RAIN - Sprint 9 Specification document

Goal

At the end of Sprint 9 RAIN team will deliver the HIP registry integration as well as OAUTH research.

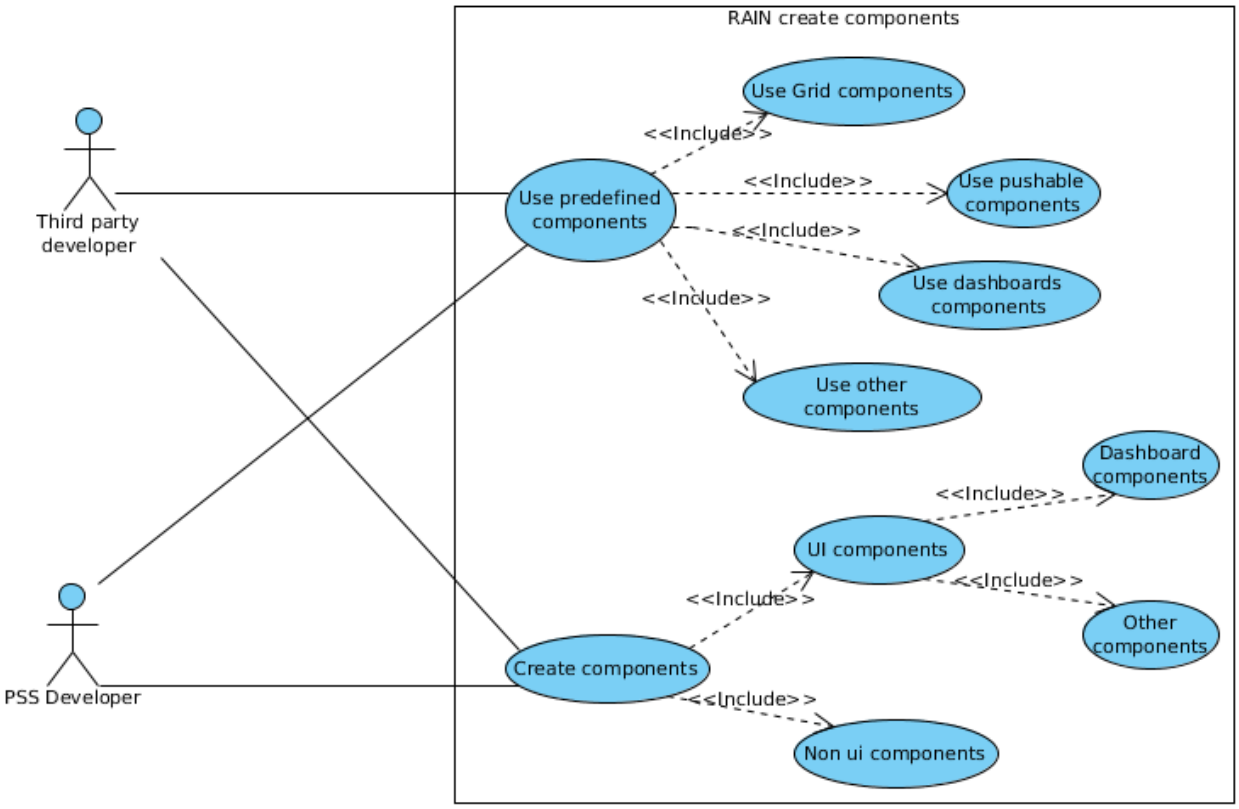
Below you can find a summary of what team will deliver:

- I18N transport layer + client side localization
- HIP Registry - RAIN locate operation
- OAUTH research
- Minor improvements: customer logout, session restore

Visual Elements

All developers using RAIN will create components that will finally be assembled to an application. Most of the time developers will simply create a new RAIN component using composition. To create a component probably developers will use existing components and I will add some new "from scratch" behavior. RAIN platform sdk provided for 1&1 should include plenty of components so that PSS / Third party developers focus only on product. Most common categories of components (as seen in other frameworks as well) are:

- Grid components - Components that allow developers to present data in list / tabular fashion.
- Button components - Different types of buttons: Drop down button, Rounded button, etc.
- Dashboard components - Components that are usually found on home pages / main screen of an application.
- Other components - Components that can't be placed in one of the above mentioned categories.



User stories

1. **RAIN datagrid component.** - **RAIN-10** - **(13 USP)**

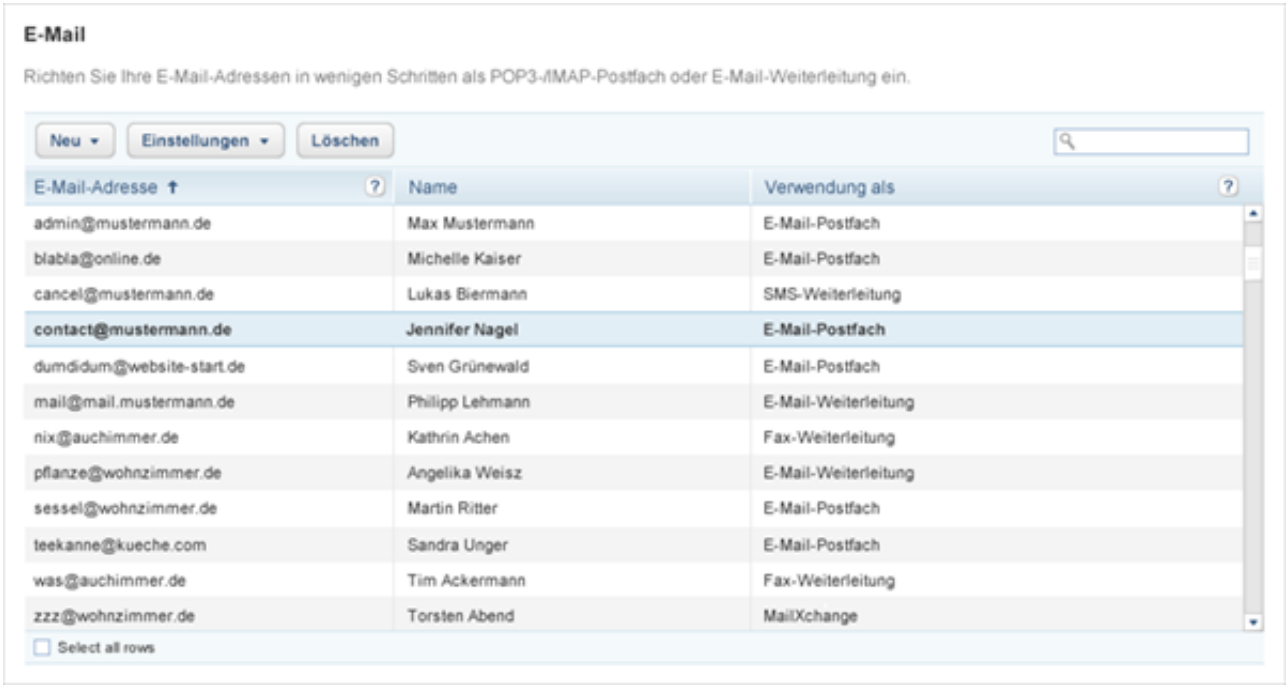
As a Developer / Third party developer I want to use a filterable and sortable table so that I present large set of data in a friendly manner. This must be a reusable table component with Webdesk look & feel (as UCD is not yet available). The table must provide filtering, sorting and searching features. In addition no performance penalties are accepted (the table must react fast for large data sets > 300 records). We must not write the component from scratch. We will choose an existing and tested component. [RAIN-105]

More information can be found on:
http://styleguide.schlund.de/konzeption/guidelines/wtg1/designpattern_workplace/interaction_tables.htm

You can find the attached information within the issue attachments.

Acceptance

- A sample application must be created to show how the component can be used. The headers must be localized in en_US and de_DE format. Example must emphasize support for UTF-8 special characters.
- Documentation must be written for PSS / Third party developers (how to / examples).
- The sample application must be presented from dev machine.



RAIN security

RAIN must provide a pluggable authorization system. The authorization process must be compliant with the RBAC standards so that it is easier to understand from developer perspective and protect us from making mistakes (by not following a well known security standard).

User stories

1. **RAIN OAUTH2 client research - RAIN-202 - (21 USP)**

   As PSS / Third party developer I want RAIN security to be integrated with OAUTH2 so that authentication / authorization is done transparently for me. It means that RAIN must have an OAUTH2 client that can be used to fulfill OAUTH2 1&1 flows. Here you need to collaborate with CloudIA team to understand all the OAUTH2 flows that are / will be implemented. Check https://github.com/ciaranj/node-oauth to see an example of node oauth consumer project.

   Acceptance

   - A feature proposal for OAUTH2 RAIN client will be presented.
   - The login flow with OAUTH2 enabled must be presented (doc).
   - The data store flow with OAUTH2 enabled must be presented (doc).

HIP Registry

HIP platform must provide a single service to PSS / Third party developers that allows them to register / unregister / locate all deployed **bundles** from the running platform. A bundle from HIP registry can contain:

- CloudIA services
- RAIN frontend components
- Monitoring endpoints

The bundle is a logical unit of deployment versioned automatically by HIP Registry. Detailed information can be found on: http://vm2384.development.lan:8080/upcoming/proposals/Features-proposal-hip-registry.html

Based on the HIP Registry the platform will also provide a HIP Registry Frontend secured with OAUTH. More information about the frontend can be found on: http://vm2384.development.lan:8080/upcoming/proposals/Feature-proposal-hip-registry-frontend.html

User stories

1. **HIP registry meta model - RAIN-290 - (13 USP)**

   As PSS / Third party developer I want to know what is the bundle meta model so that I can register RAIN components easily.

   Acceptance

   - A document describing the bundle meta model will be presented
   - CloudIA / RAIN will agree on the bundle meta model.
   - Examples of how register / unregister requests will look like.

RAIN I18N

RAIN framework must support localization of components. In RAIN localization can be static or dynamic. Static localization mean that a tag will be added to the view and RAIN renderer will automatically replace that tag with the correct localization information. A second use case occur when developer wants to load localization information dynamically (on fly). For instance I as a developer want to do an AJAX request and based on the answer I want to display "Ok" / "Error 404" based on the response.

Moreover, for 1&1 deployment RAIN must be compliant with Translation API process described in great detail in: http://wiki.intranet.1and1.com/bin/view/Home/MainTechnicalConceptLocalization#5_2_Content_Localization_Process

This must be very well understood and tested in order to ensure consistency is obtained. It is clear from the document mentioned above that RAIN must use PO Files or property files (or both) for achieving localization of text.

User stories

1. **RAIN localization service - RAIN-110 - (8 USP)**

   As RAIN I will provide a service to return modules text resources in json format so that PSS / Third party developers can localize client side texts. The service will retrieve the po catalog content for the specified language and will give message ids and message text under a json representation. If the locale specified is not found the default platform locale will be retrieved but a warning message must be logged. See also [RAIN-100]

   Acceptance

   - An integration test must be presented.
   - Unit tests for success / fallback / failure scenarios.
   - Documentation must be written for PSS / Third party developers so that they know how to obtain texts for client side localization.
   - The integration test must be presented for dev environment.

2. **RAIN client side text translations. - RAIN-117 - (13 USP)**

   As PSS / Third party developer I want to access the localized text resources from client side controller so that I can do client side text translation.

   Acceptance

   - The code must be commited on rainjs.
   - A sample application must be commited on rainjs so that client side text translation is proved.
   - The login form from previous sprints must be translated to en_US and de_DE (use google translate). The login form will be shown in both locales.
   - Unit tests must be written for success / failure scenarios.

- Documentation must be written for PSS / Third party developers to prove how client side text translation works.
- The sample application must be presented from dev environment.