Borderline**:** Design Specifications

REVISION 139 | 21-04-2017



[www.etriks.org](http://www.etriks.org)

This project is supported by eTRIKS, a collaboration between 17 different partners, driven to develop platform and services for data staging, exploration and use in translational research.

Document Approval

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Date | Signature |
| Prepared by: | F. Guitton P. Agapow  J. Grizet | 04-02-2017 |  |
| Checked by: |  |  |  |
| Approved by: |  |  |  |

Distribution List

|  |  |  |  |
| --- | --- | --- | --- |
| Party | Name | Function | For Info / Approval |
|  |  |  |  |

Change History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Author | Description |
|  |  |  |  |

Table of Contents

[1. Introduction 5](#_Toc480533809)

[1.1. Purpose 5](#_Toc480533810)

[1.2. Scope 5](#_Toc480533811)

[1.3. Reference Documents 5](#_Toc480533812)

[1.4. Abbreviations and acronyms 6](#_Toc480533813)

[2. Background 7](#_Toc480533814)

[2.1. Reviews 7](#_Toc480533815)

[2.1.1. Review of tranSMART 1.2 UI 7](#_Toc480533816)

[2.1.2. Review of related R1: Glowing Bear UI 7](#_Toc480533817)

[2.1.3. Review of related R3: Evaluations of user experience 7](#_Toc480533818)

[2.1.4. Review of I2B2 7](#_Toc480533819)

[2.1.5. Review of cBioPortal.org 8](#_Toc480533820)

[2.1.6. Review of Synapse 8](#_Toc480533821)

[2.1.7. Review of Galaxy 8](#_Toc480533822)

[2.1.8. Review of SmartR 8](#_Toc480533823)

[2.2. Key observations 9](#_Toc480533824)

[2.2.1. Successful features 9](#_Toc480533825)

[2.2.2. Missing features 9](#_Toc480533826)

[3. Tasks and scenario 11](#_Toc480533827)

[3.1. Tasks 11](#_Toc480533828)

[3.1.1. Browse and filter data 11](#_Toc480533829)

[3.1.2. View, edit and create analysis 11](#_Toc480533830)

[3.1.3. Submit, consult and chain analysis 11](#_Toc480533831)

[3.2. Scenarios 12](#_Toc480533832)

[3.2.1. Analysis 12](#_Toc480533833)

[3.2.2. Bookmarks 13](#_Toc480533834)

[3.2.3. Workflow 13](#_Toc480533835)

[3.2.4. Plugins 14](#_Toc480533836)

[4. Architecture design 15](#_Toc480533837)

[4.1. Components overview 15](#_Toc480533838)

[4.1.1. Borderline – Graphical User Interface 15](#_Toc480533839)

[4.1.2. Borderline – Server 15](#_Toc480533840)

[4.1.3. Borderline – Middleware 16](#_Toc480533841)

[4.1.4. Borderline – Cloud Connector Interface 16](#_Toc480533842)

[5. Borderline – Graphical User Interface 17](#_Toc480533843)

[4.1 Sample page name 17](#_Toc480533844)

[6. Borderline – Server 18](#_Toc480533845)

[6.1. Technologies 18](#_Toc480533846)

[6.1.1. Node.js & Express 18](#_Toc480533847)

[6.1.2. MongoDB 18](#_Toc480533848)

[6.2. Architecture overview 19](#_Toc480533849)

[6.3. Extensions 20](#_Toc480533850)

# Introduction

## Purpose

This document defines the eTRIKS tranSMART system new graphical interface design – codenamed “Borderline” - in the scope of the work carried by Work Package 2 (Infrastructure Platform Design and Development) implementing scalable, secure and reliable eTRIKS Knowledge Management platform by extending and enhancing tranSMART core architecture from Description of Work revision 0.7.

## Scope

This document is the Graphical User Interface Design Document for the eTRIKS tranSMART system. The purpose of this document is to present the design approach for the graphical interface for the web codenamed “Borderline”. This include both the front-end and the back-end middleware necessary to obtain a workable infrastructure against the tranSMART data warehouse The primary intended audience of this document are system designers and system builders. The document intents to provide the members of the eTRIKS consortium a unified view of the technical details of the system design to be followed during the development of the respective application. The document may need to be updated later to incorporate possible changes during development.

## Reference Documents

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Reference | Title | Version |
| R1 | The Hyve | tranSMART Glowing Bear UI Design Document | 24-11-2016 |
| R2 | The Hyve | tranSMART Pro Project Design | V1.2 |
| R3 | Mansoor Saqi | GUI DP Summary of evaluations of user experience  and functionality offered in tranSMART 1.1 | - |

## Abbreviations and acronyms

A list of the principal abbreviations and acronyms used in the document is provided here for a better understanding of this document.

|  |  |
| --- | --- |
| Abbreviation | Definition |
| TM | tranSMART |
| DSI | Data Science Institute |
| CSS | Cascading Style Sheets |
| HTML | HyperText Markup Language |
| JSON | JavaScrip Object Notation |
| eAE | eTRIKS Analytical Engine |
| eHS | eTRIKS Harmonisation Service |
| URI | Uniform Resource Identifier |
| REST | REpresentational State Transfer |
| CPU | Central Processing Unit |

# Background

This section summarizes the reviews of the preceding user interfaces used with or close to tranSMART. It also attempts to highlight successful & missing features to capitalize on when designing the new borderline graphical user interface.

## Reviews

### Review of tranSMART 1.2 UI

The current interface of tranSMART is articulated around a single menu bar or ribbon containing an obscure selection of inconsistently correlated items; some containing unusable option, some useless option. Only two options of that menu are proven to be used: Browse and Analysis. The core design principle of tranSMART is the tree-like structure it holds in both those tabs. The iconic design allows for dragging and dropping of tree items into boxes leading on to the creation of patients sets of which correlated data can be manipulated. Across tabs though, the behaviour of interface components is inconsistent featuring in multiple places cascade of undocumented tabs and submenus leading to semi-functional pages.

### Review of related R1: Glowing Bear UI

Document R1 presents a complete rewrite of the interface using similar web technologies but streamlining the implementation and adapting it to the most recent available features described in document R2 and implemented by the same company.

While it is a seducing approach to improving the interface in the context of our requirements we will have to go away from the technological stack employed here. From a static REST based polling type interaction to a more dynamic bidirectional socket based message bus one.

### Review of related R3: Evaluations of user experience

Document R3 describes the tests and evaluations carried out with tranSMART 1.1. The two main areas discussed are the user interface and the functionality. It focuses on the assessment of the user interface, very like the current interface at time of the writing and categorizes problems with the Search and Dataset Explorer interfaces. The document proposes a set of diagrams and possible schematics for evolution of the interface. It also goes over issues with the clarity of the interface and the clutter that can be found in some area.

Not providing technical insights and building upon an already outdated and cluttered interface we will be going away from the proposed design organization but retain the overall improvement concepts featured in the document.

### Review of I2B2

I2B2 is the ancestor project of tranSMART it works in a similar fashion and in fact tranSMART on its own is structure internally in very similar ways. It works around the same idiom of patient sets and concepts but has very limited analytical capabilities which corners it as a cohort discovery tool more than data exploration. Contrary to tranSMART is has a provenance and query history feature that is present front and center. The interface contains components the visibility of which can be toggled on or off.

### Review of cBioPortal.org

cBioPortal has confusing tool categorization and is sparse. We note a very complete set of features including 3D rendering capacity for chemistry and a network graphing tool very complete.

### Review of Synapse

Synapse is a software that is more of a wiki-like platform focused on data sharing and governance/compliance of licensing. It has a built-in detailed granular permission system with approval chains.

### Review of Galaxy

Galaxy takes the approach of being a library of tools chainable into workflow that can be run offline with limited scaling abilities. It has a very interesting concept of customizable menus.

### Review of SmartR

SmartR takes the data visualization interactivity to an interesting level by allowing the manipulation of the input data filtering directly from the visuals themselves. This permits an interesting feedback loop which holds great advantage in terms of the user ability to explore the data more intuitively.

## Key observations

After reviewing existing user interfaces and collecting user feedback, we present in this section the features that should be embedded in the core of tranSMART systems.

These can be features from existing systems or requested features. In both cases users value these features in their interfaces.

### Successful features

* + - 1. Non-blocking interface navigation

Working with large datasets often leads to long computation times: several hours and even days for the most extreme cases. This is problematic when the interface force the users to wait for their result, blocking all progression on their tasks. To solve this problem, Galaxy has a job logic where users can submit and queue computations, while allowing fluid navigation in the interface.

* + - 1. Subset based analysis

In many use cases, it is of interest for the users to refine a given dataset in their interface as the first step of their workflow. This feature allows users to work directly within the interface instead of downloading the whole data set, filtering by hand and re-uploading a subset. In the context of tranSMART & I2B2, this process is the cohort selection step.

* + - 1. Auditing (from Galaxy)

TODO 4 Flow

### Missing features

#### Control running code

After selecting a subset from the well-known data tree, multiple processing tools can be used and none of them displays the code running. We believe that a tranSMART GUI, as a research environment, must allow users to consult the source code because it is imperative for them to control every step of the data analysis.

#### Cache ‘computation-expensive’ results

After computing a visualization or a transformation on a data-set, none of the tranSMART interfaces stores the output. Forcing to run the computation every time the user returns on his own work. This is problematic for large data-sets where users must wait for a long time before continuing their work.

#### Sharing & Collaboration

Operating with complex data sets in large organizations leads to multiple users working on the same pipeline at various stages. Therefore, we believe that introducing collaboration in a tranSMART could be beneficial for the user experience. For example, by sharing entire pipelines on the same platform, users should quickly achieve their collaborative tasks.

#### End to end encryption

TODO 4 Flow

#### Large scale data visualization

TODO 4 Flow

#### Distributed computation

In the past years, datasets have become too large for analysis to be run in sequential. As we have seen the raise of big data, we have seen new distributed software tools been used to reduce computation time with this ever-increasing data sizes.

TranSMART is not an exception, especially when dealing with genomics data and the community should be looking forward to speedup analysis using distributed software.

#### Operational scalability

Horizontal scaling is a widespread practice in industry. This technique helps preventing bottlenecks and data loss for each component of the system horizontally scaled. TranSMART based systems could be improved by implementing it on the databases, data storage and servers. We believe scalability is most important at the data access level because most of the bottlenecks are located there.

# Tasks and scenario

This section describes the main tasks and scenarios that must be achievable from this new Borderline - GUI. Even though these require all the components of the borderline project to work together, we will just focus on non-technical descriptions from the user perspective.

## Tasks

### Browse and filter data

As the first step in any analysis, a user must be able to navigate through a given data set previously made available on a connected tranSMART instance.

The task here then consists in creating subsets by choosing one or more concepts from the data set, selecting all the data entries belonging to this concept. Optionally, users can refine the subsets using a filter specific to their concept type in question.

After creating subsets of the data using filters, the tasks require creating new subsets composed from the existing subsets. This is done by choosing a composition operation and the subsets. For example, one may create a subset from the intersection/inner join of two or more subsets created previously.

For clinical data sets, by the end of this task, the user has created a cohort from concepts filters and concept composition operation.

We intend on improving the user experience on this task by separating the filtering and the composing into two separate steps. We also believe users should be able to repeat this task to create as many queries as needed where it is currently limited to two.

### View, edit and create analysis

This second task comes after the data selection. It is here question of running code where the data is provided as input.

It starts by choosing which queries will be used in this analysis. Then, the process to create analysis involves choosing what to run from an existing analysis list or by creating a new one.

Finally, the code can be viewed and/or edited as well as other input parameters the current analysis may require.

### Submit, consult and chain analysis

After configuring or creating an analysis, it must be run. The user will just have to submit from the interface to create a job in our system.

Then users want to use the results of this job for visualization, consulting and exporting.

Next step allows for chaining into another analysis by choosing the current results for input during the 3.1.2 task. Then by submitting again we create a task loop in between tasks 3.1.2 and 3.1.3 called analysis chaining.

## Scenarios

We describe here the main collections of subsequent navigation actions that users will execute in the new Borderline - GUI interface. These scenarios allow users to achieve, pause, and resume their tasks.

### Analysis

#### Create analysis

A – From Home page, go to Workflow tab

B – Create one query or multiple queries

C – Choose which analysis to run

D – Choose which data to input

E – View or edit analysis code

F – Configure input parameters

G – Submit analysis

* + - 1. Consult analysis progress

A – From Home page, go to Analysis tab

B – Find the targeted analysis in the page

C – Get the status from the displayed information

#### Consult analysis results

A – From Home page, go to Analysis tab

B – Find analysis in the page

C – Advance to this analysis result page

D – Consult results

#### Chain finished analysis

A – From any page, go to Analysis tab

B – Find analysis in the page

C – Advance to analysis result page

D – Advance to next step

E – Choose which analysis to run

F – Choose result data as input

G – View or edit analysis code

H – Configure input parameters

I – Submit analysis

### Bookmarks

#### Bookmark analysis results

A – From analysis result page

B – Tag this result a bookmarked

#### View bookmarked query

A – From Home page, go to Bookmarks

B – Consult bookmarked queries.

C – Choose bookmarked query

D – Navigate to query creation page.

#### View bookmarked analysis result

A – Go to Homepage / Bookmarks

B – Consult Bookmarked analysis results

C – Choose bookmarked results

D – View chosen analysis results

### Workflow

#### Consult current workflow overview

A – From any page under the Workflow tab

B – View workflow graph

#### Navigate to workflow point

A – From any page under the Workflow tab

B – View workflow graph

C – Select workflow point

D – View workflow point details

E – Navigate to workflow point

### Plugins

#### Browse plugin store

A – From any, go to Plugin store

B – View Plugins

C – Expand some plugin details

#### Enable analysis template plugin

A – From any, go to Plugin store

B – Search plugins under analysis template

C – Select plugin

D – Enable plugin

#### Disable a user interface extension plugin

A – From any, go to Plugin store

B – Search plugins under user interface

C – Select plugin

D – Disable plugin

# Architecture

In this section, we want to introduce the structure of the Borderline project. We envisage the interface to sit as a user facing component within a larger solution constituted of a graphical user interface, an orchestration server, an on-demand analysis platform and a middleware abstraction layer for data management.

## Components overview

Figure 1: Component diagram

### Borderline – Graphical User Interface

The borderline graphical user interface is accessible with any modern web browser and is the only component of the solution users can interact with. It communicates directly with the Borderline – Server component via a RESTful API to manage users queries and analysis within authenticated sessions.

### Borderline – Server

On top of providing all the business logic for the Borderline – GUI component, the server orchestrates the storage of work sessions, enables data access, data storage and handles computation management.

It keeps track of the queries towards tranSMART instances, as query unique identifier provided by the Borderline - Middleware. The server can then retrieve the data for any step of an analysis process. By exchanging these identifiers again.

Computation tasks are also stored using a unique identifier, provided by the Borderline - Cloud Connector Interface component. Submission, status query and results query are then automatically performed when users advance in their workflows.

### Borderline – Middleware

To enable cross-study analysis over multiple tranSMART instances, the middleware component merges authentication, endpoint and constraints into a unique query. It keeps tracks of the queries and caches them so that the Borderline – Server and Borderline – Cloud Connector Interface components can manipulate data from multiple endpoints seamlessly.

### Borderline – Cloud Connector Interface

When installing the Borderline – Server component, at least one Borderline – Cloud Connector must be provided. This component being an interface, this means that it can be implemented by any project.

This interface provides access to a cloud computing infrastructure where the Borderline – Server can submit jobs as requested by the users through the Borderline – GUI. It also provides access to a data cache where the Borderline -Server can fetch the analysis results for display or further analysis.

In the context of the eTRIKS project we envisage the analysis party to be the eAE. The eAE would progressively implement the Borderline – Could Connector Interface to be a pluggable component in a Borderline system installation.

## Repositories

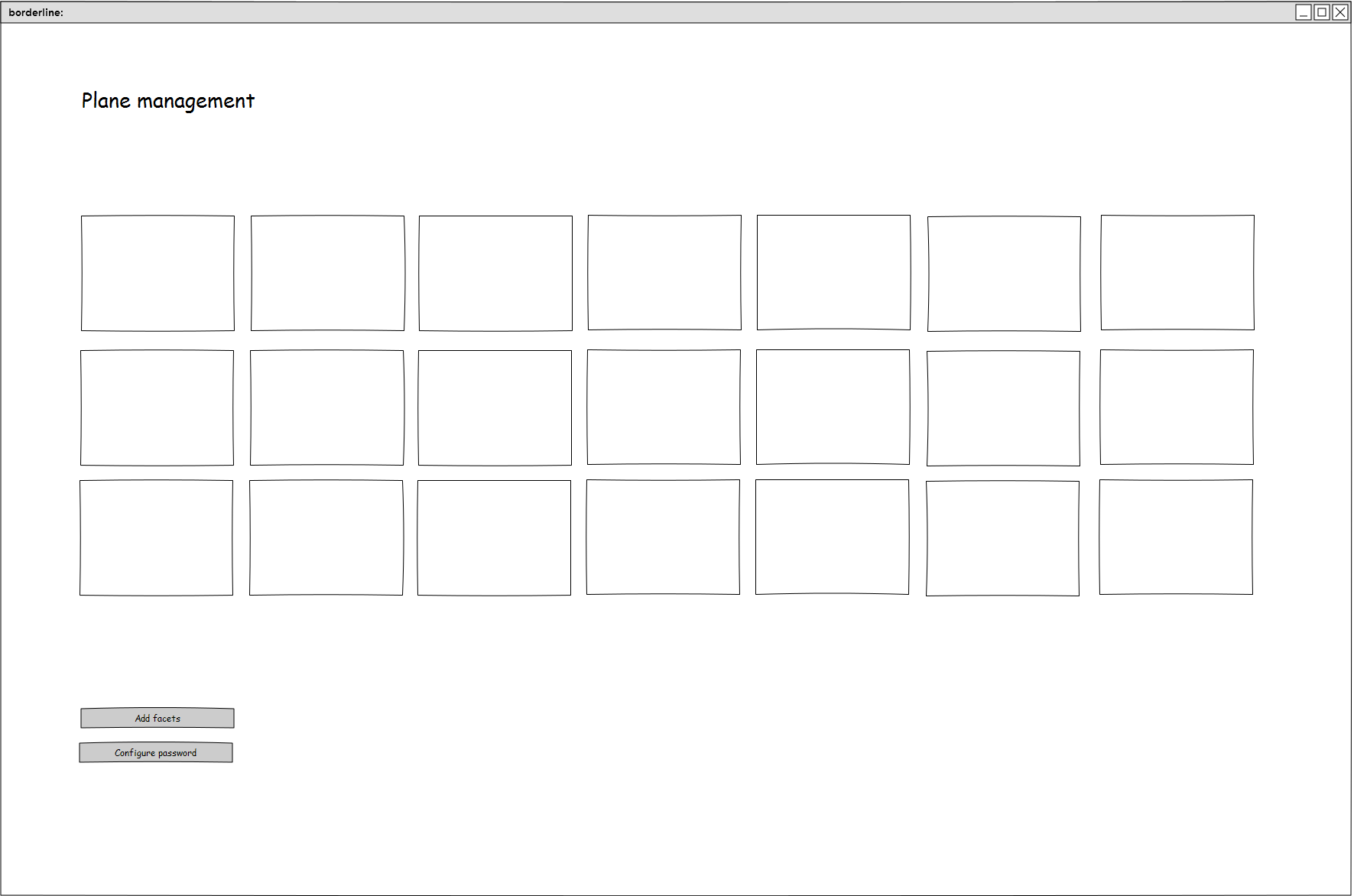
Borderline components have their own git repository privately hosted within the Data Science Institute group on Github: **borderline-server** and **borderline-ui**.

Two additional repositories exist: **borderline-documentation** that is used for documentation and file sharing. Whereas **borderline-create-extension** is a helper npm package for extensions development bootstrap.

# Borderline – Graphical User Interface

## Sample page name

This section describes the “Sample page name” page. Describe here.



|  |  |
| --- | --- |
| Purpose | User can drag and drop screen icons to organise their plane |
| Navigation & User Interaction | This page allows the user to configure the layout of their plane and add a optional password. |

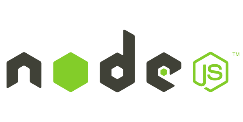
# Borderline – Server

As the principal component of Borderline, the Borderline - Server provides a RESTful API for the client applications. It is responsible for authentication, sessions, and storage logic of the work realised on connected clients.

## Technologies

### Node.js & Express

Borderline – Server software stack is composed of Node.js and Express. This is a choice we made because this stack has proven to be free, efficient, easy to work with, well documented and relatively simple to scale.

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.[[1]](#footnote-1)

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.[[2]](#footnote-2)

### Image result for mongodbMongoDB

For data storage, we choose to rely on MongoDB. It is a free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemas and is free and open-source.[[3]](#footnote-3) It is also a well-known database that has proven to be efficient, well documented and simple.

In addition, the RESTful API of the Borderline – Server data format is JSON as this is standard practice in web applications. Using MongoDB then allows us to work with JSON objects, without serialization effort in between the Node.js framework, the database and the network protocol.

## Architecture overview

Borderline – Server class hierarchy is what you would expect from Node.js & Express project. Being a Node.js package itself, it exposes a top-level class “BorderlineServer”, constructed by passing configuration to its constructor and returning an Express router.

Internally, the server initializes sessions and then binds RESTful endpoints organized in separate controllers. Where a controller in the Borderline – Server oversees the HTTP request handling of a group of related endpoints.

Functional implementation is located in separate logic classes, called by the controllers. As this logic require access to the database, they have access to the collections within their scope. In most of the case we are looking at one controller calling upon one business logic class which access one collection, but more complex features like the Workflows depends on multiple logic classes and collections.

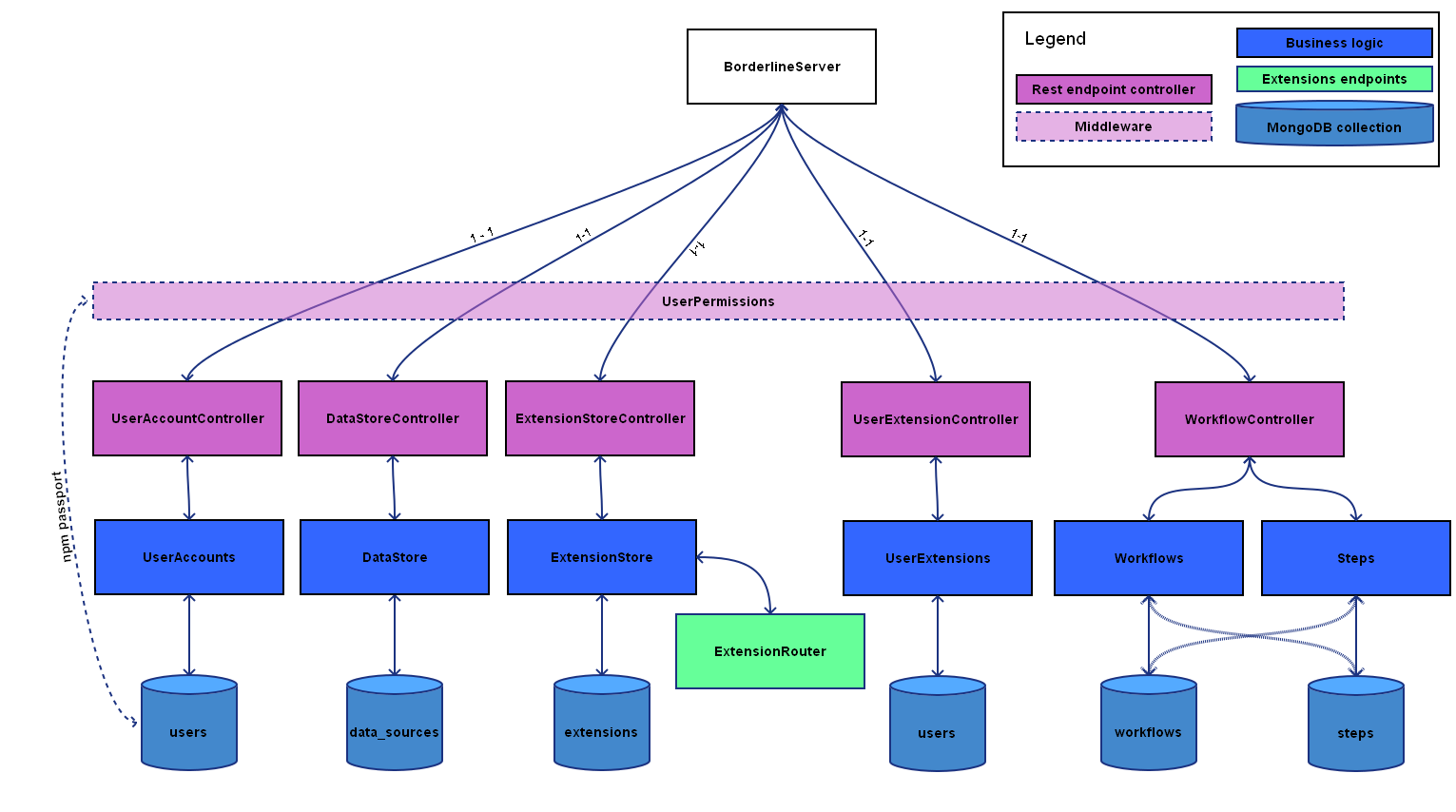


Figure 2: Borderline - Server components diagram

## Extensions

Borderline – Server supports extensions. Each extension has access to its own router, filesystem, http, https, path and URL modules form the Node.js & Express framework.

An extension is a zip package that must contain a manifest file **plugin.json**. Borderline – Server expects the manifest file to have the following format:

{  
 "name": "example",  
 "version": "0.0.1",  
 "description": "Example extension",  
 "author": "example author",  
 "build": "00000000000",  
 "id": "00000000000",  
 "server.js": "server.example.js",  
 "client.js": "client.example.js"  
}

Where the file pointed by the field “server.js” can be either a file generated using Webpack or a raw JavaScript file. If provided this file is giving access to the framework with the **borderline** keyword. Then the *attach* method of this same file is called, with a new express router as parameter. The router will then be mounted at ***/extension/{id}/***

When the field “client.js” is provided, the server hosts this file at ***/extension/{id}/client***

## RESTful API

The specifications and documentation for the RESTful API interface the server implements can be found in the documentation repository. The file is ***/Technical/API\_borderline-server.yaml*** and is written using the Swagger 2.0 format (http://swagger.io/). It is possible to easily generate a HTML version at http://editor.swagger.io/.

# Borderline – Middleware

Borderline – Middleware is a server exposing a small interface to manage data access across multiple storage solutions and instances. It is mainly responsible for standardizing the data format and keeping track of the queries.

## Technologies

We wish Borderline – Middleware to be a lightweight component providing horizontal scalability. As demonstrated in 6.1 for the Borderline – Server, we believe Node.js and MongoDB should be the preferred technologies to answer these requirements. In addition, this should allow developers to use standards re-use code and collaborate across both projects.

## Data format

The expected data format is JSON, as the middleware interface depends on the latest TranSMART 17.1 which uses JSON format.

When extending the support to other projects such as I2B2 or eHS, the middleware must ensure that their data format is converted to a compatible JSON format. This must allow the Borderline – Middleware to be used with any supported underlying data source seamlessly.

## RESTful API

The specifications and documentation for the RESTful API interface the middleware implements can be found in the documentation repository. The file is ***/Technical/API\_middleware.yaml*** and is written using the Swagger 2.0 format (http://swagger.io/). It is possible to easily generate a HTML version at http://editor.swagger.io/.

# Borderline – Cloud Connector Interface

Borderline - Cloud Connector Interface is the only component that isn’t a standalone software. It is instead a generic interface to be implemented by external project in order to achieve compatibility with Borderline.

This interface must allow Borderline to submit and manage job submissions when properly implemented. Borderline – Cloud Connector Interface also includes blob management for managing the jobs related data.

## Job description

In Borderline, a Job is composed by the following elements:

* A main script: This script is the one edited directly by the user, where the actual analysis happens.
* A wrapper script: This is the script intended to be executed on cloud infrastructure. It is in charge of formatting and feeding the inputs to the main script.
* Data blobs: A job may require input data, or it may need to output data. This list of blobs gives the cloud infrastructure access to storage for these needs.
* Status: A job lifecycle goes through different steps. Namely these steps are Pending, Scheduled, Running, Cancelled or Done.

## RESTful API

The specifications and documentation for this RESTful API interface can be found in the documentation repository. The file is ***/Technical/API\_eae.yaml*** and is written using the Swagger 2.0 format (http://swagger.io/). It is possible to easily generate a HTML version at http://editor.swagger.io/.

1. https://nodejs.org/ [↑](#footnote-ref-1)
2. https://expressjs.com/ [↑](#footnote-ref-2)
3. https://en.wikipedia.org/wiki/MongoDB [↑](#footnote-ref-3)