# Earth System Grid Federation monitoring infrastructure

**P. Nassisi, S. Fiore, G. Aloisio**

Euro Mediterranean Center on Climate Change (CMCC)

**Rios-Lovell Estate Winery**

*Livermore, CA Dec 9-11 2014*

is-enes

INFRASTRUCTURE FOR THE EUROPEAN NETWORK FOR EARTH SYSTEM MODELLING

cmcc

Centro Euro-Mediterraneo sui Cambiamenti Climatici

# Outline

- *Introduction*
- *Goals*
- *Requirements analysis*
- *Use cases and scenarios*
- *Catalog extension*
- *Architecture in the large*
- *ESGF Dashboard*
- *ESGF Desktop: design, architecture and main views*
- *REST APIs*
- *Conclusions and future work*

# Introduction


This work is funded by the EU FP7 IS-ENES2 project under the task "*Developing software infrastructure for data archive services*"

Specifically, it's related to the design and the implementation of the Federated Archive System Monitoring (FASM).

CMCC activities:

i. FASM design
ii. Milestone: MS111 "Monitoring System and Dashboard Design" (March 2014)
iii. Update/Extension of the existing framework to adhere to the new requirements and design

# Goals

The main goals of this system is to provide a ***distributed and scalable monitoring framework*** responsible for:

- capturing usage metrics, system status and aggregated information at the single site level, at the ENES archive level and at the global ESGF level
- providing the user with a dashboard including views, aggregated statistics and monitoring information.

The monitoring system faces this important challenge through two main components:

**ESGF Dashboard**
**(FASM-N)**

**ESGF Desktop**
**(FASM-D)**

# Requirements analysis

The analysis of the requirements allowed to identify a set of system features and functionalities.

The system has to be able to provide an **easy and transparent access** to data and statistics gathered by a set of *sensors* related to:

- **real time** or **aggregated** system status monitoring information per node, like CPU, memory, etc.
- federated system status monitoring information
- *data download statistics* (both per-node, institution-based and at federation level) provided according to different views as **time, models, variables, datasets, experiments, etc.**
- **data client usage statistics** (both per-node, institution-based and at federation level) grouped by country or continent and aggregated over time
- federation-level monitoring information regarding nodes status, registered users and deployment information
- bandwidth information among nodes
- …

In addition, the system has to be able to:

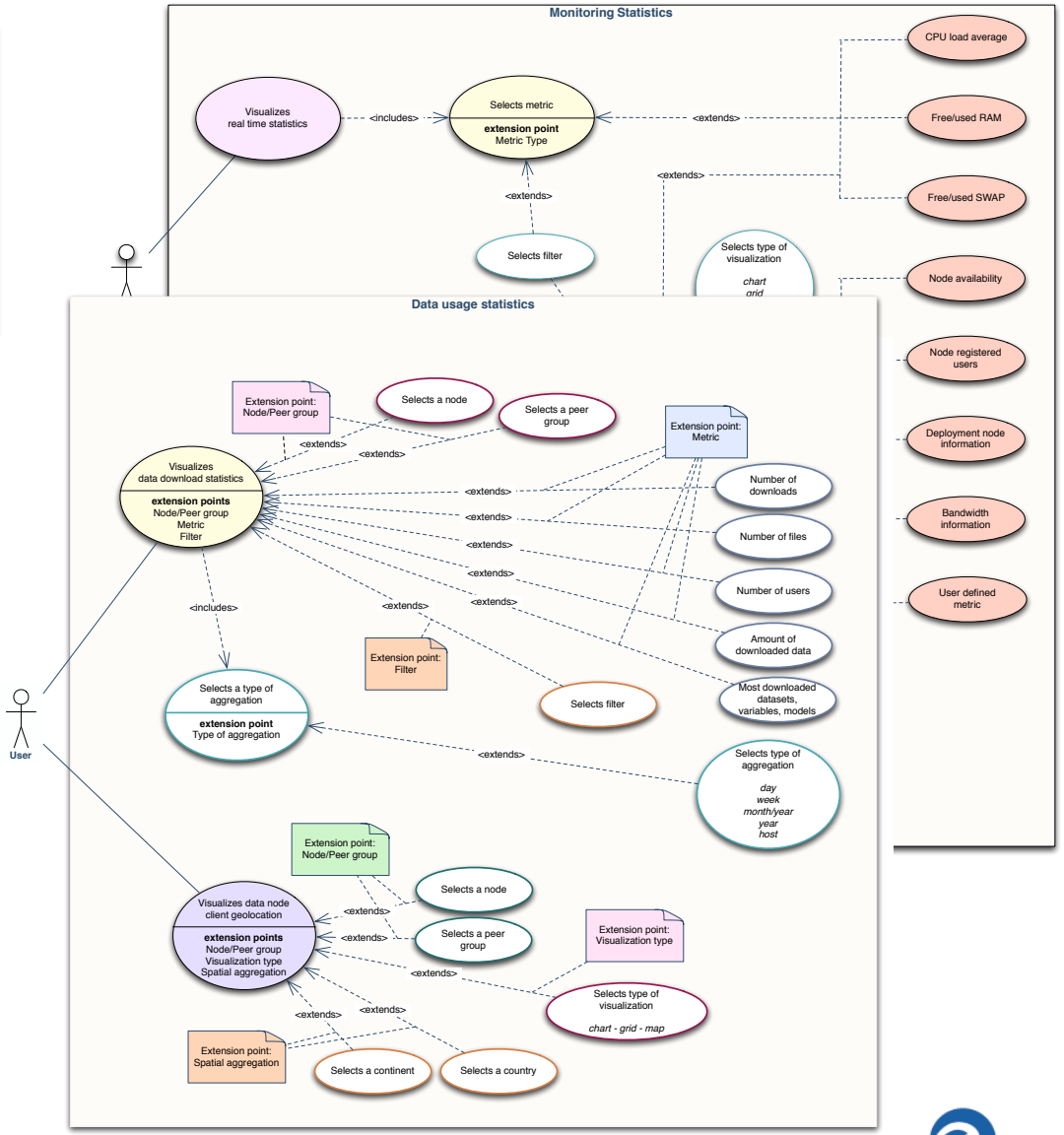
- offer a configurable way to instantiate and configure **new sensors** based on a set of pre-defined sensor classes
- provide an **ECA-based (Event Condition Action) system** to associate events, conditions and actions regarding the monitored metrics
- support a ***single-node running*** mode for separate portal instances like the climate impact portal

# Use cases and scenarios

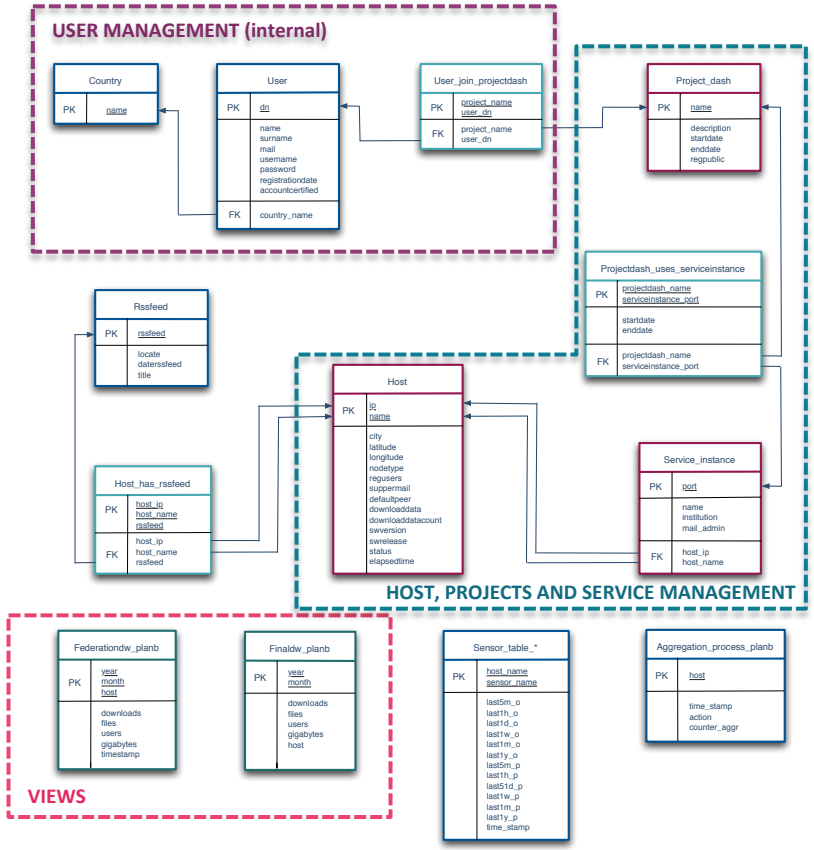
The design phase of the monitoring framework was carried out according to the principles of the *software engineering* with the support of UML diagrams.

Main usage scenarios of the system:

- User authentication
- Monitoring statistics
- Management console
- Data usage statistics
- Dissemination/help



# ESGCET catalog extensions




Each node in the federation already exploits a system catalog called ***esgcet***.

In order to persistently manage in the back-end information related to the ESGF dashboard (peer groups, hosts, services, aggregated views, etc.) a new namespace has been defined:

the ***esgf_dashboard*** namespace.

Such a design choice has several benefits, since:

i. it isolates the new tables from the other ones
ii. it provides a good coupling with the existing system by exploiting the same RDBMS engine
iii. it does not introduce any additional dependency on other/new software.

# ESGCET catalog extensions

| DATA DICTIONARY (Entities) | | | |
|---|---|---|---|
| **ENTITY NAME** | **DESCRIPTION** | **ATTRIBUTES** | **IDENTIFIER** |
| **User** | The registered user of the FASM system. They will be mapped onto OpenID registered users | dn,<br>name,<br>surname,<br>mail,<br>username,<br>password,<br>registrationdate,<br>accountcertified | dn |
| **Country** | Contains the list of the countries. | name | name |
| **Project_dash** | Istance of the peer group/project shared by some node of the federation. | name,<br>description,<br>startdate,<br>enddate,<br>regpublic | name |
| **Service Instance** | Monitored service istance of a federation node. | port,<br>name,<br>institution,<br>mail_admin | port |
| **Host** | Node participating in the federation. | ip,<br>name,<br>city,<br>latitude,<br>longitude,<br>nodetype,<br>regusers,<br>suppemail,<br>defaultpeer,<br>downloaddata,<br>downloaddatacount,<br>swversion,<br>swrelease,<br>status,<br>elapsedtime | ip |
| **RSSfeed** | Istance of RSS feed, useful for sharing feeds. | rssfeed,<br>locate,<br>daterssfeed,<br>title | rssfeed |
| **Federationdw_planb** | Istance of the data usage information at a federation level. | year,<br>month,<br>downloads,<br>files,<br>users,<br>gigabytes,<br>host,<br>timestamp | year,<br>month,<br>host |
| **Finaldw_planb** | Istance of the data usage information at a node level. | year,<br>month,<br>downloads,<br>files,<br>users,<br>gigabytes,<br>host | year,<br>month |
| **Aggregation_process_planb** | Support entity for the management of the aggregation process. | host,<br>timestamp,<br>action,<br>counter_aggr | host |
| **Sensor_table_*** | Table for aggregated statistics related to a | host_name,<br>sensor_name, | host_name,<br>sensor_name |

Each node in the federation already exploits a system catalog called ***esgcet***.

In order to persistently manage in the back-end information related to the ESGF dashboard (peer groups, hosts, services, aggregated views, etc.) a new namespace has been defined:

the ***esgf_dashboard*** namespace.

Such a design choice has several benefits, since:

i. it isolates the new tables from the other ones
ii. it provides a good coupling with the existing system by exploiting the same RDBMS engine
iii. it does not introduce any additional dependency on other/new software.

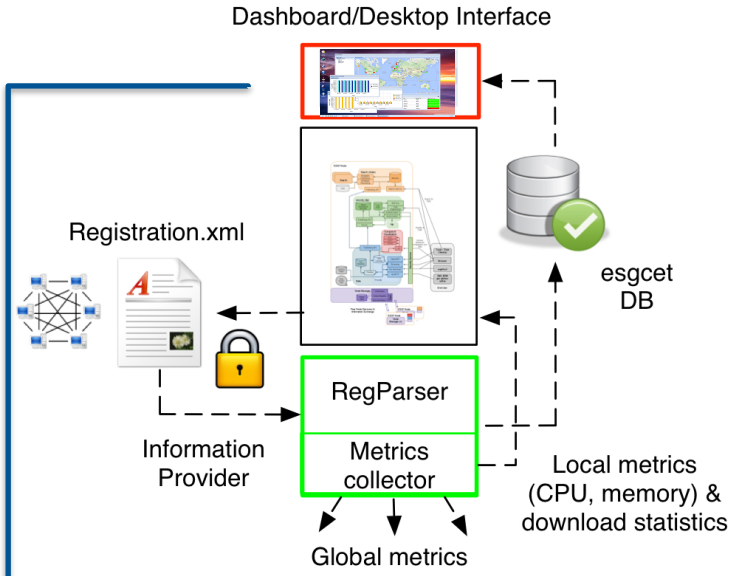
# Architecture in the large

The main modules of the monitoring system are, for the ***back-end***:

- the Information Provider
  It strongly interacts with the node manager and it's responsible for retrieving all the metrics and storing them in the *esgcet* catalog and binary files (for long term statistics).
- the dashboard catalog
  A system database which stores all the information about hosts, peer-groups, services, users, availability, deployment, etc.
- the sensors
  Global and local metric sensors to retrieve and manage information about *network topology*, *node type*, *registered users*, *downloaded data*, *system metrics*.

For the ***front-end***:

- a web-based environment GUI: the ESGF Desktop
  modular web application relying on a strong adoption of Web 2.0 concepts and providing several views at different granularity levels
- a set of configuration files

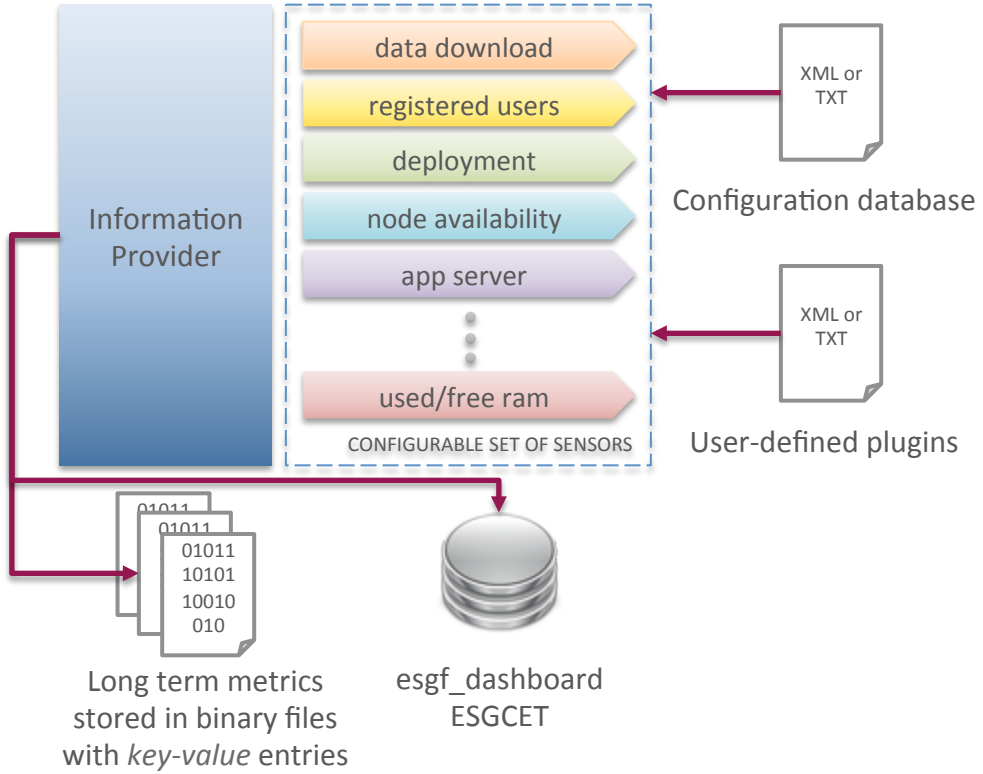



# ESGF Dashboard

The ESGF Dashboard represents the **core of the system**. By design, it synchronously runs sensors and stores information to get updated snapshots related to the status of the federation.

***ESGF Dashboard***

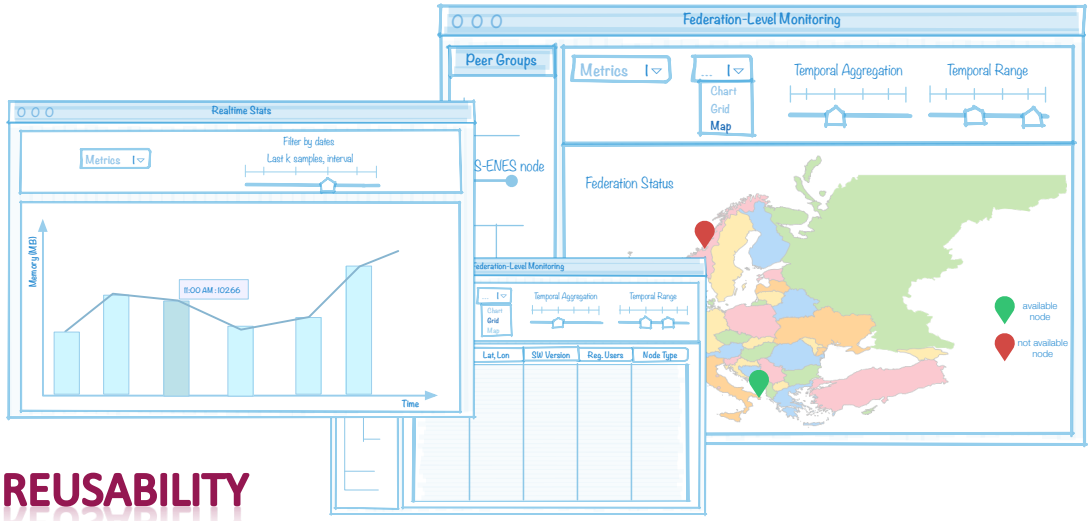
***Architecture in the small***



# ESGF Desktop

The main goal of the ESGF Desktop is to provide a comprehensive view at single site level, at ENES archive and at the global ESGF level, in terms of monitoring statistics, data usage and clients distribution.

It exploits the **MVC design pattern** and it relies on a strong adoption and implementation of **Web 2.0 concepts** such as mash-up, Google maps and permalinks.




REUSABILITY

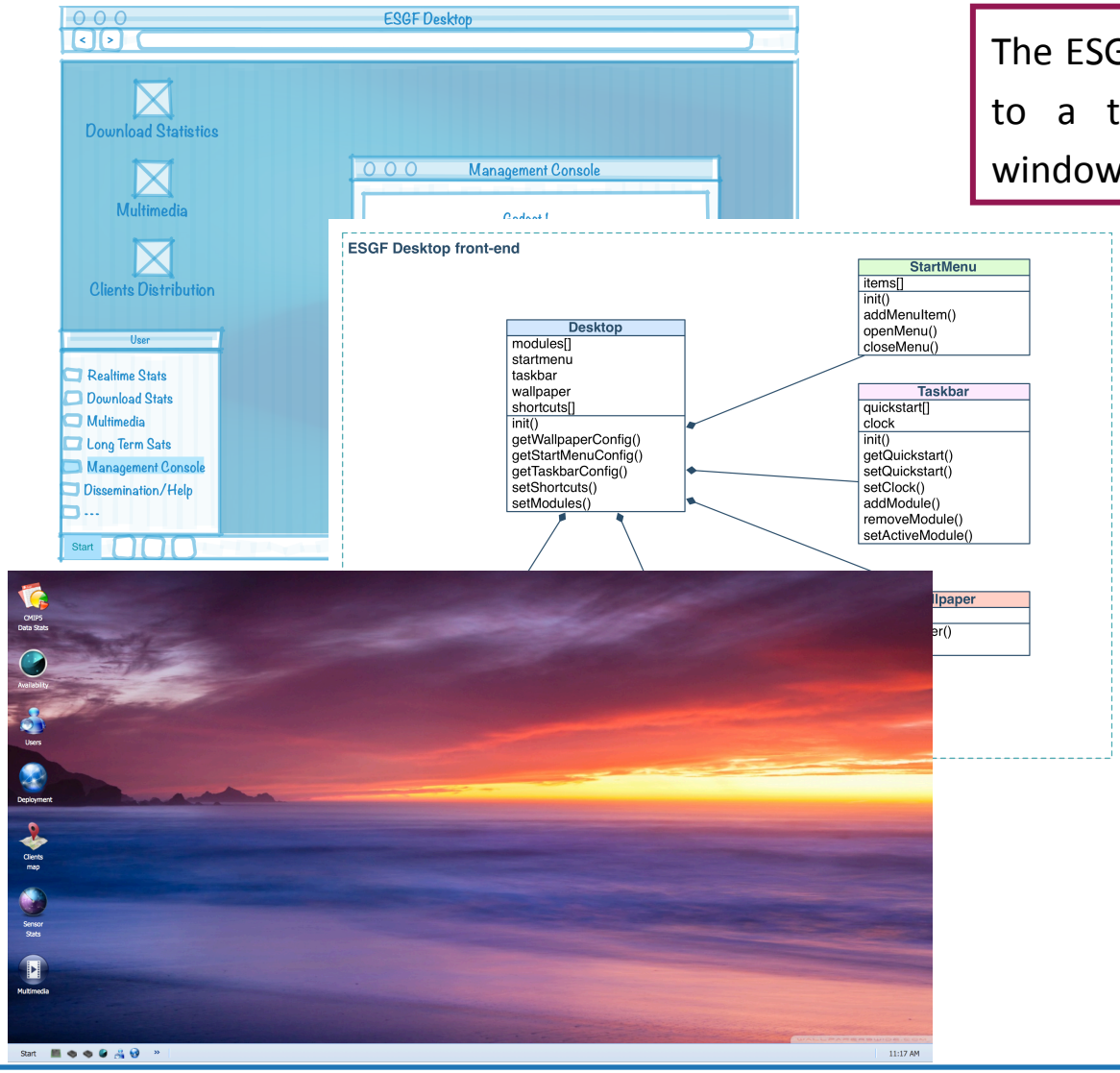
CONFIGURABILITY

EXTENSIBILITY

SCALABILITY

*This component is strongly coupled and integrated into the ESGF architecture.*

# The “desktop metaphor”




The ESGF Desktop is an environment similar to a traditional “desktop” with several windows, but using a *web-based approach*.

***Features:***

- Shortcuts to provide a quick and direct link to some key views;
- Multiple menu/sub-menu available;
- “window-based” functionalities and self-consistent views;
- Multiple windows available on the screen at the same time;
- a taskbar keeping track of open forms;
- system-level capabilities to configure the environment.

# ESGF Desktop: architecture

## Metadata Information

- database *esgcet* (*esgf_dashboard* namespace)
- other files containing gadgets configurations

## ESGF Desktop back-end

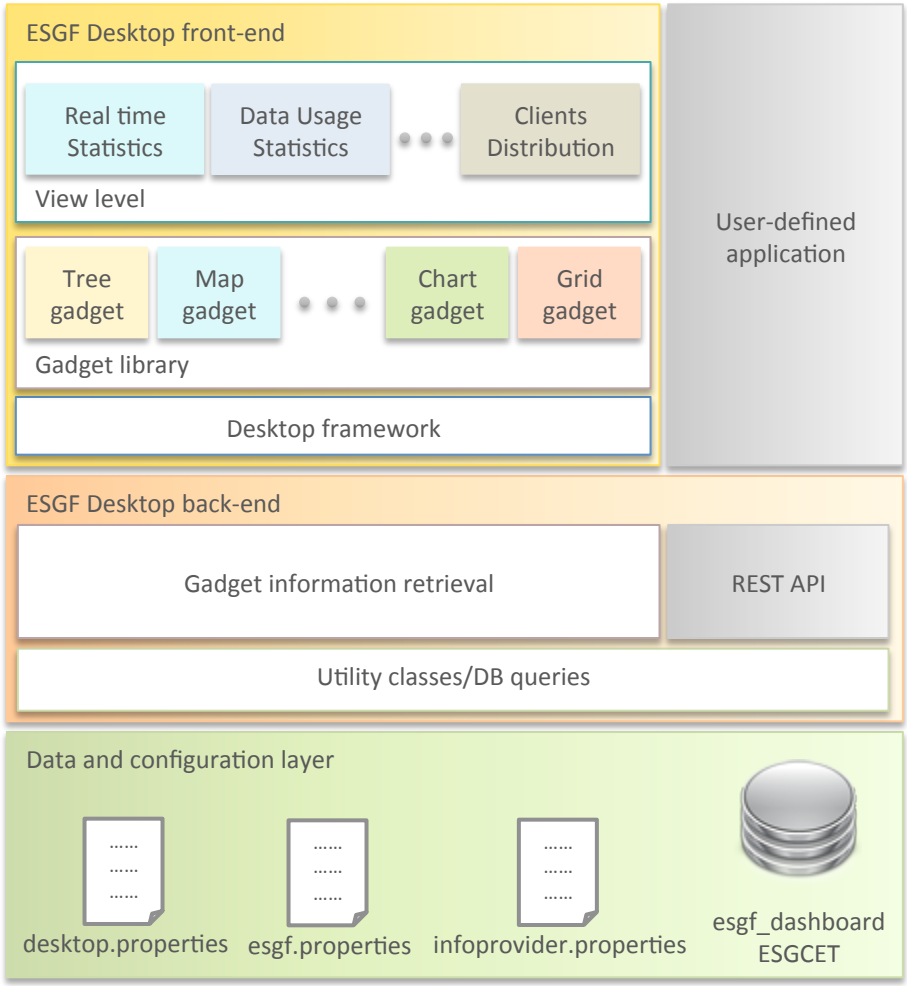
- Utility classes for the connection to the database
- business logic classes
- REST APIs

## ESGF Desktop front-end

- Desktop framework, which deals with the management of the overall environment
- Gadgets library, consisting of a number of base module combined in the system views
- Views, consisting in more complex and completed modules

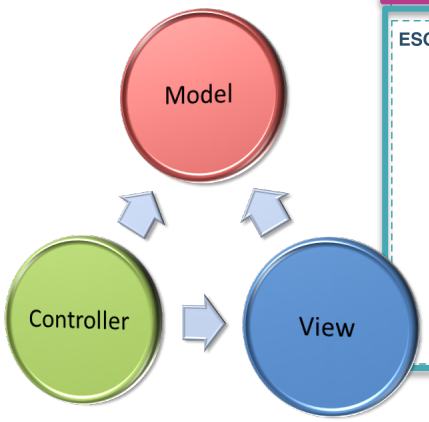
## User-defined application

The user can define new applications exploiting the REST service provided by the Desktop back-end.



# ESGF Desktop: design

## MVC approach




VIEW

ESGF Desktop front-end

VIEW

| Long Term Statistics Module |
|---|
| items[]: [PeergroupsTree, FilterPanel, OutputPanel] |
| createDesktopWindow(items) |

| Peergroups Tree |
|---|
| Peergroups Store |
| initComponent() |

| Output Panel |
|---|
| Type of visualization |
| StatisticInfo Store |
| initComponent() |

| Filter Panel |
|---|
| Filters[] |
| initComponent() |

CONTROLLER

| Controller |
|---|
| views[]: [Long Term Statistics Module, Peergroups Tree, Output Panel, Filter Panel] |
| init() |
| controlEvent(query submission) |

| Behaviours |
|---|
| loadPeergroupsTree() |
| loadStatisticInfo(Output Panel, selected peergroup/node, filters[]) |

MODEL

| Peergroups Store |
|---|
| Peergroups Model |
| Proxy |
| load() |

| StatisticInfo Store |
|---|
| StatisticInfo Model |
| Proxy |
| load() |

| Peergroups Model |
|---|
| Fields[] |

| StatisticInfo Model |
|---|
| Fields[] |

ESGF Desktop back-end

| Authentication Interceptor |
|---|
| UserSession |
| getUserSession() |
| setUserSession() |

| LoadPeergroupsTreeAction |
|---|
| PeergroupsTree |
| loadTree() |

| Peergroups Model |
|---|
| Fields[] |

| GetStatisticsAction |
|---|
| StatisticInfo[] |
| loadStatistics() |

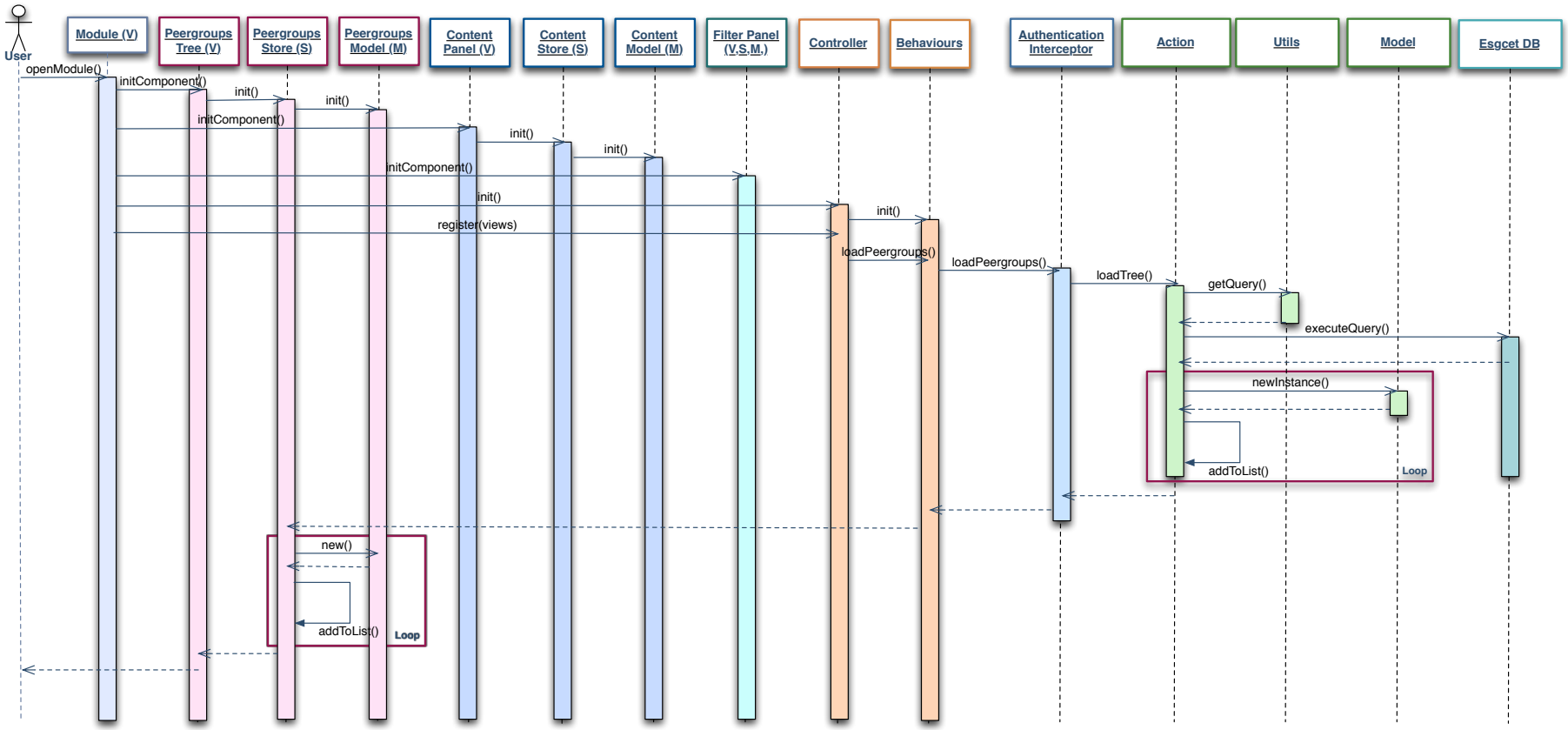
| StatisticInfo Model |
|---|
| Fields[] |

| Utils |
|---|
| PeergroupsQuery |
| StatisticsQuery |

BUSINESS LOGIC AND MODEL

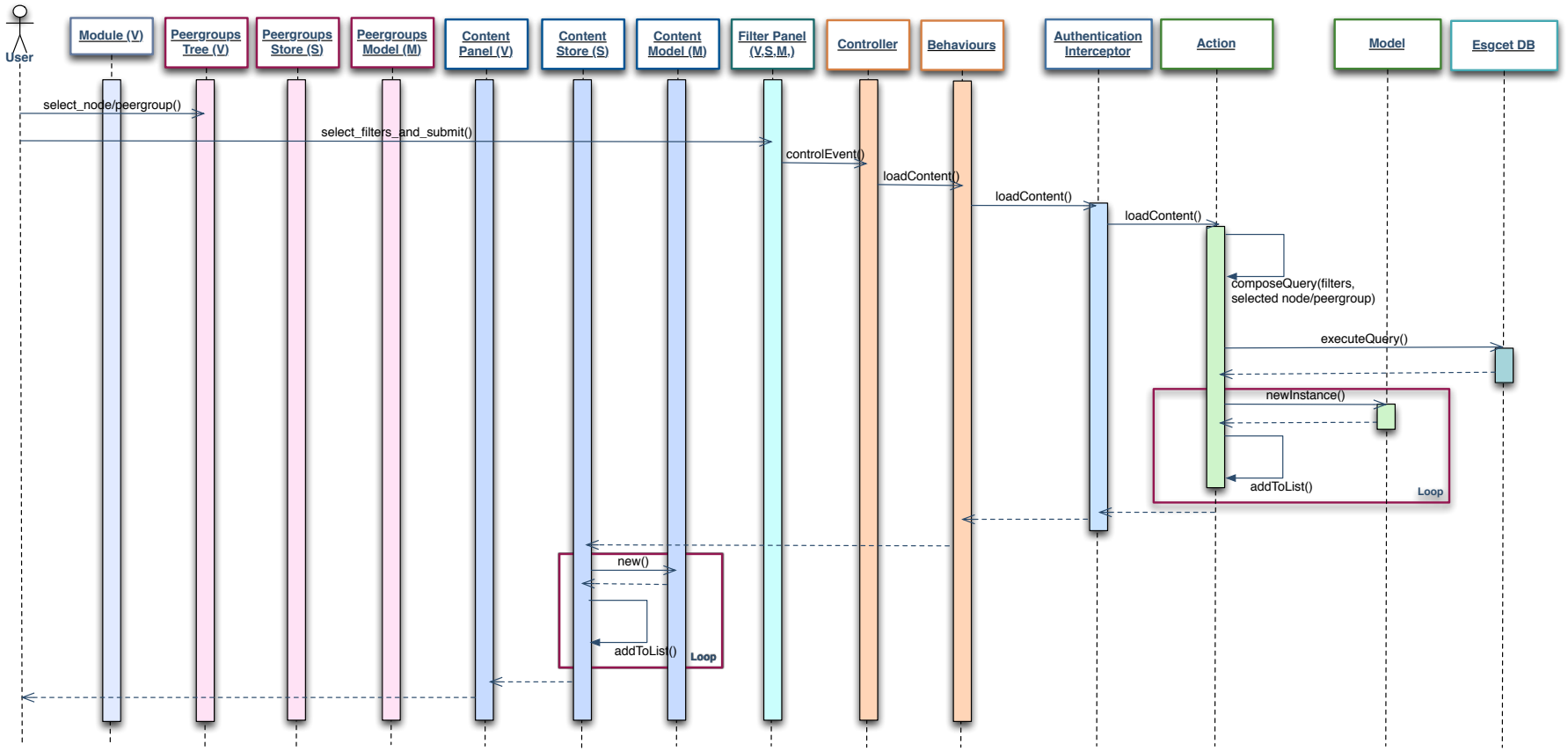
# ESGF Desktop: design

## MODULE STARTUP

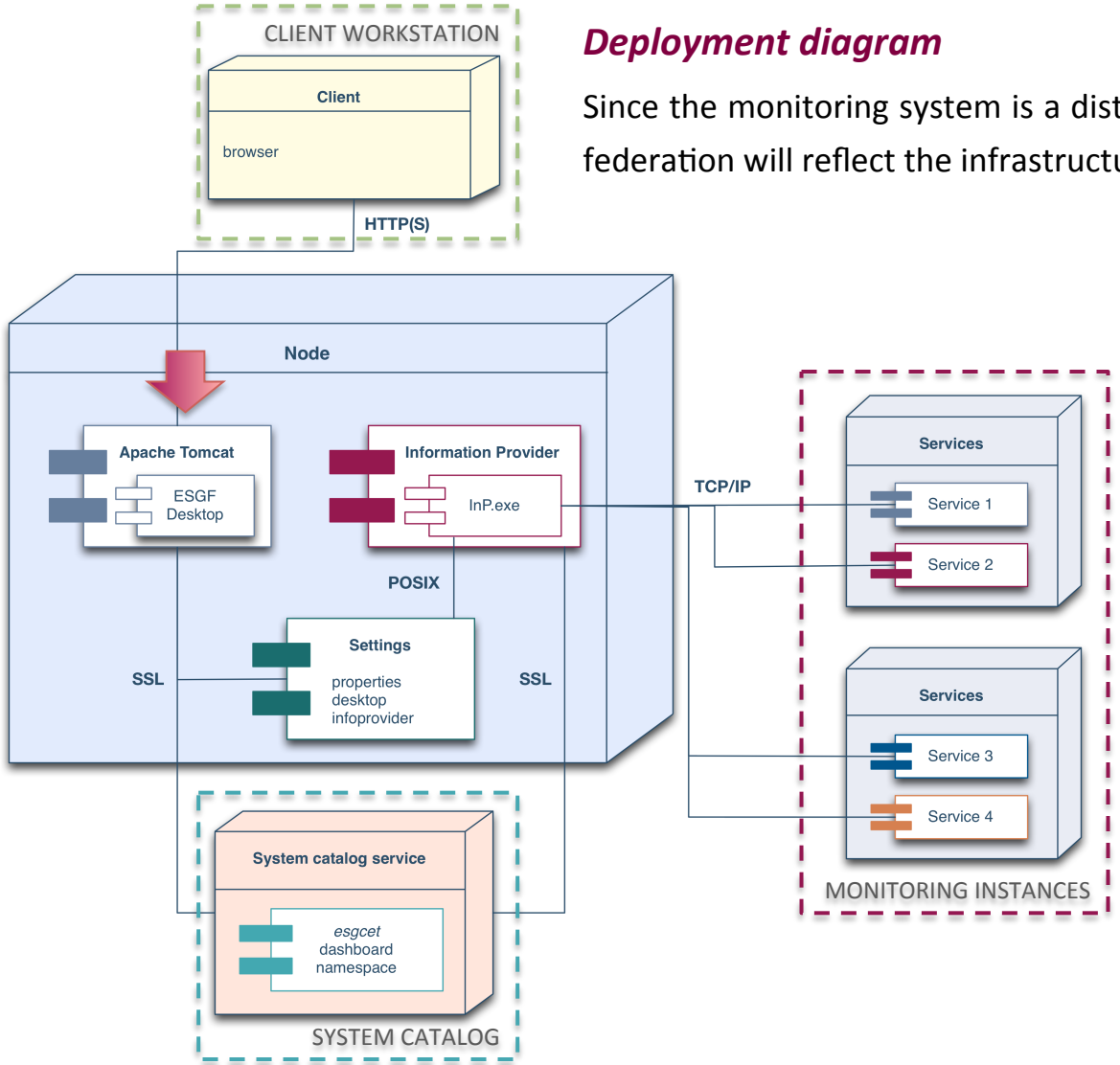


# ESGF Desktop: design

## USER INTERACTION



# ESGF Desktop: design




## *Deployment diagram*

Since the monitoring system is a distributed framework, each host of the ESGF federation will reflect the infrastructure represented here.

Framework components

- System catalog
- Client workstation
- Node configuration
  - Apache Tomcat web server
  - Information Provider script
  - Settings component

Communication protocols

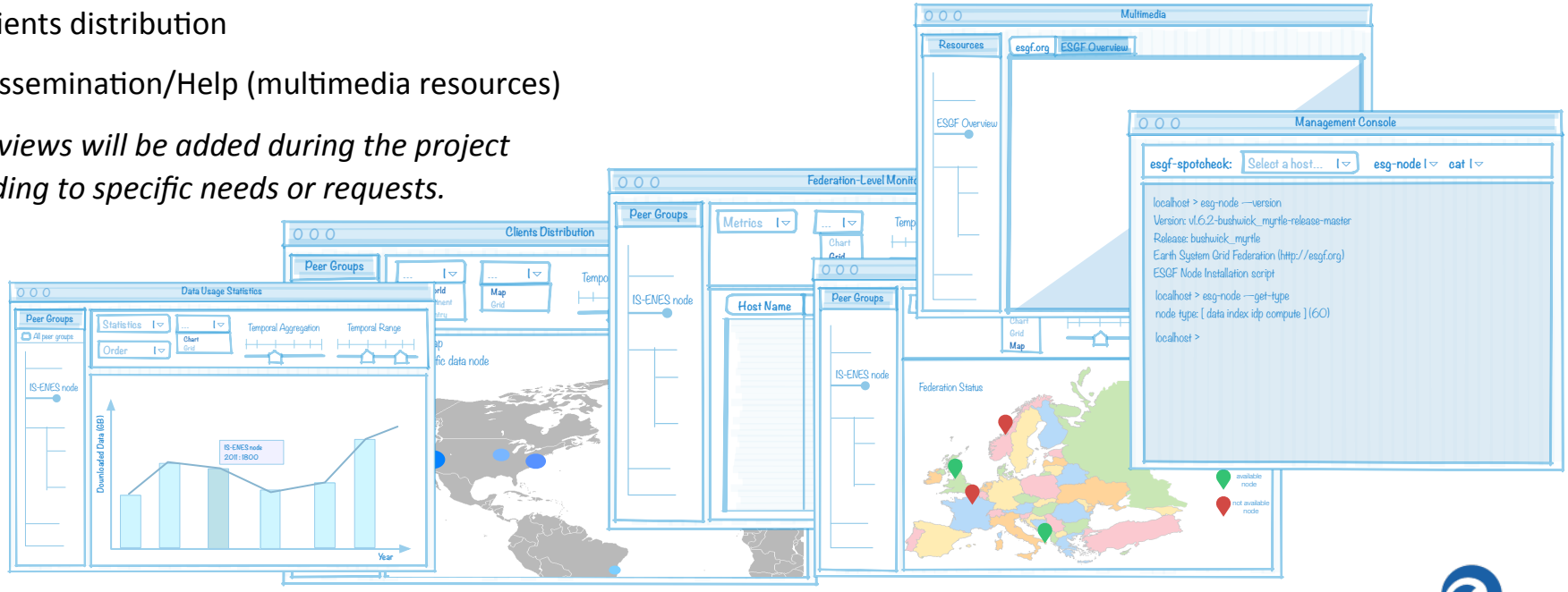
- TCP/IP protocol
- SSL security protocol
- HTTP or HTTPs

# ESGF Desktop: main views

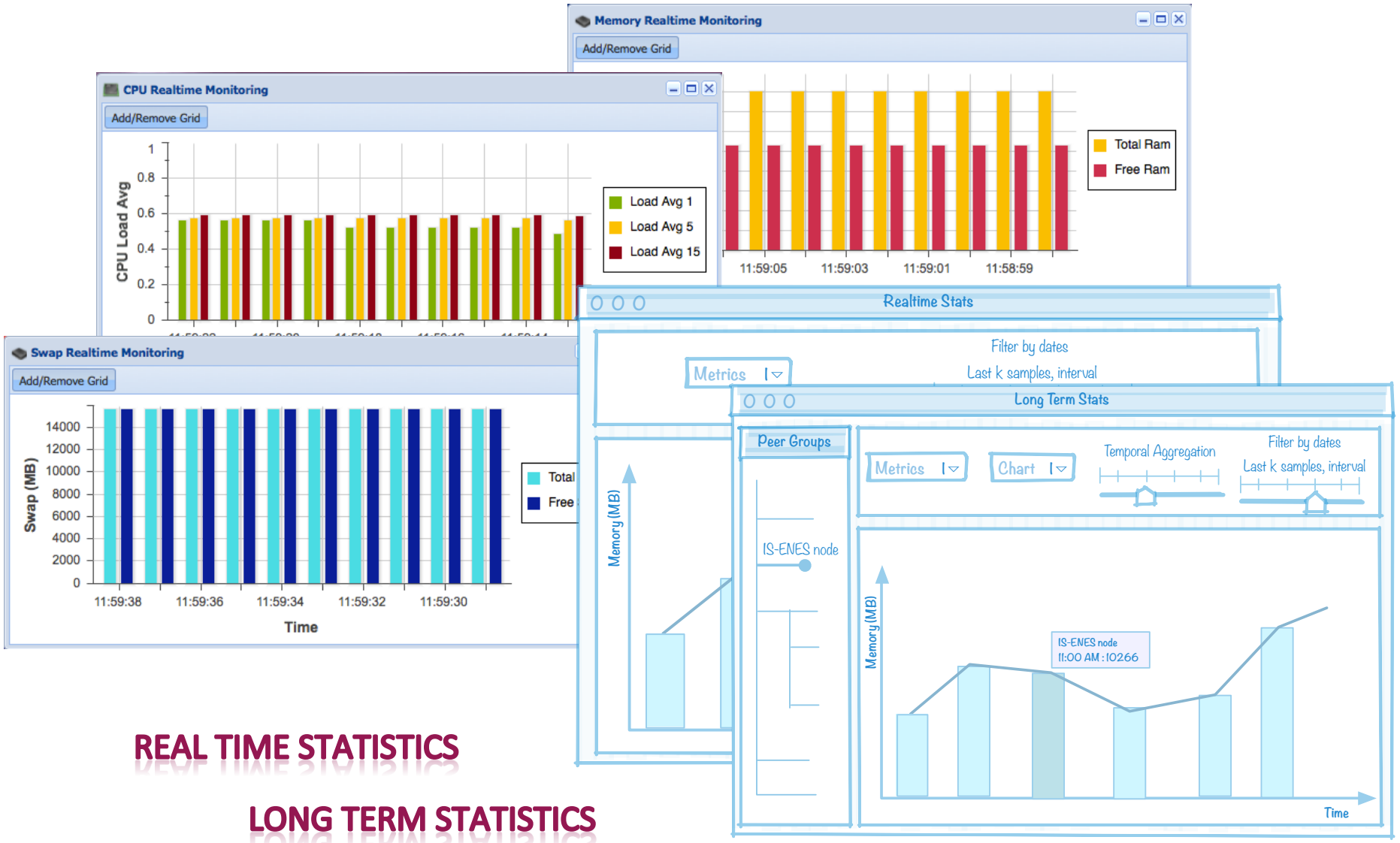
The most relevant views planned in the ESGF Desktop are:

- Real time statistics (e.g. CPU, RAM, SWAP, etc.)
- Long term statistics (e.g. CPU, service availability, registered number of users, search results, etc.)
- Federation level status (e.g. availability, registered users, deployment, bandwidth)
- Data usage statistics (download information)
- Management Console (a terminal-like interactive view, with a well-defined set of commands)
- Clients distribution
- Dissemination/Help (multimedia resources)

*other views will be added during the project according to specific needs or requests.*



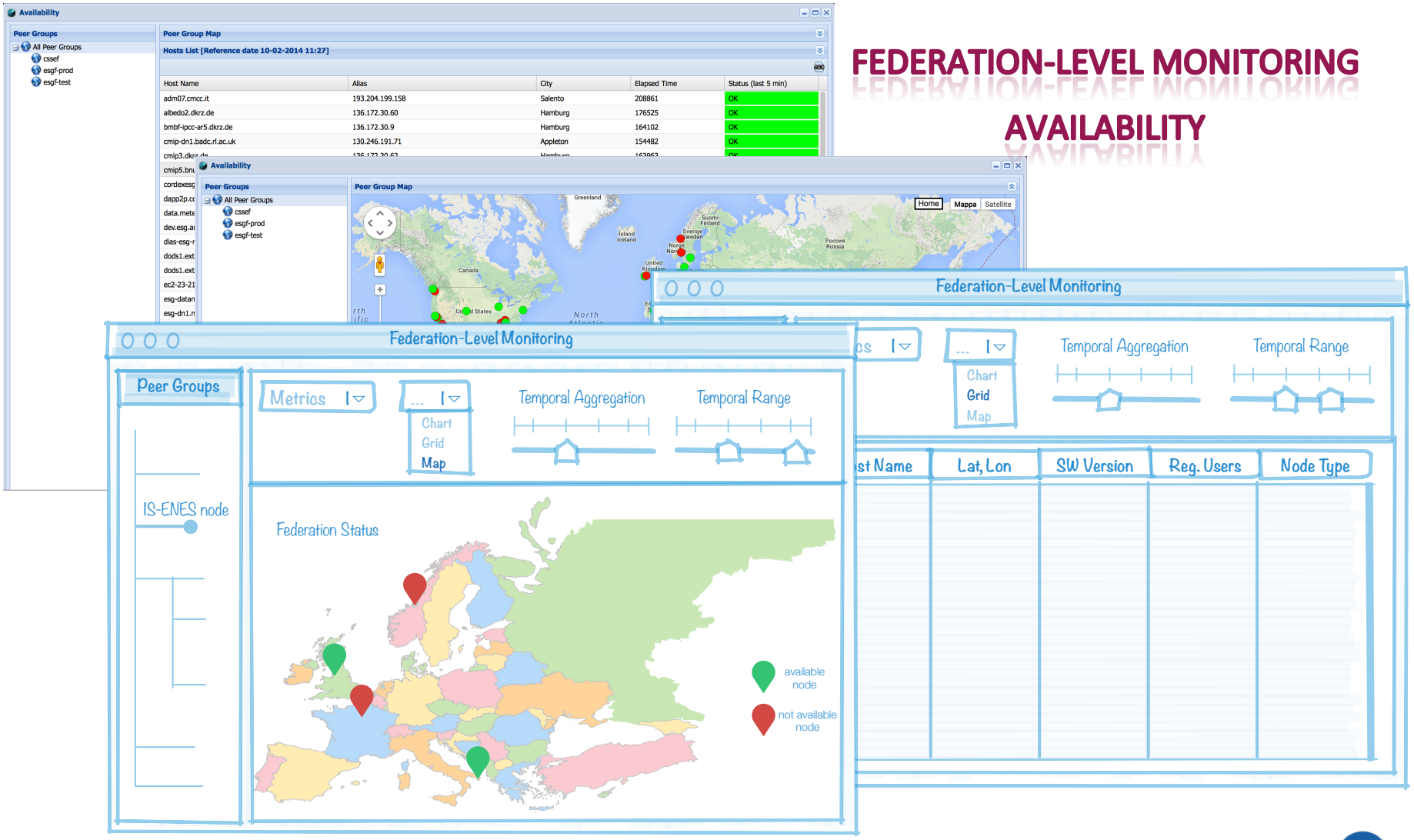
# ESGF Desktop: main views




REAL TIME STATISTICS

LONG TERM STATISTICS

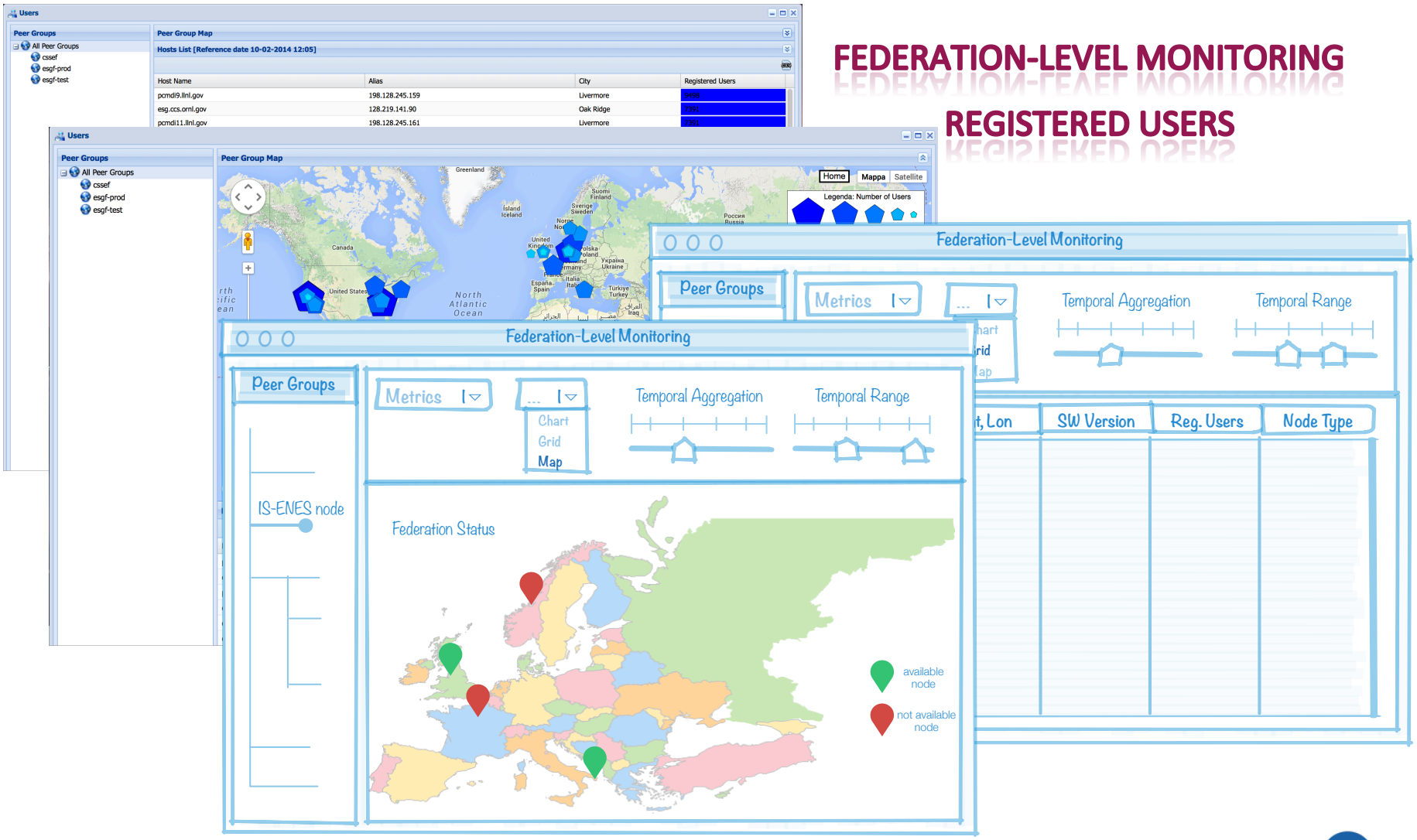
# ESGF Desktop: main views




## FEDERATION-LEVEL MONITORING

## AVAILABILITY

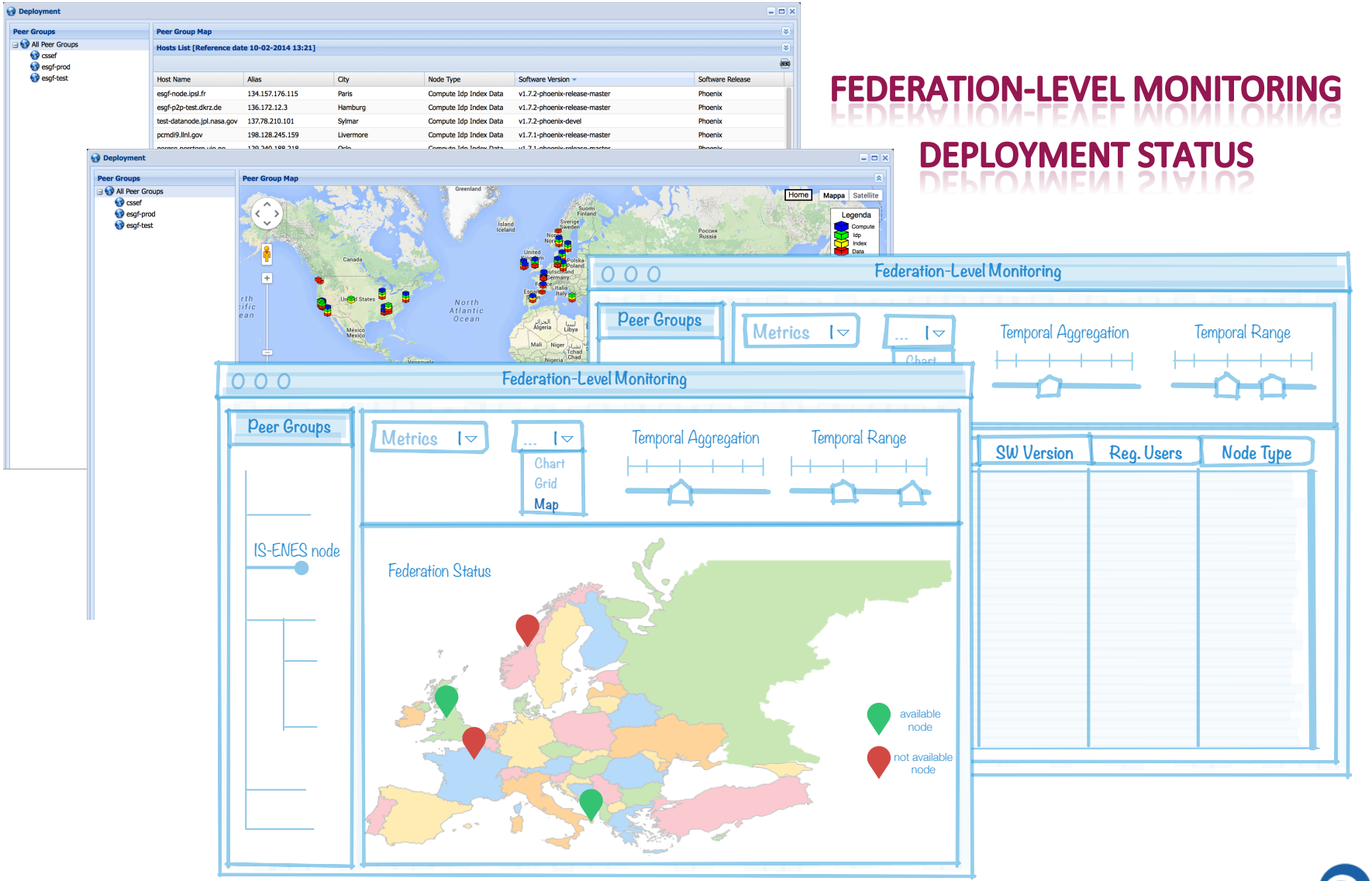
# ESGF Desktop: main views




FEDERATION-LEVEL MONITORING

REGISTERED USERS

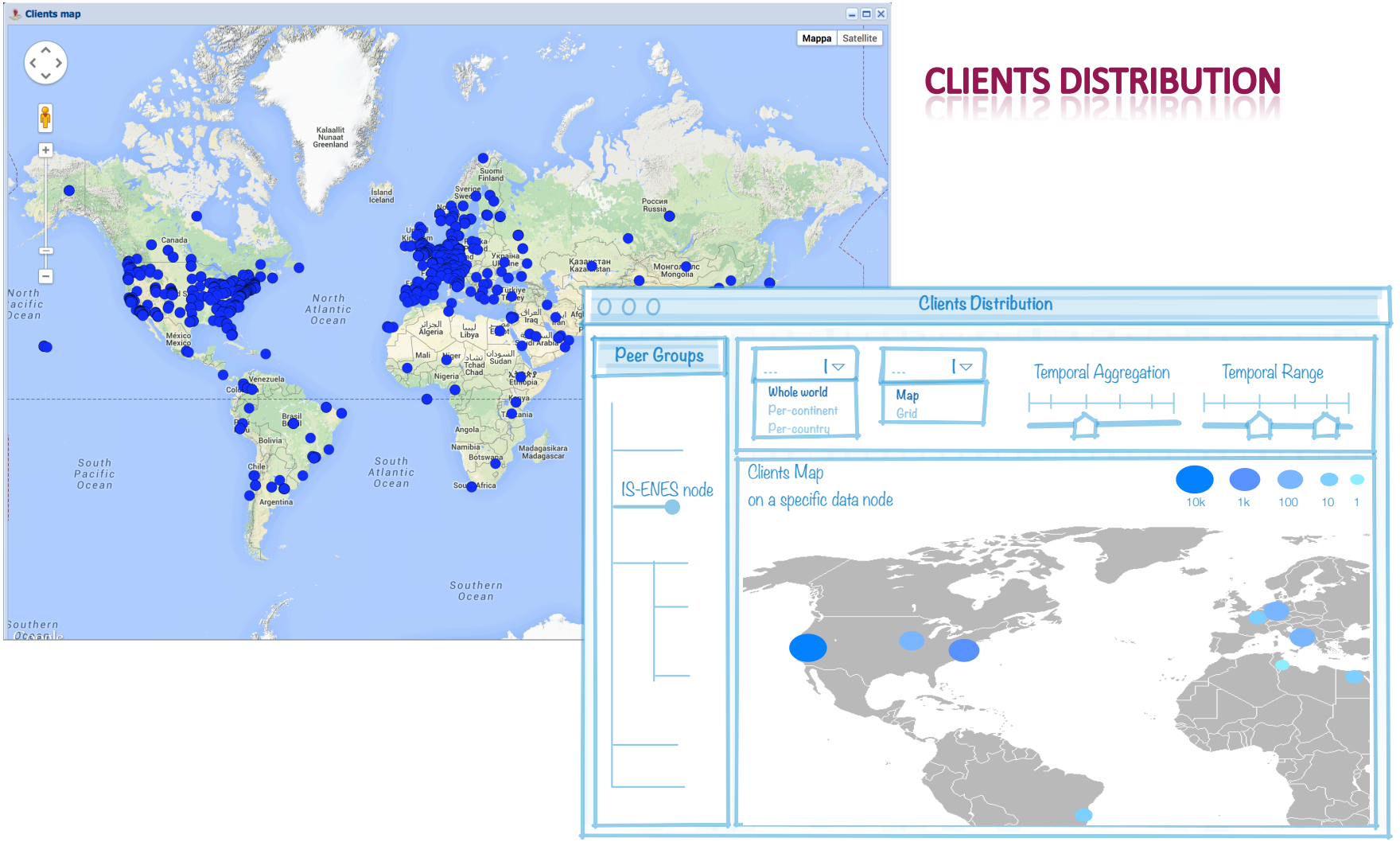
# ESGF Desktop: main views




## FEDERATION-LEVEL MONITORING

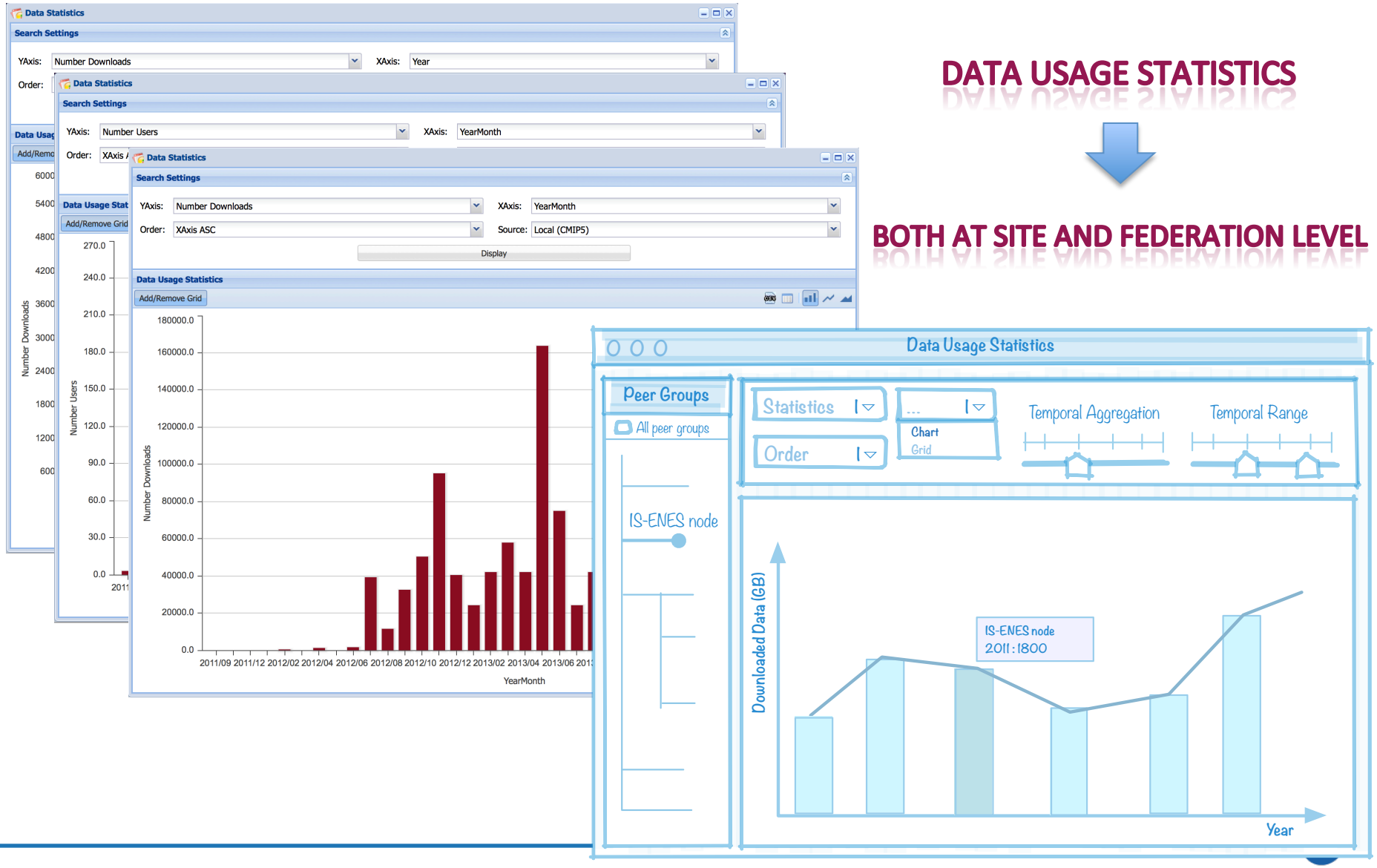
## DEPLOYMENT STATUS

# ESGF Desktop: main views

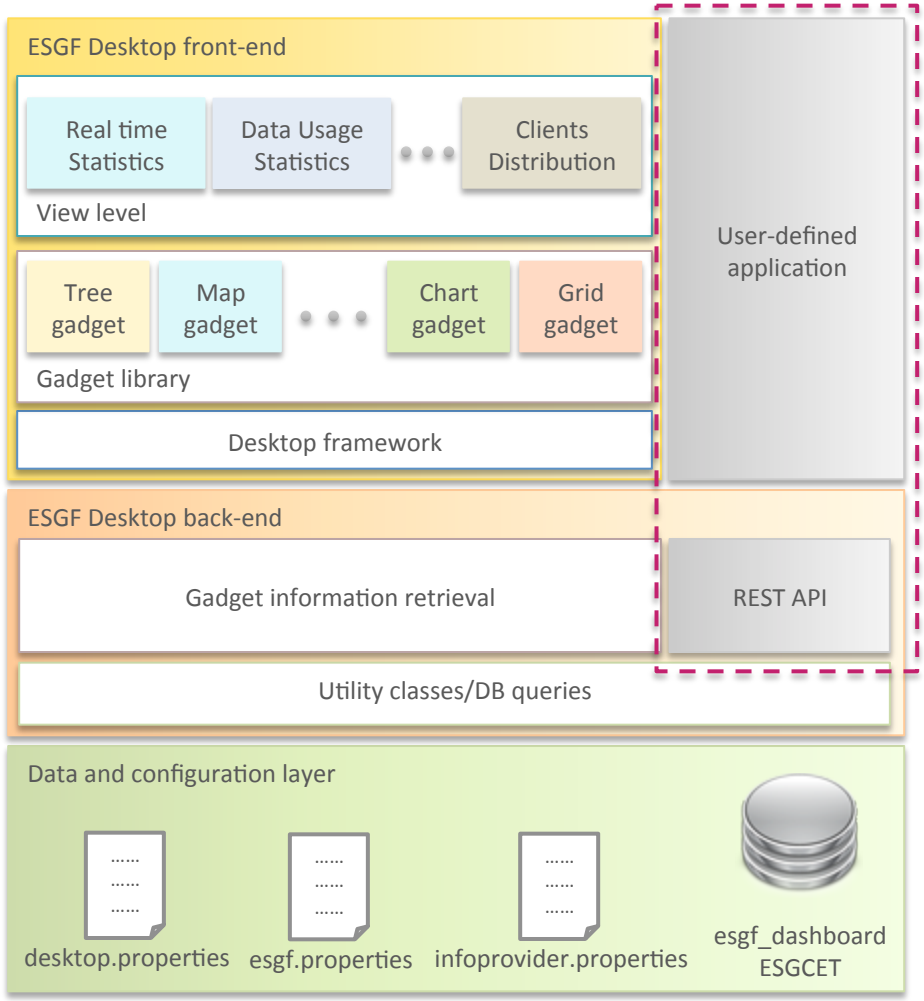



## CLIENTS DISTRIBUTION

# ESGF Desktop: main views



# ESGF Desktop RESTful APIs



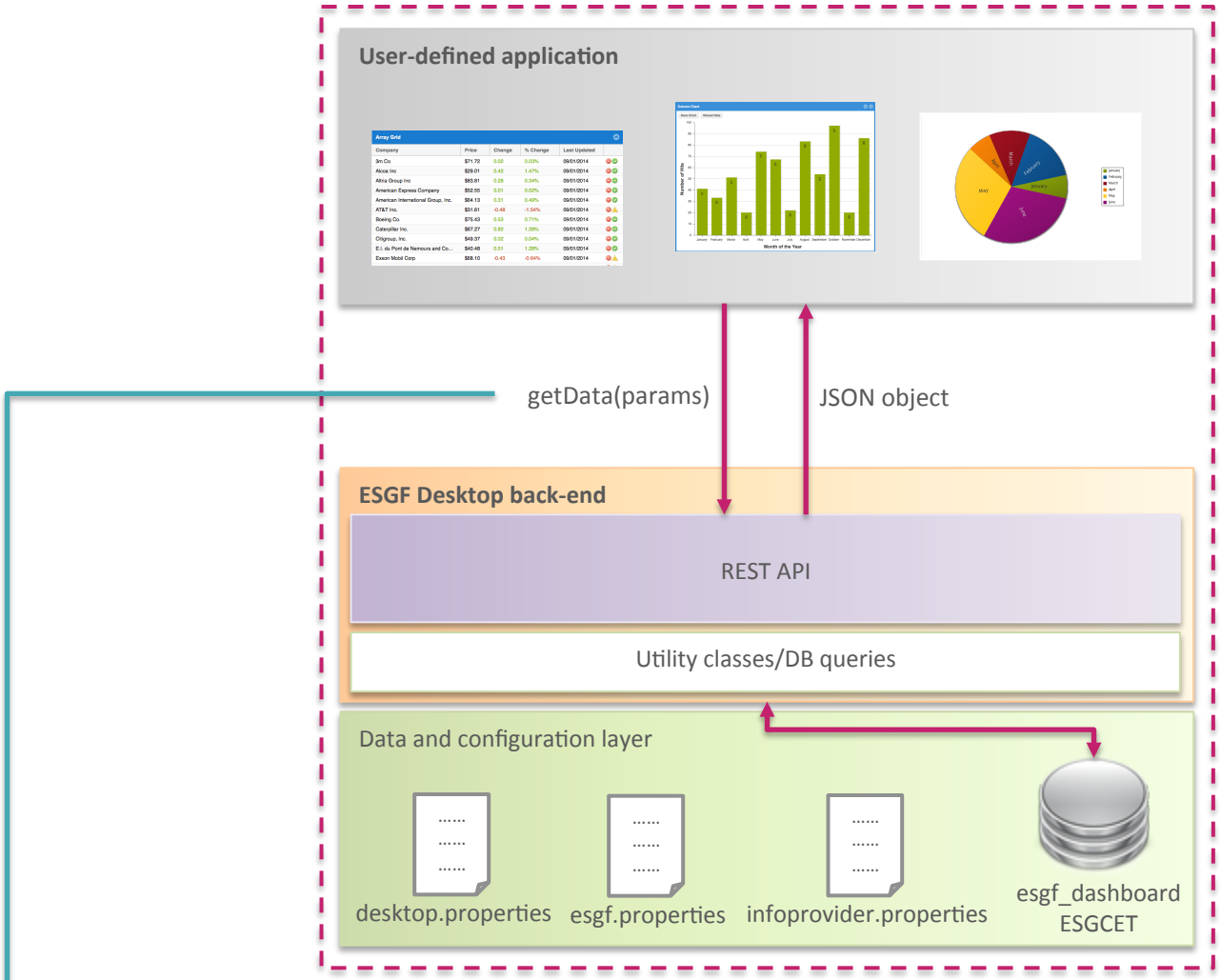

To provide the user with a **programmatic access** to the metrics managed by the Dashboard, a **RE**presentational **S**tate **T**ransfer (**REST**) API was defined.

Users can define and implement their own applications and query a node of the federation to retrieve the desired information.
The response of the node will be sent to the application on HTTP protocol and will consist of a **JSON object** containing the data required.

It will be possible to query the nodes for:

- host information
- peer groups
- system statistics (including real time stats)
- data usage statistics

# RESTful APIs




http://ESGFnode:port/esgf-desktop/RESTapi/loadDatastats.action?metric=metric&dimension=dimension&order=order&type=type&startdate=startdate&enddate=enddate

# Conclusions and future work

- A design report of the ESGF dashboard and ESGF desktop components (IS-ENES2 deliverable)
  - Back-end and front-end design
  - System catalog *esgcet* with the namespace *esgf_dashboard*
  - Multiple views for system and user metrics
  - REST API for programmatic access to the statistics
- From an implementation point of view the existing framework is being improved/ extended taking into account the design report
  - Extension of the existing views to support new requirements coming from the community
  - the design and implementation of new views
  - Federation level mechanisms for global statistics
- Close collaboration with esgf-node-manager developers to investigate logging issues and define a new version of the logging database
  - This is critical to move further towards federation-level statistics, or fine grain statistics by model, variable, dataset, etc.
- Explore interaction/integration with CoG (target 2015)
  - This will rely on the new REST APIs for the dashboard system

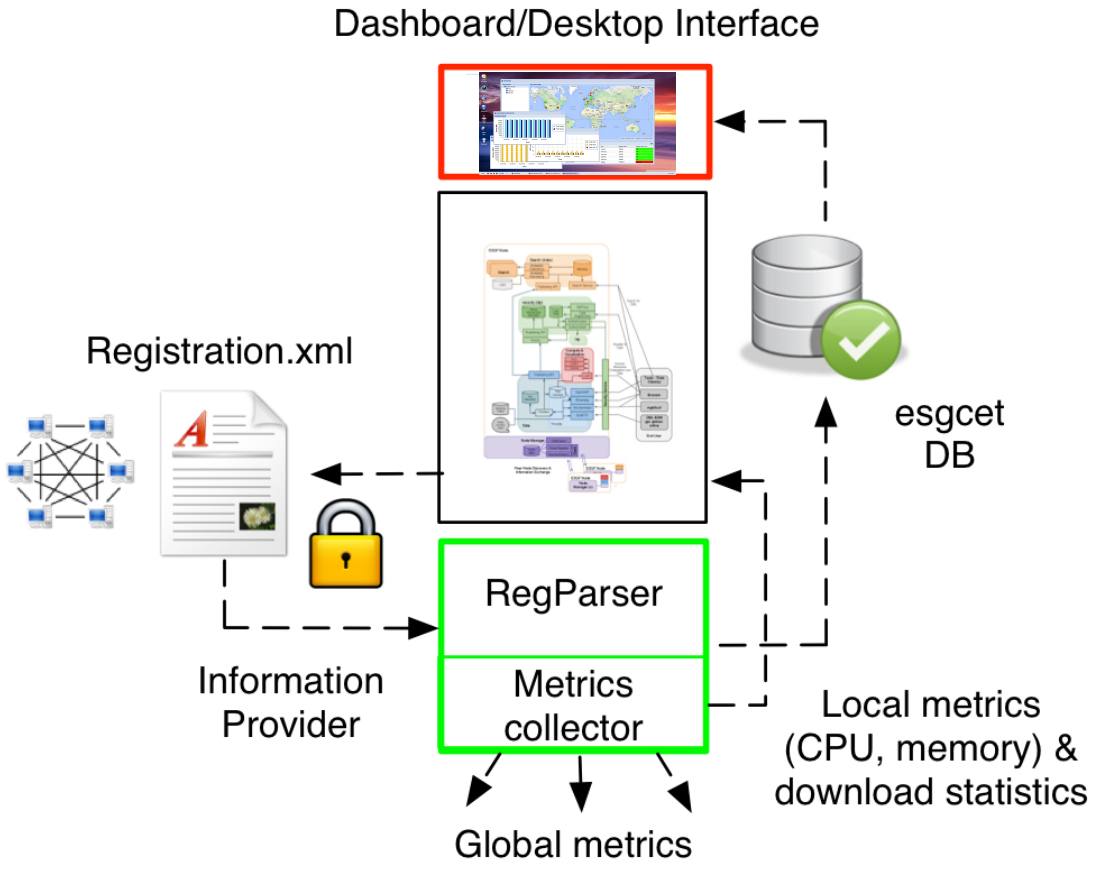
# Thank you

cmcc
Centro Euro-Mediterraneo
sui Cambiamenti Climatici

# Goals

Starting from what has been done in the past, in the last months the activities have focused on **reengineering** the system in order to:

- identify the most relevant classes of users and their expectations
- perform an analysis of the functional and non-functional requirements
- improve the performance and, at the same time, the modularity of the entire system

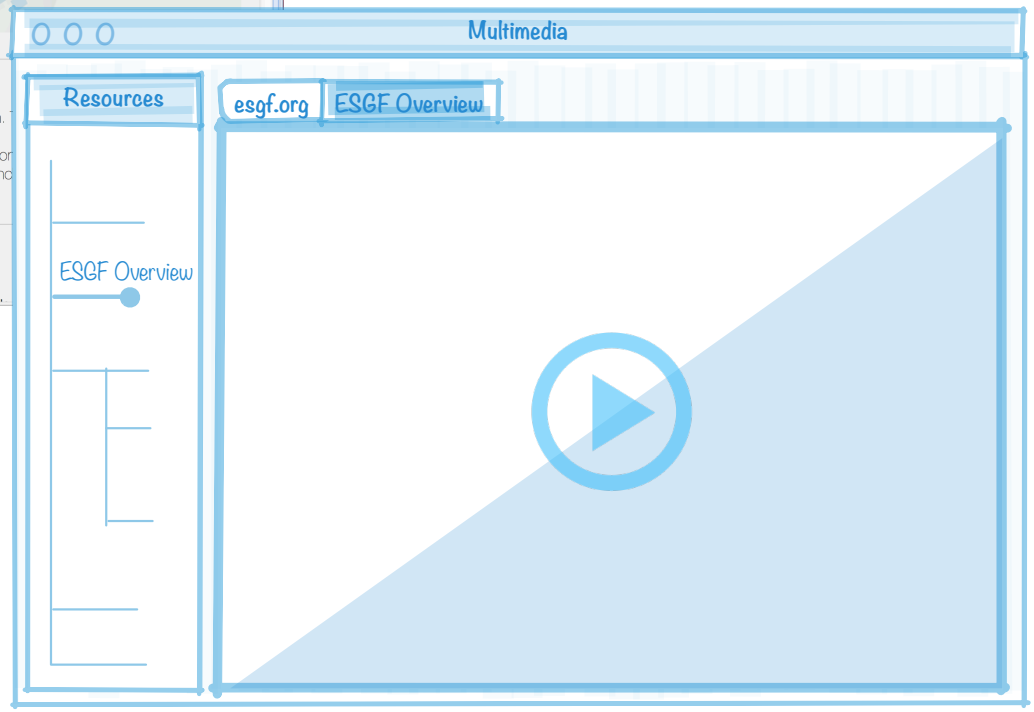


# Conclusions and future work

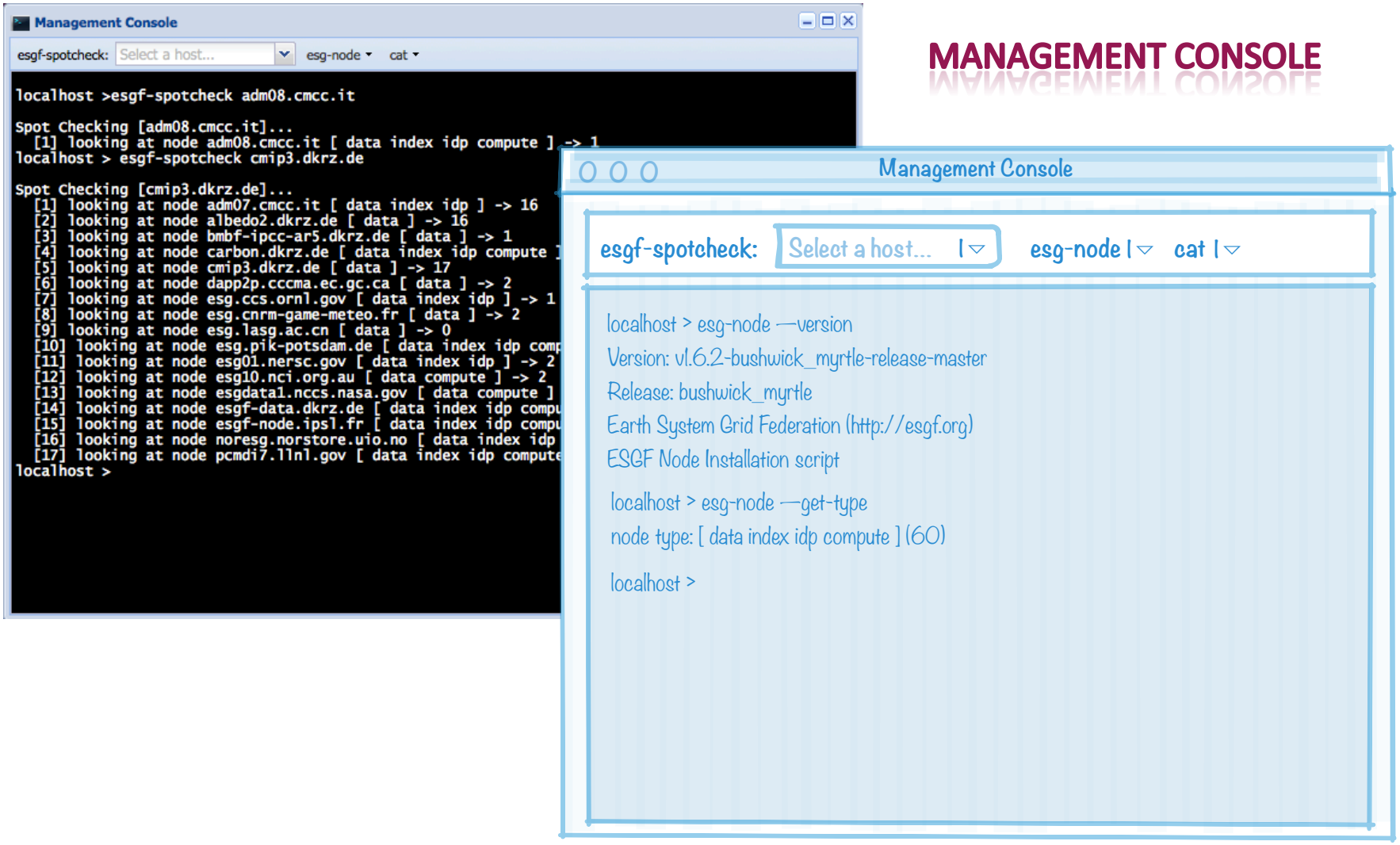
- A design report of the ESGF dashboard and ESGF desktop components (IS-ENES2 deliverable)
  - Back-end and front-end design
  - System catalog *esgcet* with the namespace *esgf_dashboard*
  - Multiple views for system and user metrics
  - REST API for programmatic access to the statistics
- From an implementation point of view the existing framework has been improved taking into account the new design.
  - Extension of the existing views to support new requirements coming from the community
  - the design and implementation of new views
- Close collaboration with esgf-node-manager developers to investigate logging issues and define a new version of the logging database
  - This is critical to move further towards federation-level statistics, or fine grain statistics by model, variable, dataset, etc.

# ESGF Desktop: main views


**DISSEMINATION/HELP**



# ESGF Desktop: main views




MANAGEMENT CONSOLE

# ESGF Dashboard RESTful APIs

## RESTFUL SERVICE EXAMPLES

Sensor statistics: http://ESGFnode:port/esgf-desktop/RESTapi/getSensorStats.action?sensor_table=sensor_table&sensor_name=sensor_name&host_name=host_name

- host_name - DNS name of the host
- sensor_name - name of the sensor related to more specific info about the selected metric
- timestamp - time instant of the value collection and
- a list of metric value related to different time intervals

Data usage stats: http://ESGFnode:port/esgf-desktop/RESTapi/loadDatastats.action?metric=metric&dimension=dimension&order=order&type=type&startdate=startdate&enddate=enddate

- dimension - the value of the required dimension
- measure - the sum of the metric value s grouped according the required dimension

Data usage stats: http://ESGFnode:port/esgf-desktop/RESTapi/loadDatastats.action?metric=metric&dimension=dimension&order=order&type=type&startdate=startdate&enddate=enddate

- dimension - the value of the required dimension
- measure - the sum of the metric value s grouped according the required dimension