Implementation Plan HEES.CC

Empowering EU citizens and EU contractors

How to achieve empowerment

- By delivering back office functionality (HEES ABC)

- By delivering Portal functionality

- By facilitating Collaborations
- By facilitating Change

- By enabling IT for Power users
- By enabling Administration for Power users
- By enabling Change management for Power users

See http://hees.cc/whitepaper.pdf

Implementation choices

- Browser based responsive Single Page web Applications
- Smart database design and every subject his own database!
- Smart development life cycle based on Github
- Synology Nas for a 'Hosting' like solution behind the firewall
- Any PHP, any PDO DBMS on any hosting solution
- SymetricDS for multimaster DBMS, File synchronisation
- VUE JS2 Framework plus DBarrest restfull services
- Google authentication

Universal HEES.CC ID

- The UHID is 8 characters, 64 bits long
- 32 bits subject-id, 32 bits object-id
- Within the application/dbms the local subject-id is zero
- During export, import the zero is replaced by the actual id
- The id is encoded with 0-9,a-z,A-Z (base 62) max 6 characters
- The subject (Citizin, Contractor, Community) does autoincrement
- The UHID never changes!!!!!
- Example: Subject-id=12, Objectd-id=61
  Local UHID (Id) is integer 61, encoded 'Z', Global UHID is 'c-Z'

Universal Concept Table

- All the tables with PK ID are merged together in one concept table
- The concept type defines which table/entity it concerns
- The table is defined with a view:

  create view agenda as select id, type, desciption, date,time from concept where type=”A”
- Now you can link any table to any table, hyper links
- All the tables share one global ID, so one autoincrement
- All the childs have their own local 'ID' called SEQ, PK: ID,SEQ

Concept Table shared attributes

- Id
- Type
- Status
- Title
- Description
- Target
- Quality
- Owner
- Last-Timestamp
- Last-Author

- Title and Description contains the local language and the global english translation

  Afspraak met CEO | Appointment with CEO
  Within description it is separated by <hr>

Concept Table Status

- Activities: Todo, Done
- Authoring: Concept, Approved, Published, Archived
- Leads: Prospect, Contacted, Proposal, Rejected, Accepted
- Invoice: Send,First notice,Second notice,Collection Agency,Paid
- Project: Consencus ,Chances, Constraints, Coherence, Concepts, Classes, Components, Configurations, Collaborations,Cockpit Indicators, Changes, Conclusions

Concept Table Quality

NULL	not applicable
'?'	needs QA
'X'	garbage, please delete
'*'	unsatisfactory
'**'	satisfactory
'***'	good
'****'	very good
'*****'	excellent

Concept Table Target

The target determines where a concept is placed in the portal'.

The portal is consists out of 4 flex-boxes: header, navigation, content, footer. The header always contains a dropdown menu, a title and a reference to the google authenticated user.

• 'content'	will be rendered in place
• 'navigation'	will be rendered in a separate page
• 'footer'	are pop-ups, example: contact – question – FQA -

Concept Table Owner

Ownership is very important, it I used to delegate responsibilities within a community / project.

Anything can be owned by the group or the members.
So you can delegate the ownership of an action or a deliverable to the whole group or one individual memeber

Tag Management

- #tags managed by the subject owner
- During import and export not matching tags are placed behind a second separator in description
- $tags are budget tags, applied for balance and profit and loss
- @tags are target tags also for administration for journals like 'trip amsterdam', 250 euro, 27 vat, $travel @creditcard

  @cash, @bank, @loan, @mortgage and @cross

- The tag-table structure: Id, Tag, Id-Tag-Type
  The tag table is local and cannot be exported
  The Tag-Type table is centrally managed at HEES.CC

- The tag child table is: Id-concept, Id-tag

Staging areas Development Life Cycle

- Development, everybody develops under his own branch in his own development environment
- Quality assurance, the changes are merged with the production version in the master branche but in the qa staging area. The test databases got the qa- prefix.
  The code is deployed under qa.website
- Production. The current production code is staged to backup. The views are prefixed with backup-
  All the old functionality is still available from backup.website. The local htacces prefixes all the restfull requests with backup. The current tables will be extended with the new columns based on the qa-tables. Code tables will receive their new content. The new production code is staged to production. The QA area is cleared
- The master branche remains available for hotfixes

Development Life Cycle Principles

- One is only allowed to extend functionality
- Mind business rules: when changing the age rule from required age is 18 to required age is 21
  This is an organisational problem, not an IT problem.
  The rule 21 can only apply for new accounts or all the invalid accounts will be removed. It is up to the business to decide!!!
- The enduser has no notion of specific versions.
  His own version, the qa version, the production version and the backup version but not a specific version
- The UHID never changes
- Tag management is a local responsibility
- Max 25 members in a project, community
- Every local community needs at least 3 local servers
- Blackbox and whitebox components are cascaded.
  So one extends functionality with local and group components.
  In case of conflicts the newest wins.