NAVINUM®

Navinum technical & functional at a glance

http://navinum.net/

Architecture

- One or more Navinum servers
  - Local at the exhibition
  - Remote in the internet
  - LAMP + Symfony

- Any number of navinum clients
  - Mobile (tablets, smartphones…)
  - Static (PCs, Arduinos, proprietary devices…)
  - Simple readers (RFID or any technology)
  - Connected websites (any oauth2 enabled CMS)

Navinum server

- Handles (via the **navinum** package):
  - A database
  - A RESTful API
  - An admin interface

- Optionally handles (via the **navinum-websocket-sso** package) :
  - SSO (with an oauth2 server)
  - Real-time notifications and clients live interactions (via websockets)

- Can synchronize with other servers.
  - Currently, simple bi-directional synchronisation with **Unisson**,
  - EAI/ESB integration is in the roadmap.

Navinum database

- Exhibitions
  - Can be shared / replicated between museums
- Visit courses and experience units
  - Courses alternatives (full or partial)
- Visitors (anonymous or not) and groups
  - Profiles and parameters (l10n, a11y…)
  - Gamification framework (XP, medals…)
- Full visitor log
  - individual course, experience unit scores…
- Devices fleet (RFID, tablets…)

Notifications & triggers

- A generic kind of rule,
- Written in LUA
- Listens to internal database events
  - profile change, score update, user XP update…
- Sends notifications or updates DB
  - User notifications, medals, new XP, new scores…

Usage

- Implementations:
  - http://navinum.net/category/references/
- Tutorials:
  - (under writing)
- Installation:
  - https://github.com/CapSciences/navinum/wiki/