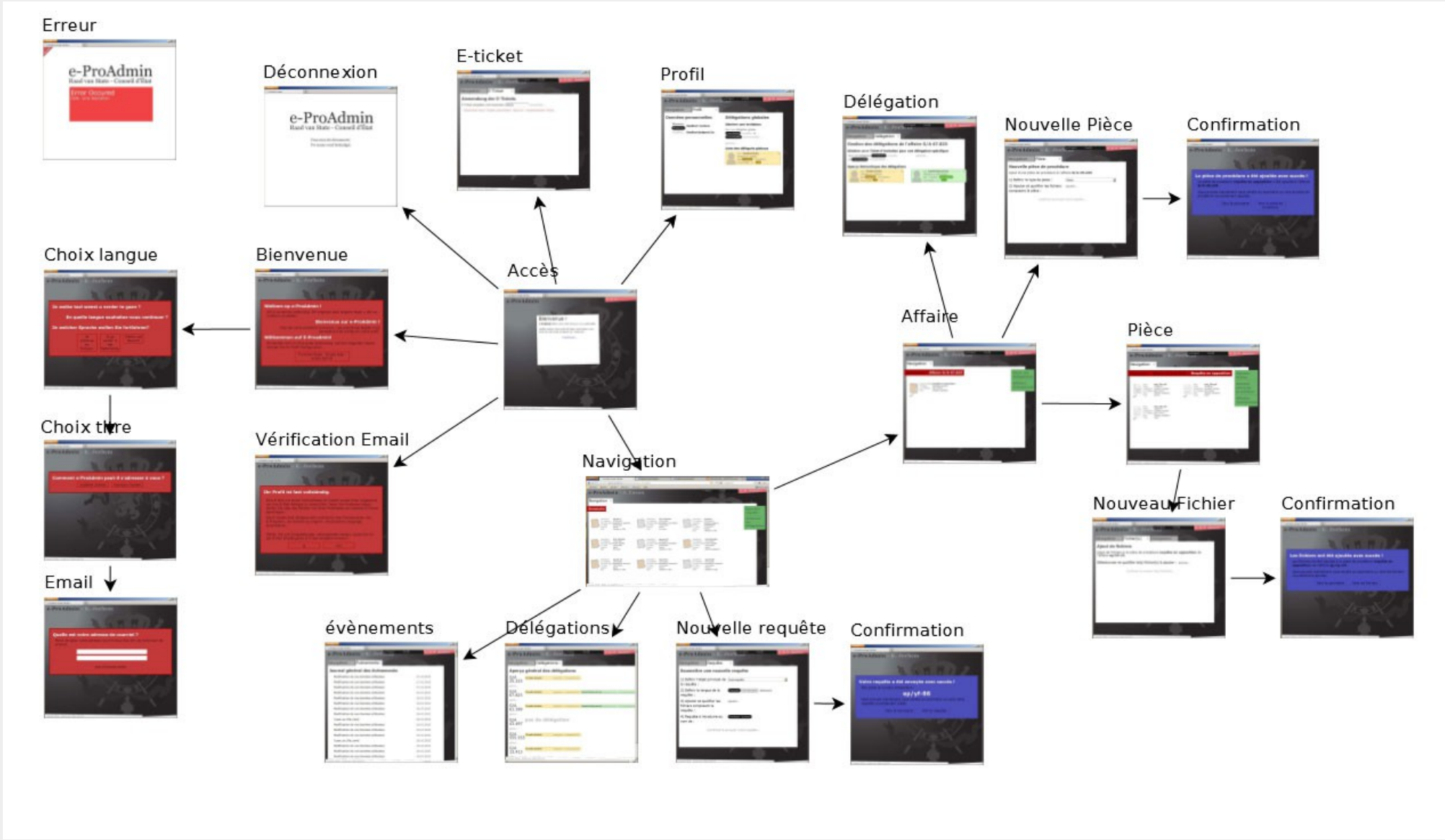
How it evolved

// How it evolved 1 : the birth of a platform

- The application isn't just a server. It's a platform. A domain specific database.
- Multiple services : REST (HATEOAS) API, sockets, e-mails, sms, dropbox-like client, …
- Two layers (independent OTP applications), distributed application core
- Web clients, simple & efficient : HTML, JS, PHP, CSS
  - Functional Js is fun : promises, web workers, web sockets, ...

Demo break

// How it evolved 2 : screen map

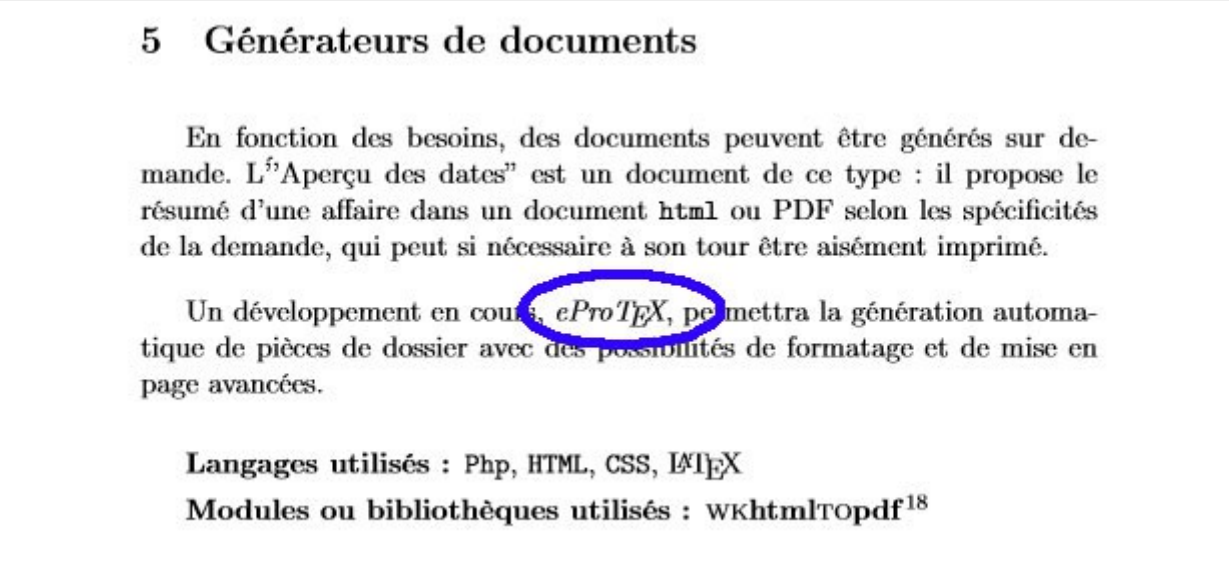


// How it evolved 3 : where it is now

- 5 years in development, 2 years in service
- Continuous development
  - Short iterations, fast reaction time, scrum-style
  - eunit
  - Gui's take a long time to make
- Interest in many forms, from many actors
- Electronic signature support with eID in Firefox, Chrome, Safari, Edge
  - in-house C# IE plug-in, Firefox crypto.signtext mess (pkcs#7)
  - Estonian eID software team does a great job
  - First open-source project of the CoS: xmldsig_js (XML-DSig)
- Stability : first year, no downtime.

// How it evolved 4 : what will/could come now

- Better document generation
- Cloud-like services
- Integration with 3rd parties
- GUI improvements
- Improved cluster operation



5 Générateurs de documents

En fonction des besoins, des documents peuvent être générés sur demande. L'"Aperçu des dates" est un document de ce type : il propose le résumé d'une affaire dans un document `html` ou PDF selon les spécificités de la demande, qui peut si nécessaire à son tour être aisément imprimé.

Un développement en cours, *eProTEX*, permettra la génération automatique de pièces de dossier avec des possibilités de formatage et de mise en page avancées.

Langages utilisés : Php, HTML, CSS, LaTeX

Modules ou bibliothèques utilisés : wkhtmltopdf[18]

// How it evolved 5 : common problems & cause

- User error
  - Software must mitigate risks and consequences (checks, rollbacks)
- Developer error
  - Deployment: Introduction of “test sheets”
  - Introduction of common software practices:
    - Change-lists on update (internal) + code diff
    - Proper ticketing (trac, redmine, paper...)
- Specification changes
- File formats: a little too conservative

// Bonus : the paperless paper office

checklist.odt(lecture seule) - LibreOffice Writer

Fichier Édition Affichage Insertion Format Tableau Outils Fenêtre Aide

EproAdmin Checklist

Push Report:

Date:

SVN Version:

OK / Not Ok for Prod

Notes:

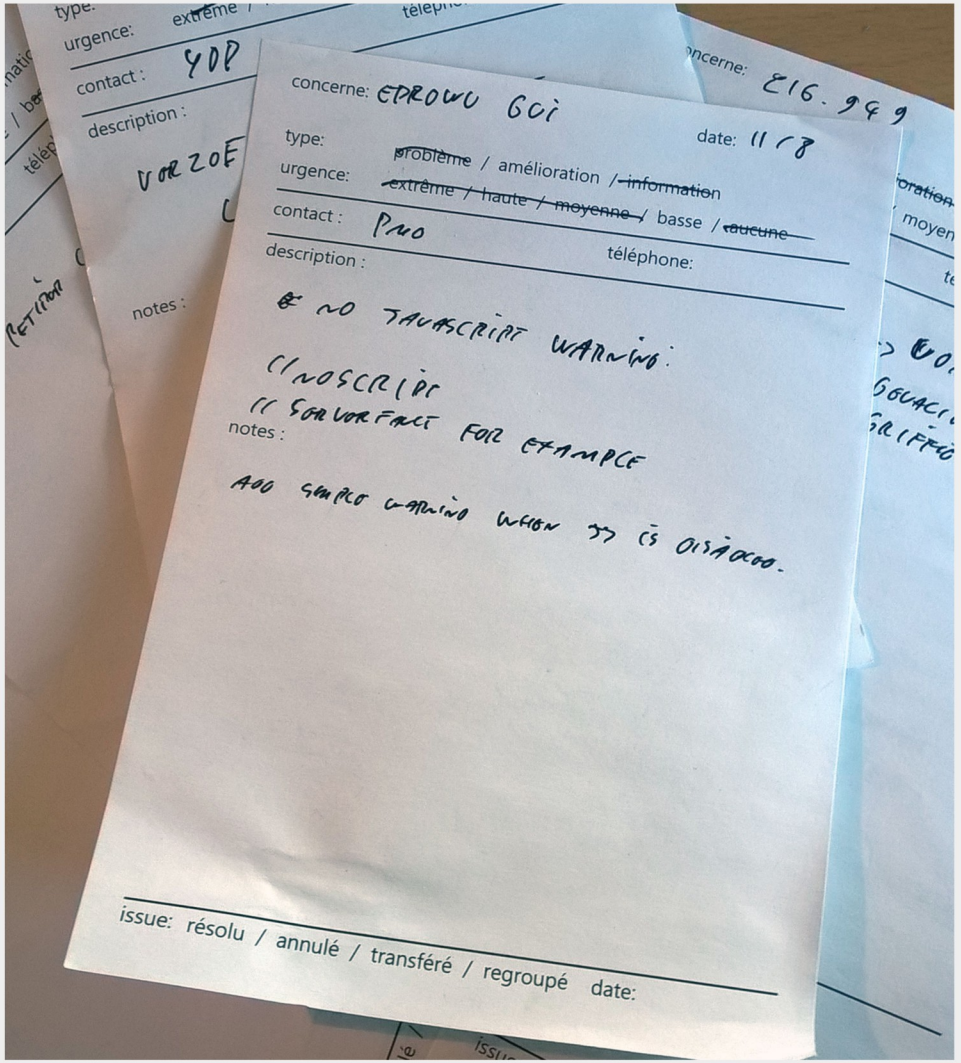
Push Notes:

Pushed on:

components that changed	Errors that happened

Rechercher | Tout rechercher | Respecter la casse | Naviguer par

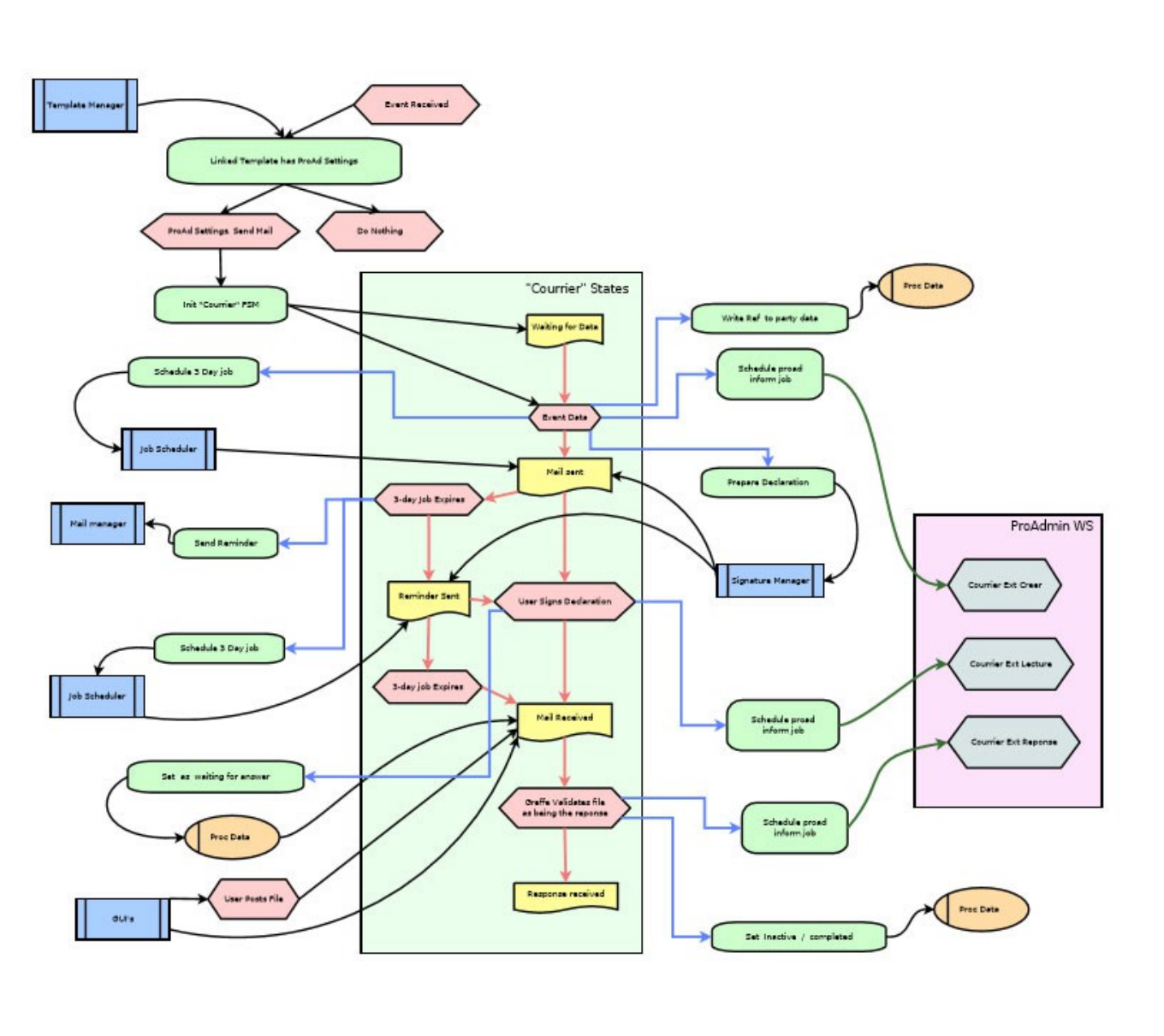
Page 4 / 5 | 223 mots, 1426 caractères | Style par défaut | Table8:B9 | 100 %



// How it evolved 6 : critical problems, experience gains

- Brain split
  - Network is partitioned, cluster members start their own life, consistency problem
  - Too few cluster members (at least 3, keep it uneven, quorum)
- Asymmetrical development teams
  - Internal web services, vendor-locked software stacks and the joy of SOAP RPC-through-HTTP
- Backups, or lack thereof
  - Mnesia makes it very easy
  - Erlang also (ets to file, …)
  - Need platform independent backups

// How it evolved 7 : “simple” internal web-service work-flow



Benefits of Erlang/OTP

// Benefits of using Erlang/OTP - 1

- Access layer
  - Easy data validation
  - Granular access rights
  - Synchronous & Asynchronous tasks
- OTP : “simple” & stable
  - Event handlers : gen_event, easily attach new handlers on event streams
  - simple_one_for_one : easy query handling parallelization
  - Supervision tree : can't crash, if done well
- Language
  - Atoms, pattern matching: great for refactoring

// Benefits of using Erlang/OTP - 2

- Language (continued)
  - Data structures (proplists, records, dicts, maps, gb_trees, binaries, …)
  - Type and function specification
  - Single variable assignments and simple in-and-out functions help build stable code, even in bad days (// TDD)
- Remote console : helps in developing, debugging, maintenance, support, ...
- Data storage : mnesia, ets, dets and storing native Erlang terms
- Integration with other languages
  - .Net, Prolog, Php, Python, Js
- Platform tools : eunit, dialyser, code coverage, vimerl, …

// Benefits of using Erlang/OTP - 3

- Quasi-DSL
  - Only needs some parse transforms with yecc or a simple lexer/tokenizer
  - Great for code re-use
  - Add or enhance functionality quickly and error free
  - Separates task definition & execution
  - Tasks can be dynamically altered or generated
- Code hot swapping. NOT !
  - Didn't use it yet, though I should
    - Full recompilations take time
    - Larger Mnesia DB's take time to load and sync

Example break

Streaming compressed file archive

(while still archiving/compressing)

In hindsight...

// In hindsight...

- Mnesia may or may not be the DB you're looking for
  - Great for intermediate/hot storage buffer, very fast reads, great for web application, transient data
  - Net-splits can occur, need tool for granular consistency check
- String handling, utf-8 support
  - Much better since v.16 and 17
  - Still no specific library : making your own tools promotes understanding of Unicode multi-byte encoding (which is good, and easy with Erlang's binary pattern matching)
- For long & serious projects, mitigate code rot risks: use 3rd party tools you know will still be maintained in 5 to 10 years
- Take time to set up the right environment for fast & easy iterative and incremental development (didn't do that, still no hot code swapping after 7 years)
- More open-minded look on interface, up-loadable file-types (*hard* problem)
- Prolog is stabe, but not very scalable

Prolog TCP server (rock stable)

interface.pl (~\Documents\eproadmin\tools\status_aligner) - GVIM

Fichier Edition Outils Syntaxe Tampons Fenêtre Aide

```
 7 % Socket Interface Implementation
 6 :- use_module(library(socket)).
 5 :- use_module(library(dialect/ifprolog)).
 4
 3 :- [text_parser].
 2
 1
 0 start_interface :-
 1     start_interface(25001).
 2 start_interface(Port) :-
 3     format("Starting Prolog server on port ~w~n",[Port]),
 4     tcp_socket(Socket),
 5     tcp_bind(Socket,Port),
 6     tcp_listen(Socket,5),
 7     tcp_open_socket(Socket, Read, _),
 8     dispatch(Read).
 9
10
11 dispatch(Read) :-
12     tcp_accept(Read, Socket, Peer),
13     thread_create(process_client(Socket,Peer), _,
14                   [ detached(true)
15                   ]),
16     dispatch(Read).
17
18 process_client(Socket,_Peer) :-
19     setup_call_cleanup(tcp_open_socket(Socket, In, Out),
20                        handle_service(In, Out),
21                        close_connection(In, Out)).
22
23 close_connection(In, Out) :-
24     close(In,  [force(true)]),
25     close(Out, [force(true)]).
26
27 handle_service(Stream, Out) :-
28     get_until(Stream, '$', RawText, '$'),
29     format('~n Received data ~s ~n',[RawText]),
30     atom_to_term(RawText,Query,_),
```

interface.pl 8,1 Haut

How it fitted for the gov't

How it fitted for the gov't

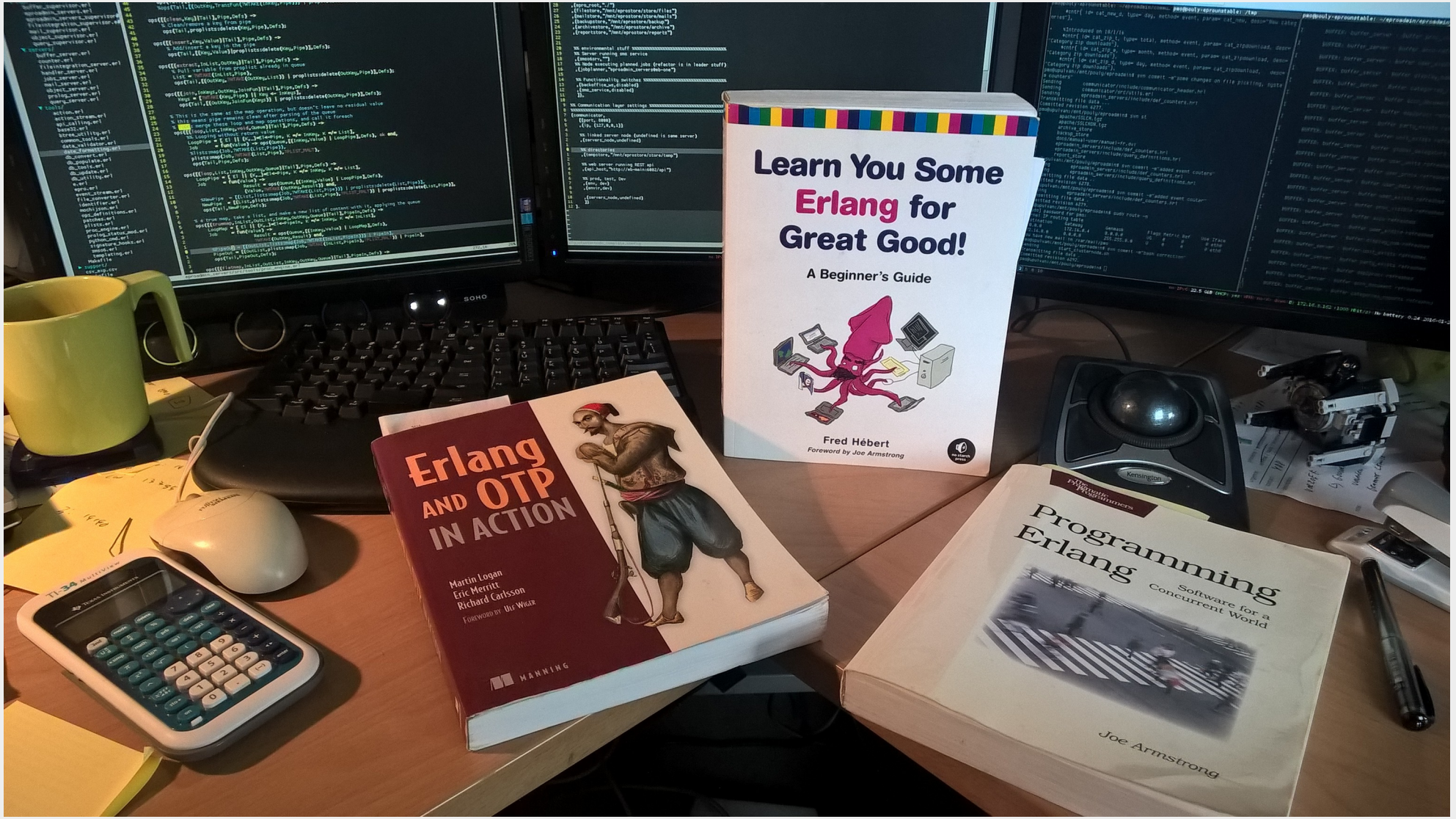
- It's open source : no charges, no vendor lock-in
- Powerful language even for small, 1-person teams
- Has been envisioned as a tool for things that (must) work, for achieving goals, for projects that must ship : adapts to new requirements, scales, provides useful libraries when needed (hmac, crypto, …), vibrant community, rich history
- Stable, reliable, refactorable, scalable
- OTP shines in server applications (which is basically everywhere these days)

After-word

// After-word 1

- The Erlang triad:
  - Programming Erlang – Joe Armstrong
  - Learn You Some Erlang for Great Good – Fred Hébert
  - Erlang and OTP In Action – M.Logan, E.Merritt, R.Carlsson
- Check the tools :
  - Rebar (deployment, releases, packaging, building, …)
  - Cowboy (modern & very fast http server)
  - Erlang.mk (Erlang build tool, dependencies,...)
- Start with a small project, take the time do to it well (OTP)

// After-word 2: some books



P. Montens – EFL Brussels Jan. 2016

That's all Folks !

Erlang/OTP In the wild: a governmental web application

Pieterjan Montens @ Erlang Factory Lite Brussels 2016

http://pieterjan.montens.net

pieterjan@montens.net

https://github.com/RvS-CdE

https://github.com/PieterjanMontens