TECHNICAL SPECIFICATION

SSN ECOSYSTEM GUI

File Name:	GMV.SEG.TS.001 - Technical Specification
Version:	V1.2
Date:	29/11/2019
Category:	Public
Code:	GMV 24566/19 V1/19
Internal code:	TechnicalSpecification.TS.001

© GMV, 2020; all rights reserved

 © GMV 2020; all rights reserved

DOCUMENT STATUS SHEET

Version	Date	Pages	Changes
V1.2	29/11/2019	44	Updated to SEG 1.9.
V1.1	06/04/2017	35	Updated to SEG 1.0 Incorporated EMSA Comments
V1.0	24/11/2015	33	Initial Version

 © GMV 2020; all rights reserved

TABLE OF CONTENTS

 © GMV 2020; all rights reserved

 © GMV 2020; all rights reserved

LIST OF TABLES AND FIGURES

 © GMV 2020; all rights reserved

1. INTRODUCTION

1.1. PURPOSE

1.2. SCOPE

1.3. DEFINITIONS AND ACRONYMS

1.3.1. DEFINITIONS

Concepts and terms used in this document and needing a definition are included in the following table:

Table 1 Definitions

Concept / Term	Definition

1.3.2. ACRONYMS

Acronyms used in this document and needing a definition are included in the following table:

Table 2 Acronyms

Acronym	Definition

 © GMV 2020; all rights reserved

2. REFERENCES

2.1. APPLICABLE DOCUMENTS

The following documents, of the exact issue shown, form part of this document to the extent specified herein. Applicable documents are those referenced in the Contract or approved by the Approval Authority. They are referenced in this document in the form [AD.X]:

Table 3 Applicable documents

Ref.	Title	Code	Version	Date
[AD.1]				

2.2. REFERENCE DOCUMENTS

The following documents, although not part of this document, amplify or clarify its contents. Reference documents are those not applicable and referenced within this document. They are referenced in this document in the form [RD.X]:

Table 4 Reference documents

Ref.	Title	Code	Version	Date
[RD.1]				

 © GMV 2020; all rights reserved

3. OVERALL SYSTEM ARCHITECTURE

3.1. SYSTEM ARCHITECTURE AND MAJOR SYSTEM ELEMENTS

The technical solution implemented was to build a multi module application following the servlet specification, where in Liferay the application will be added as an external link. This application will fetch the roles/profiles from the CMC UserDetailService taking the information regarding the logged in user from the headers added by the WebGate proxy.

The front end web application will be developed using Angular JS on the client side and Spring MVC version 4.2 for the server components. The application will feature a REST interface to be used by the web application for information updates on the web user interface.

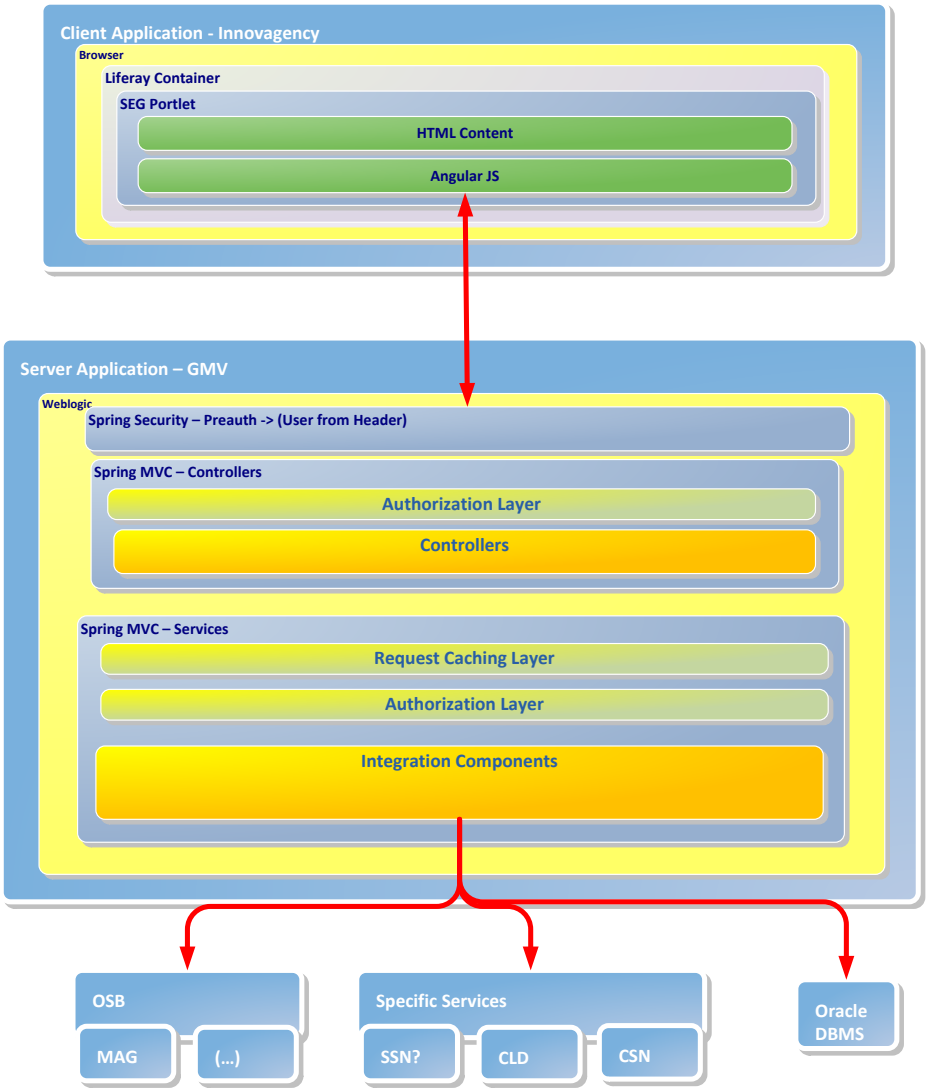
The application will then connect to the remaining EMSA services via HTTP, preferably the communication will be through EMSA's Oracle Service Bus that will proxy all internal EMSA services hiding future interface changes from SEG. Alternatively the SEG is able to connect directly to the services via REST or SOAP interfaces.

The MVC software architectural pattern will be applied on the whole application with the following element:

- Model: The objects of the Model, hosted on the backend server, will be modified or requested by the controller. The controller will select which model component will perform the desired action, and call it with 2 basic actions:
  - View current model status to fetch the current active version of the item on the database
  - Update current model status to create a new version or update a specific version of the chemical database
- Controller: Will provide the information requested by the view and relay changes to the correct underlying model object. The controller will be present on the front end weblogic server.
- View: Will request model information using the controller actions. This element of the pattern is the GUI of the application.

The image below shows the proposed high level system architecture.

 © GMV 2020; all rights reserved



Picture 1 - System elements

Integration with Liferay (Assuming Iframe approach, to be reviewed later)

The WebGate will provide a first level filtering on resources avoiding illegal access to the application on a first level.

Front-End applications will run outside Liferay and presented as portlets using Liferay's Iframe portlet. The application will have the following infrastucture:

- Initial HTML page served to Liferay to the Iframe portlet.
- REST-JSON Services for web pages served through Spring MVC controllers exposed as a servlet
- Permissions Management In Liferay:
    - Login performed on Liferay
    - Access restriction to the porlet performed by Liferay
    - Logged in user retrieved from HTTP Headers from WebGate

Logged in user fetched by Spring Security – Preauth and used for permissions checks

Technologies used:

The table below shows the main libraries to be used on the TOR system

 © GMV 2020; all rights reserved

Technology	Version	Key benefits	Used for
AgularJS	1.4.8	Clean, maintainable and unit testeable single page web applications. Support for all major browsers	Client side logic implementation
WebGate	-	Secure sessions, Single Sign on capabilities	SSO token validation, HTTP header enrichment with the current user
Weblogic Server	12.1.3	Java EE container with clustering capabilities. Providing transparent database access through Data Sources	Application server supporting the server components of the application
Spring Core Framework	4.2.4	Testable lightweight IoC framework	Base IoC framework used for server components development
Spring MVC Framework	4.2.4	Clean division between controllers, views and model. View Agnostic. Testable.	Creating the REST interface for remote communication between components.
Spring Security	4.0.3	Security as a cross concern on the whole application Extensible framework	User Authorization
QueryDSL	4.0.7	Type Safe SQL queries Code completion with SQL	Creating SQL queries to be performed to the database.
Apache Ignite	1.9.0	On Heap and Off Heap partitioned data grid	Caching of requests
Oracle Exadata	-		Data Persistence

 © GMV 2020; all rights reserved

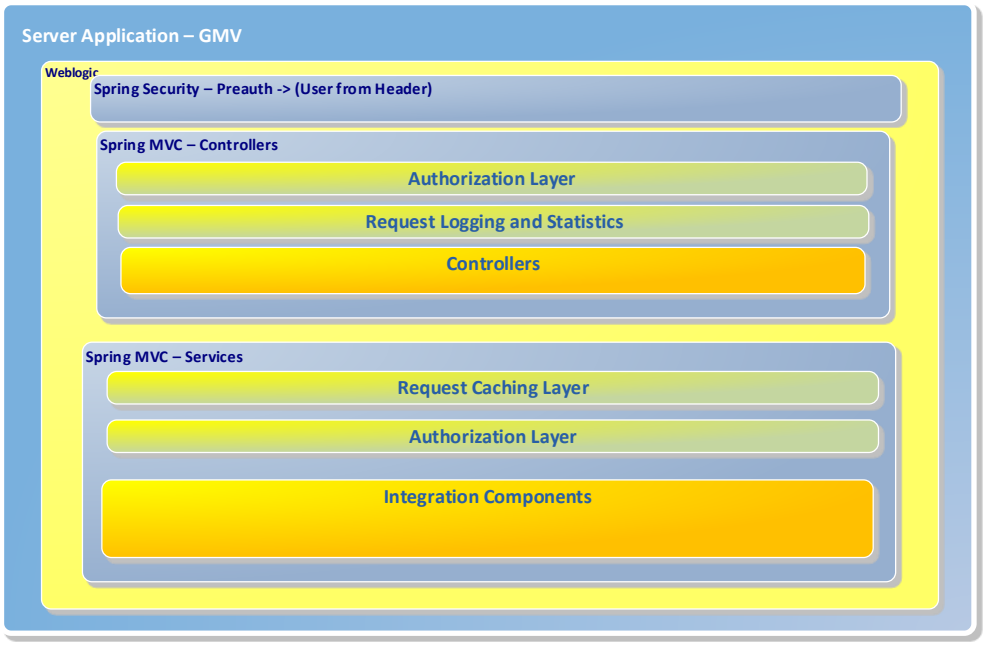
3.2. SYSTEM ARCHITECTURE

3.2.1. COMPONENTS DESCRIPTION

The system will be deployed over a WebLogic application server.

The following modular architecture is proposed for the system development:

- Spring Security – Preauth: The SEG will use this layer to identify the user making the request and obtain the authorities for the user from EMSA the already existing EMSA authenticated system.
- Authorization Layer (Controllers and Services): Once the user is authenticated (in the "pre-authenticated" scenario), this layer get the user details (role, request, etc…) to check if is granted to use the Controller / Service.
- Controllers: This layer contains the Controller component of the Web MVC framework. This layer typically will prepare a model Map with data writing it directly to the response stream and complete the request serialized as JSON.
- Services: This layer represents the Model (implemented as Spring Services/Repositories) to be used by the Controller layer.
- Request Catching layer: This layer increase the throughput of the system by executing only once expensive methods (whether CPU or IO bound) for a given set of parameters and the result reused without having to actually execute the method again.
- Integration component: This layer will contain the components developed for the SEG integration with other existing EMSA applications. This layer encapsulates different communication protocols (REST/SOAP) into a single interface to be used by the Business Layer components.



Picture 2 - SEG architecture

The following chapter will describe more detailed each of the layers of the system architecture.

3.2.1.1. Spring Security – Preauth

"Pre-authenticated" scenarios is the situation where the system to be developed will use Spring Security for authorization, but the user has already been reliably authenticated by the IdM. When using pre-authentication, Spring Security has to:

1. Identify the user making the request.
2. Obtain the authorities for the user.

 © GMV 2020; all rights reserved

The details will depend on the external authentication mechanism. For the SEG the user will be passed via an HTTP header. With ths user in the case of IdM the authorities must be obtained from a separate source, the CMC UserDetailsService that will fetch the user, his roles and operations from IdM.

3.2.1.2. Controllers

This layer contains the Controller component of the Web MVC framework. For the SEG application, the controller layer will generate the REST-Like interface to be used by the view component.

REST has quickly become the de-facto standard for building web services on the web because they’re easy to build and easy to consume. RESTfull web services take all benefits that come for free with HTTP as a platform itself. Application security (encryption and authentication) are known quantities today for which there are known solutions. Caching is built into the protocol. Service routing, through DNS, is a resilient and well-known system already ubiquitously support

It will be the responsibility of the controller layer the transformation from JSON to domain Objects and the selection of the correct Model component to process the request.

3.2.1.2.1. Authorization

After the user information is retrieved the control is passed to the SEG Security aspect. This aspect will fetch from the SEG database the permissions specific to the resource the user is calling and validate the required roles and operations against the user. The SEG authorization mechanism also allows the filtering of output based on expressions, currently configured for EO operations, only allowing the user to see images from the operations he has access to.

3.2.1.2.2. Logging and Statistics

Each request to the SEG is logged into the database storing the user performing the request, the resource requested, the total time it took to process the request and the external time (calls to external services).

3.2.1.3. Services

SEG will implement a façade to its business logic from the controllers via Spring services. These services can be seen as originally defined by Domain-Driven Design (Evans, 2003) as "an operation offered as an interface that stands alone in the model, with no encapsulated state."

If a state is required this will not be implemented on the service layer but under other component belonging to the business logic.

3.2.1.4. Request Caching layer

For the caching approach SEG will use a simple approach, a caching service will be implemented for each of the integration components, this caching service allows reducing thus the number of calls based on the information available in the cache. That is, each time an external interface is invoked, the abstraction will apply a caching behavior checking whether the method has been already executed for the given arguments. If it has, then the cached result is returned without having to execute the actual method; if it has not, then method is executed, the result cached and returned to the user so that, the next time the method is invoked, the cached result is returned. This way, expensive methods (whether CPU or IO bound) can be executed only once for a given set of parameters and the result reused without having to actually execute the method again. The caching logic is applied transparently.

For the SEG the underlying technology for caching will be set to Apache Ignite using on heap and off heap storage to reduce GC overhead.

3.2.1.5. Integration component

The integration layer for the SEG will hide the complexity of communicating to the OSB with services implemented by different contractors on which the same data is returned on different formats. This layer ensures a consistent interface for the upper layers is maintained, and the same data is returned always under a consistent taxonomy.

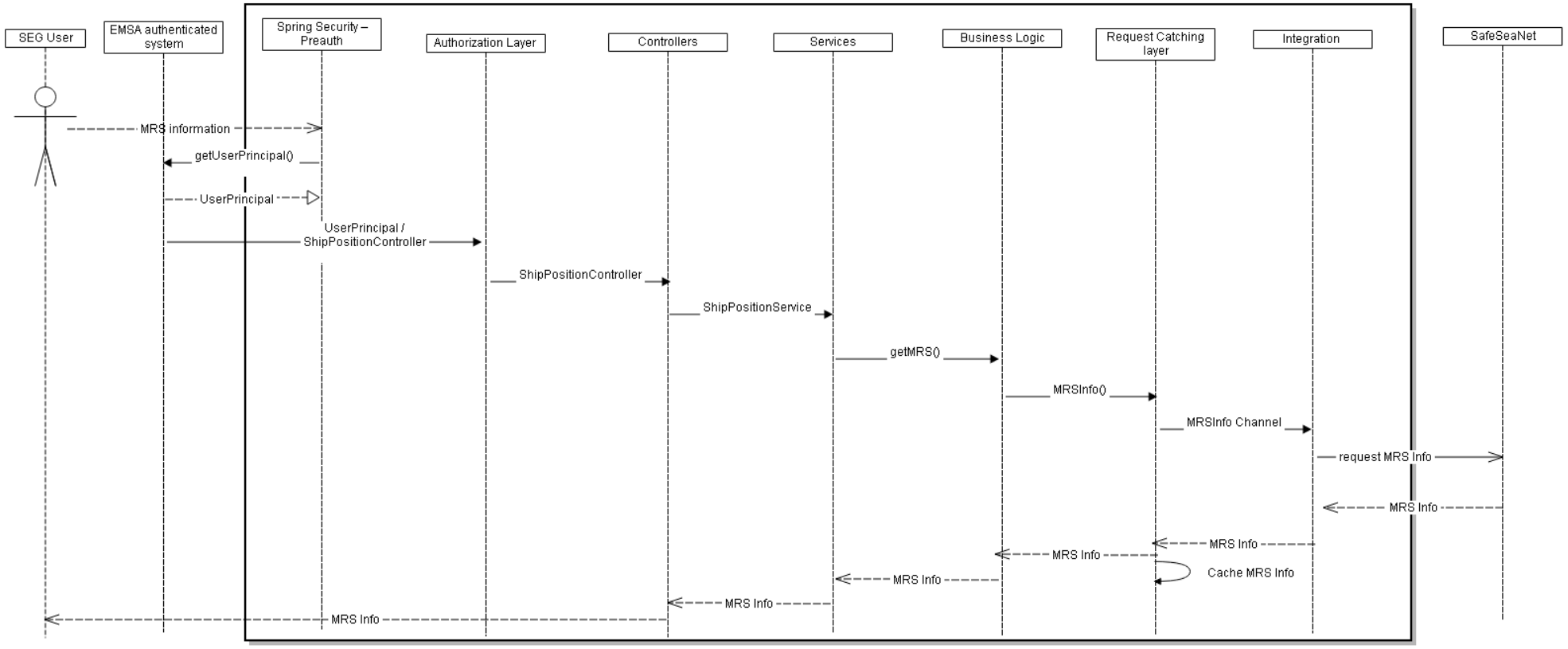
This layer will be able to integrate with the OSB and other services with the following protocols:

- REST
- SOAP

 © GMV 2020; all rights reserved

The following picture provides the overall component functionalities though a sequence diagram requesting MRS information to SSN:

 © GMV 2020; all rights reserved



 © GMV 2020; all rights reserved

3.3. DESIGN DECISIONS

3.3.1. ARCHITECTURAL PATTERNS

Service Oriented Architecture (SOA)

A service-oriented architecture has advantages by allowing the business to respond effectively to changes both in terms of speed and costs. This style of architecture allows for reuse of components as services. This architecture also facilitates integration with external systems integrating as other services in the system.

SOA is design pattern that aims to build systems with high interoperability capabilities through information exchange, reuse and composition, thus allowing the construction of complex software systems using a set of interdependent and universally interconnected services. Services can be invoked through a network or a client that uses a different type of technology and should therefore be interoperable. Discovery and lookup is another property of a service-oriented architecture that refers to the ability of a client to search for a service that matches your needs dynamically at runtime.

In SEG in communication between the various internal services will be left to the IOC container and configurable between direct calls (when they are deployed on the same server); RMI when deployed on the same network and REST-WS when interaction happens across domains.

The approach of the SEG SOA based on Spring Framework (V4.2) also allows the distribution of the various services over the network allowing, should you need to move a set of services to a different server in order to share the load among available servers.

Multi-Layer Architecture

The whole system is designed in independent layers with well-defined responsibilities. An N-Tier approach allows logical separation of the components between presentation layer, application-layer and data layer.

- **Presentation Layer**: Layer with the responsibility of presenting information to the user and allowing interaction with the system by inserting data and performing actions. On the figure they are represented as the interfaces.
- **Application Layer:** Layer responsible for the coordination of the application, where the business logic is implemented. This layer also has the responsibility to pass, filter and modify the data between the data layer and the Presentation layer and the Integration Layer.
- **Integration Layer:** The integration layer has as main responsibility the interaction and transformation of data for communication with external systems.
- **Data Layer:** The data layer has as a sole responsibility the storage of specific SEG data for later retrieval, in the case of SEG it interfaces with the Oracle database, as an abstraction to the Data Layer in SEG project, the JOOQ library will be used. The database will be located via a JNDI Lookup and a datasource configured at the weblogic portal.

Model View Controller

The usage of the Model View Controller design pattern allows the isolation of the several components , restricting for example a change on the way information is presented to the user to the "View" component, thus increasing system maintainability.

The MVC splits between information presentation, user actions control and the domain model.

The MVC design pattern is split into 3 large components:

- Model: The Model manages behaviour and application data as a whole, responding requests for the current state from the "View" and changing state information on requests from the "Controller".
- View: The View is responsible to manage the information presented to the user from the "Model".
- Controller: The controller interprets the actions from the user and notifies the "Model" or the "View" so these change their state in accordance.

 © GMV 2020; all rights reserved

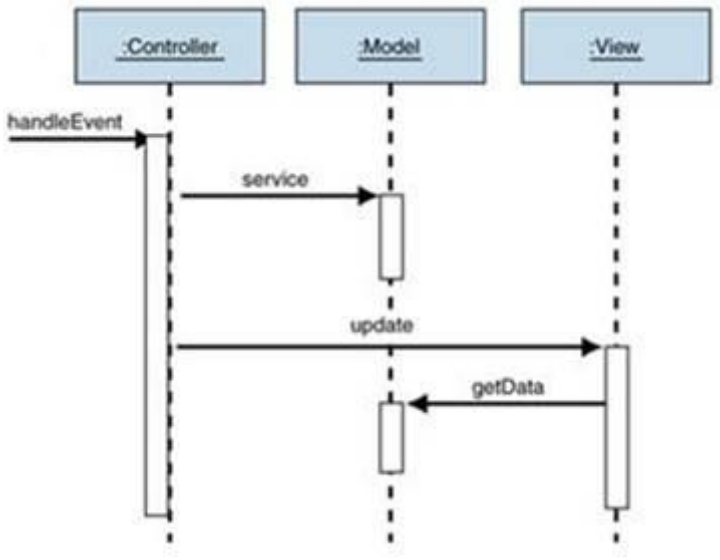


Figura 4-2: Sequência MVC

The MVC pattern should follow this motto:
"We need SMART Models, THIN Controllers, and DUMB Views"

On the SEG system the MVC model will be allowed by the usage of the Library Spring MVC (V4.2)

Inversion Of Control / Dependency Injection

Inversion of Control is an architectural pattern that allows for system components decoupling, as it is no longer needed for the components to know the specific component for all the tasks and operations it needs. In inversion of control there is an externalization of dependency management, these being injected at runtime by the framework or application server on which the application runs.

It is commonly said that "Inversion of Control" follows the "Hollywood Principle": "don't call us, we will call you"

The advantage of this pattern is to allows the change of business rules (specific tasks) without the need for any further change on the system, being just enough to configure the "injector" to use the new implementation.

In SEG IoC will be used extensively on the service attribution, where the client does not need to know a build time where the actual implementation is (Which class implements and on which server it is) juts its characteristics (Interface) being the Framework's responsibility to search for the implementation and inject it on the client.

The framework used in SEG for dependency injection will be Spring Core (V4.2)

Role Based Security

In SEG, following EMSA's IdM and CARD, security will be implemented using profiles, when each user is created a role is attributed to him that governs his permissions. The permissions are therefore assigned to the Roles and not to specific system users.

The profile validation is done on every layer of the application, providing security in depth, each access to the services available is validated guaranteeing the security at every level of the application.

The security implementation on SEG will be based on the Spring Security framework (V4.0.3). This framework will link to IdM using the PreAuthentication mechanism and to the CMC UserDetailService to fetch the user roles and permissions.

 © GMV 2020; all rights reserved

4. PHYSICAL SYSTEM ARCHITECTURE

The SEG System will be installed on 3 different environments at EMSA. Installation on all environments in logical components terms will be equal, only changing configuration items like IP addresses and number of nodes on the clusters.

4.1. PHYSICAL VIEW

4.1.1. AVAILABILITY

The availability expectations of a system relate to how many hours in the day, days per week, and weeks per year the application is going to be available to its users and how quickly they should be able to recover from failures. Since the system includes Software (including applications), Hardware and Network components, this requirement extends to all three of them.

The SEG System is expected by users to be available 24 hours/day, 7 days a week. This will be ensured by the application near stateless design and a combination of multi node clusters and a business continuity facility. Additionally the system is monitored by EMSA on a 24/7 basis to ensure a quick response in case of any incident.

To address incidents an Operations manual will be produced with steps to re-establish service levels in case of many unexpected service failures

The following mechanisms ensure the needed availability for the SEG:

- Near Stateless application design that allows clustering and will not be a bottleneck for horizontal scalability should the performance requirements not be met.
- Clustered deployment.
- Continuous monitoring of the application
- Business continuity facility.

 © GMV 2020; all rights reserved

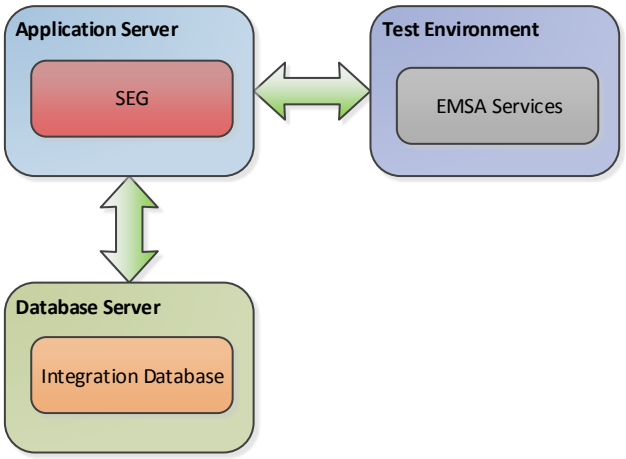
4.1.2. ENVIRONMENT

4.1.2.1. Integration Environment

The integration environment is a simplified version to be used by GMV and Innovagency to perform the initial integration tests with the following objectives:

- Execute some initial requests to create a service mock at GMV premises to facilitate development and automated testing
- Execute tests after implementation to check its correctness.

The following diagram represents the integration environment:



In terms of requirements for this environment only the SEG server needs any special attention as the other services are pre-existing at EMSA and the SEG will not play any significant load on them.

HW:

- 2 cpu cores at 2 ghz
- 4GB of RAM
- 50GB HDD

SW:

- Red Hat Linux (same version as production)
- Weblogic 12c (12.1.3)
- JDK 1.8

Database Server:

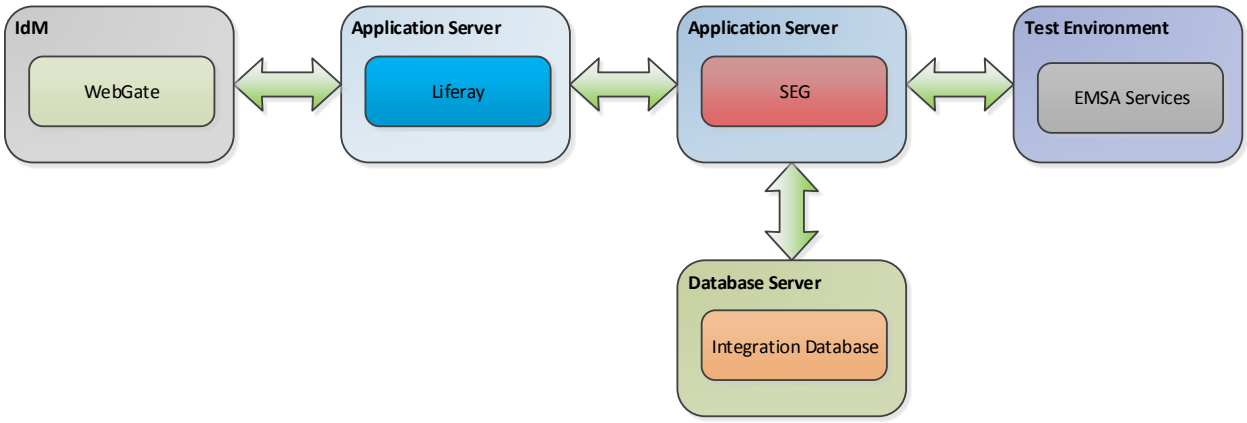
- 2 cpu cores at 2 ghz
- 4GB of RAM
- 50GB HDD

4.1.2.2. Test Environment

The test environment will be used by EMSA to perform functional testing and to check the installation instructions before moving to the next environments.

For this environment a Liferay Server is required as the user will login into the system via EMSA's portal that is based on Liferay. We do not provide any specifications for liferay as no component from SEG will run on Liferay's server.

 © GMV 2020; all rights reserved



In terms of requirements for this environment only the SEG server needs any special attention as the other services are pre-existing at EMSA and the SEG will not play any significant load on them.

HW:

- 2 cpu cores at 2 ghz
- 4GB of RAM
- 50GB HDD

SW:

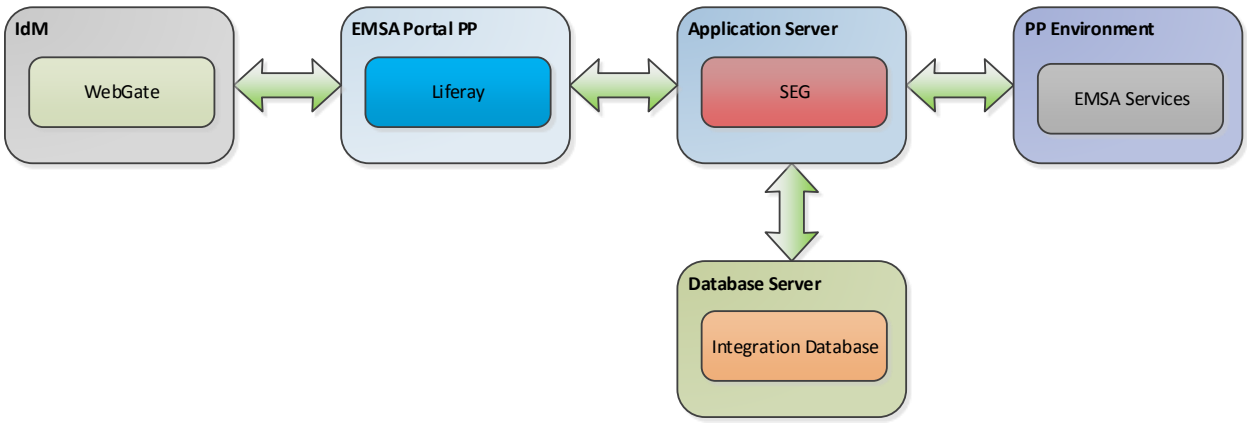
- Red Hat Linux (same version as production)
- Weblogic 12c (12.1.3)
- JDK 1.8

Database Server:

- 2 cpu cores at 2 ghz
- 4GB of RAM
- 50GB HDD

4.1.2.3. Pre Production Environment

On the preproduction environment non functional tests will be performed, and it will be validated if the application behaves correctly on an environment similar to production



In terms of requirements for this environment only the SEG server needs any special attention as the other services are pre-existing at EMSA and the SEG will not play any significant load on them.

This environment should be as close as possible to the production, possibly reducing the amount of RAM available on the machines as it will have only a percentage of the users expected for production.

On this environment a cluster of 2 similar nodes with the following characteristics:

HW:

 © GMV 2020; all rights reserved

- 4 cpu cores at 2 ghz
- 8GB of RAM
- 100GB HDD

SW:

- Red Hat Linux (same version as production)
- Weblogic 12c (12.1.3)
- JDK 1.8

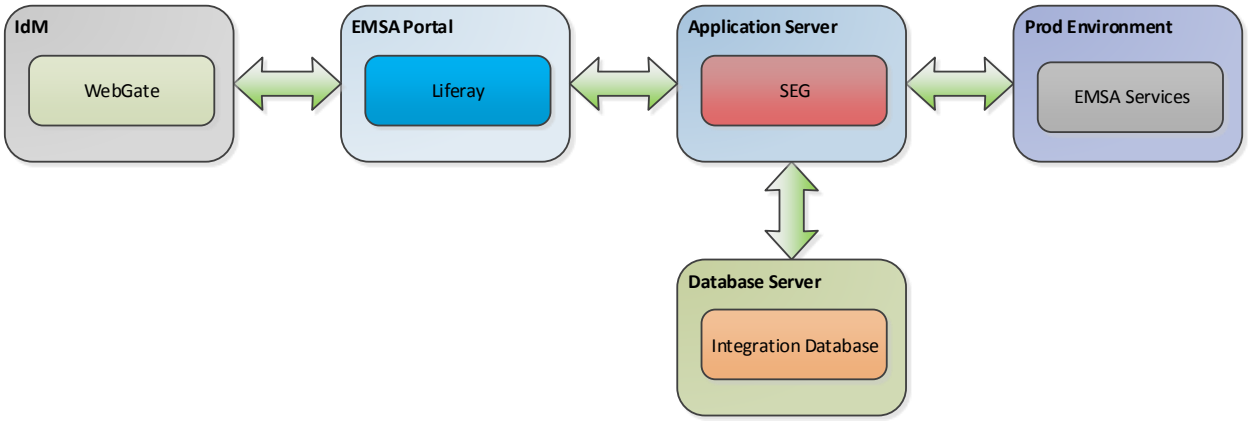
Database Server:

- 4 cpu cores at 2 ghz
- 4GB of RAM
- 100GB HDD

The SEG is not very demanding on the database as it will be used for storing only user preferences and system configurations the load will be low.

4.1.2.4. Production Environment

The production environment is the final environment that the users will access, on this environment the required SLA needs to be maintained.



In terms of requirements for this environment only the SEG server needs any special attention as the other services are pre-existing at EMSA and the SEG will not play any significant load on them.

On this environment a cluster of 2 similar nodes with the following characteristics:

HW:

- 4 cpu cores at 2 ghz
- 32 GB of RAM
- 100GB HDD

SW:

- Red Hat Linux (same version as production)
- Weblogic 12c (12.1.3)
- JDK 1.8

Database Server:

- 4 cpu cores at 2 ghz
- 8 GB of RAM
- 100GB HDD

The SEG is not very demanding on the database as it will be used for storing only user preferences and system configurations the load will be low.

The SEG is designed to be highly scalable with minimal configuration, new nodes can be added or removed, they however need to share the database.

 © GMV 2020; all rights reserved

5. SOFTWARE ARCHITECTURE

On this chapter the several software design views are presented.

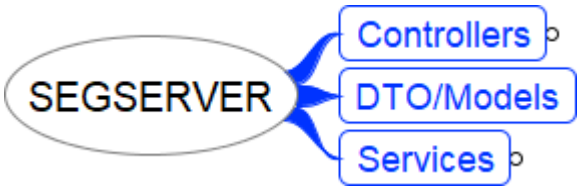
5.1. LOGICAL VIEW

The logical view of SEG will follow the EMSA recommendation for the Applications land scape:

- Client Tier: Rich Web application supported by advanced JavaScript libraries such as AngularJS and Open Layers.
- Web tier: This tier will include the following component to deliver the SEG Rich GUI based on Web browsers.
  - LifeRay portal
  - REST Web services
- Business Tier: SEG components that provide the business logic for an application. On the SEG the business logic will be very simplistic as it typically merges data from different sources, thus the business layer is mainly composed of the Data Transfer Objects used to move data between layers.

The following diagram shows how the components that compose the SEG are linked to this multi-tiered approach:

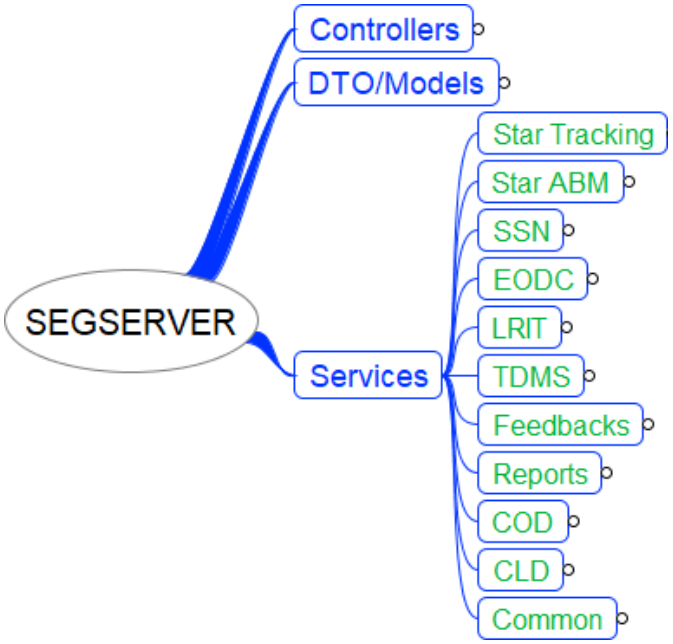
For simplification purposes we will decompose the tiered architecture abve in the following diagram:



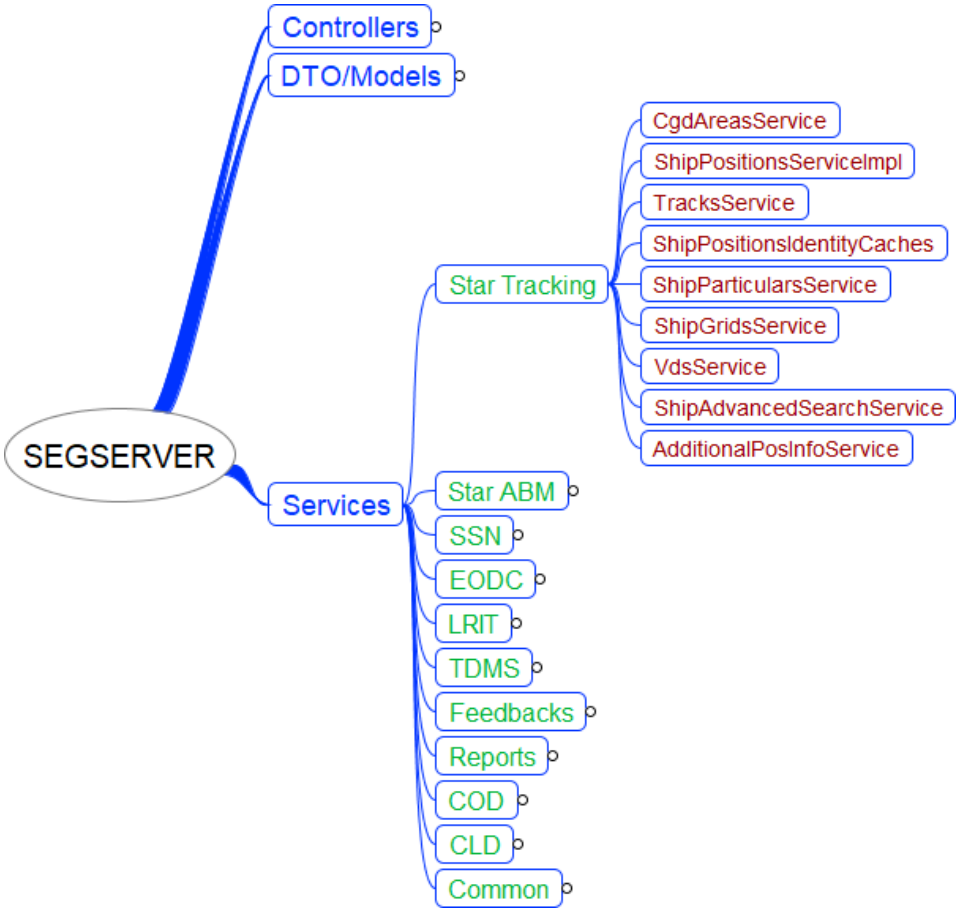
 © GMV 2020; all rights reserved

5.1.1. LOGICAL VIEW - SERVICES

The following diagrams depict the various Services that have been developed in the Web Tier:

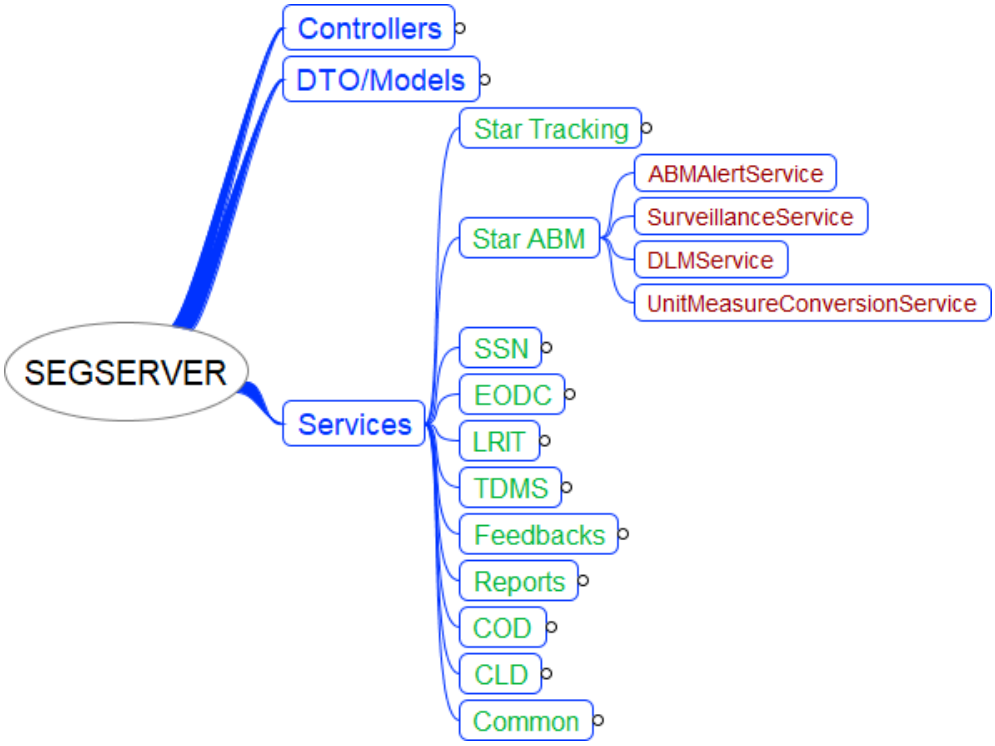


5.1.1.1. Logical View – Services Star Tracking

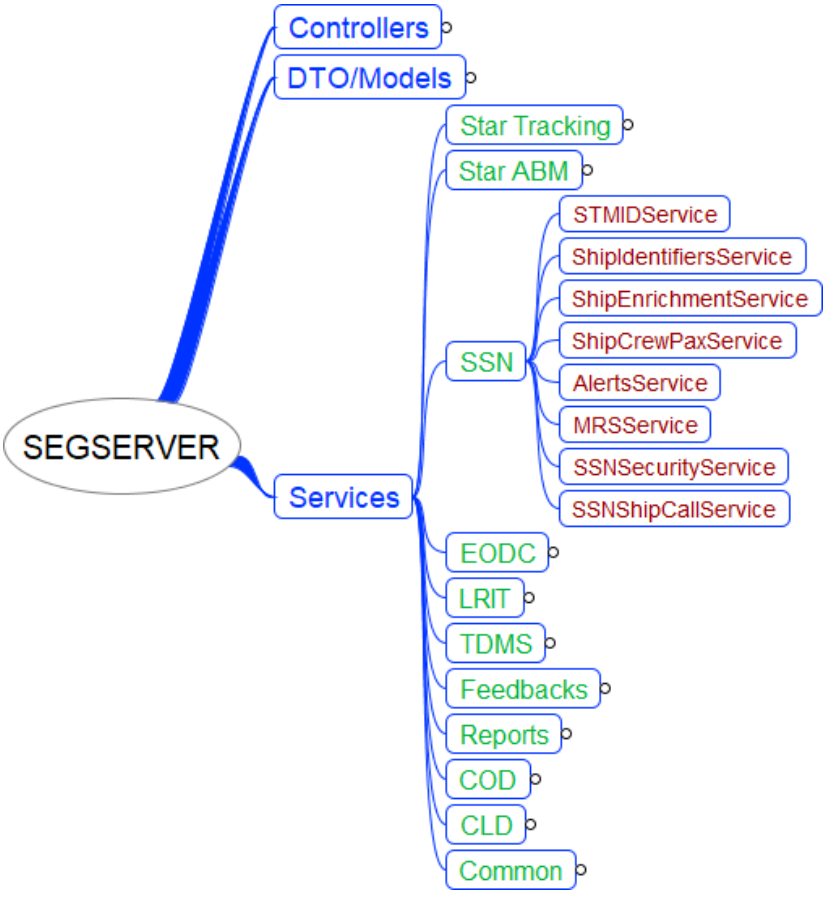


 © GMV 2020; all rights reserved

5.1.1.2. Logical View – Services Star ABM

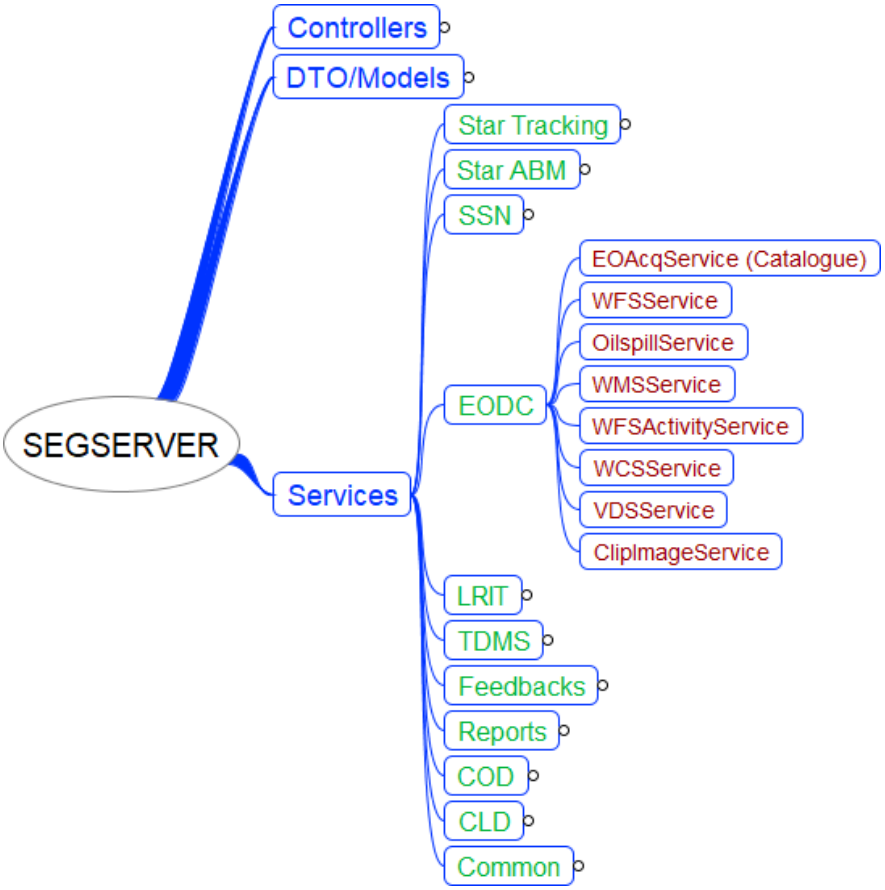


5.1.1.3. Logical View – Services SSN

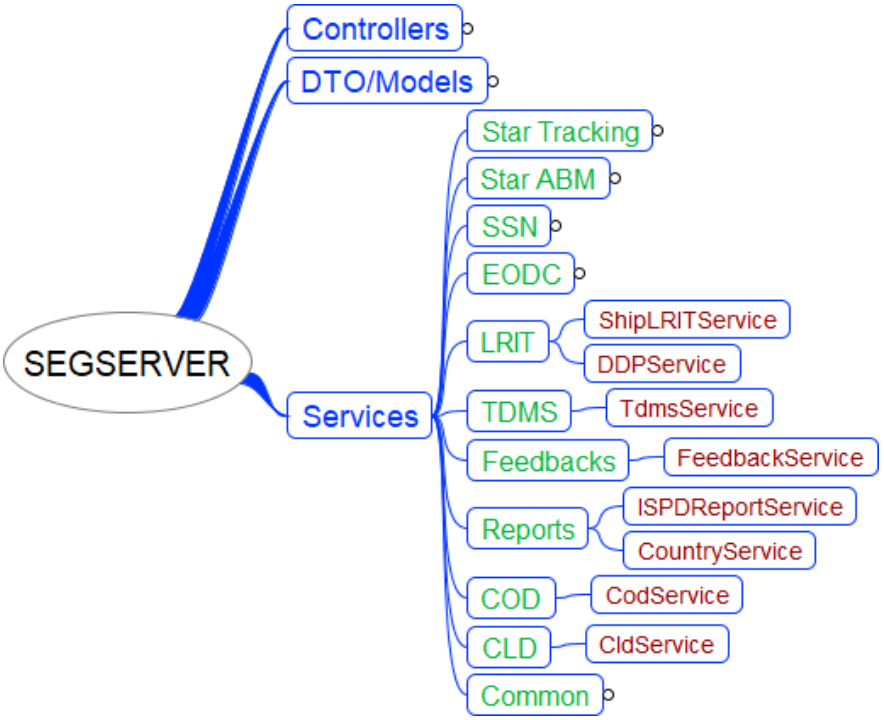


 © GMV 2020; all rights reserved

5.1.1.4. Logical View – Services EODC

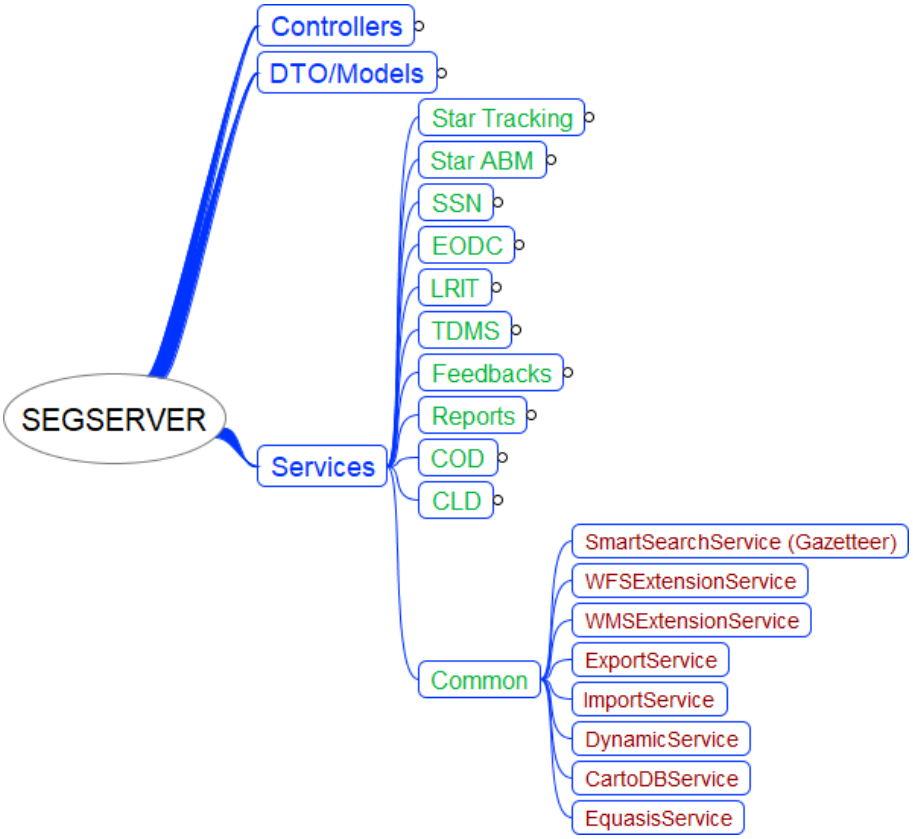


5.1.1.5. Logical View – Services LRIT, TDMS, Feedbacks, Reports, COD and CLD



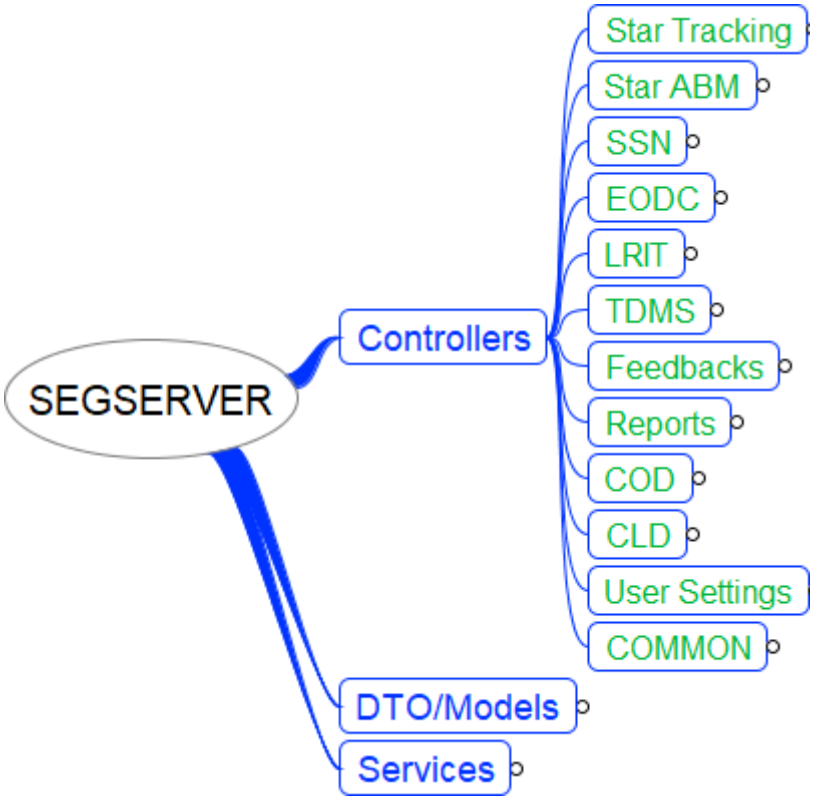
 © GMV 2020; all rights reserved

5.1.1.6. Logical View – Services COMMON



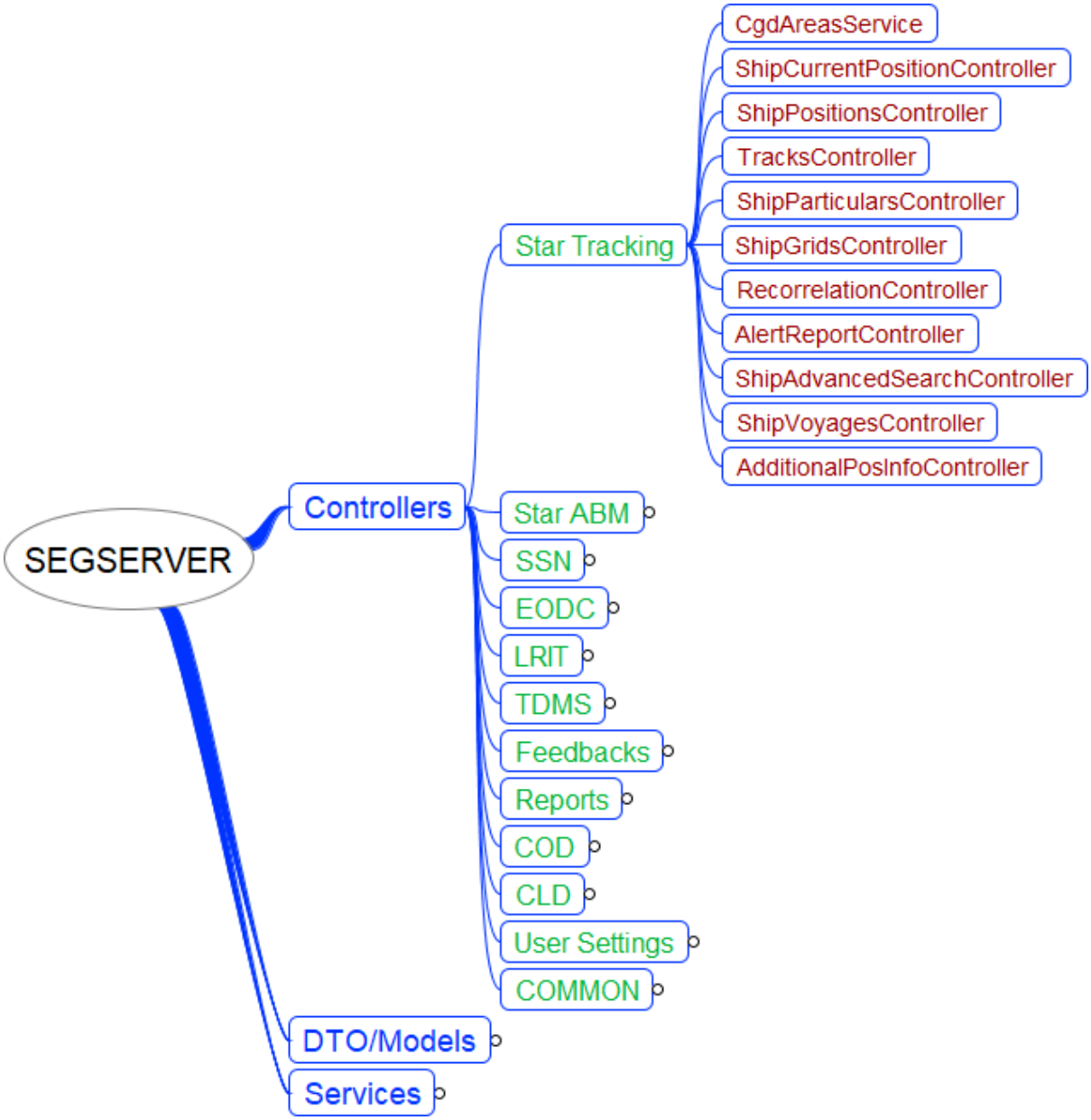
5.1.2. LOGICAL VIEW - CONTROLLERS

The following diagrams depict the various Controllers that have been developed in the Web Tier:



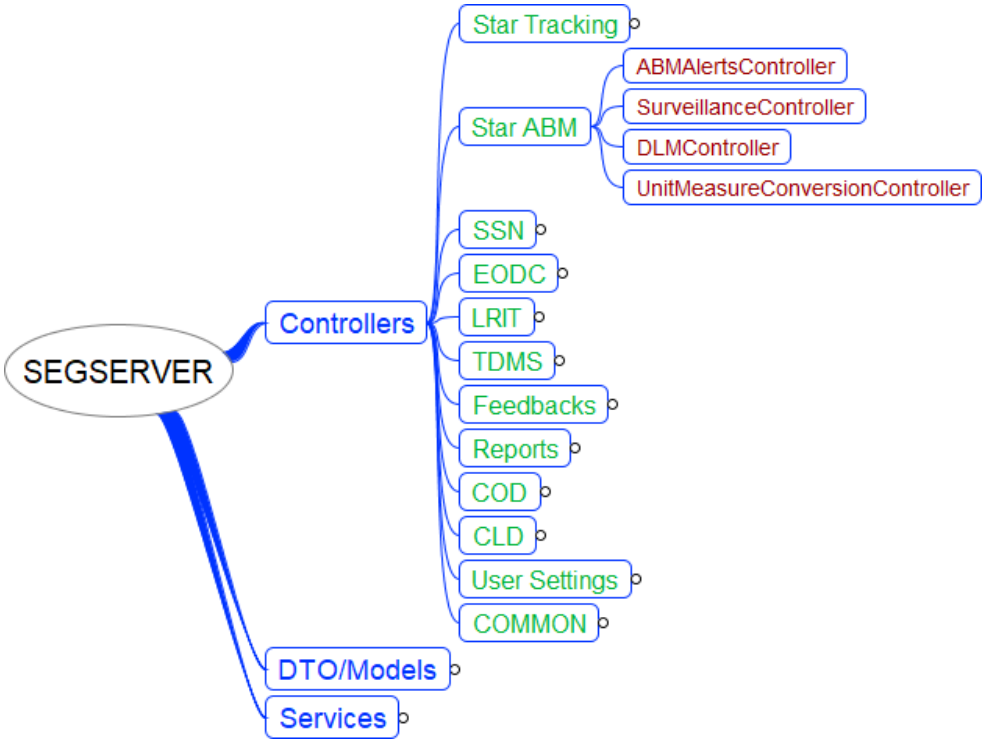
 © GMV 2020; all rights reserved

5.1.2.1. Logical View – Controllers Star Tracking

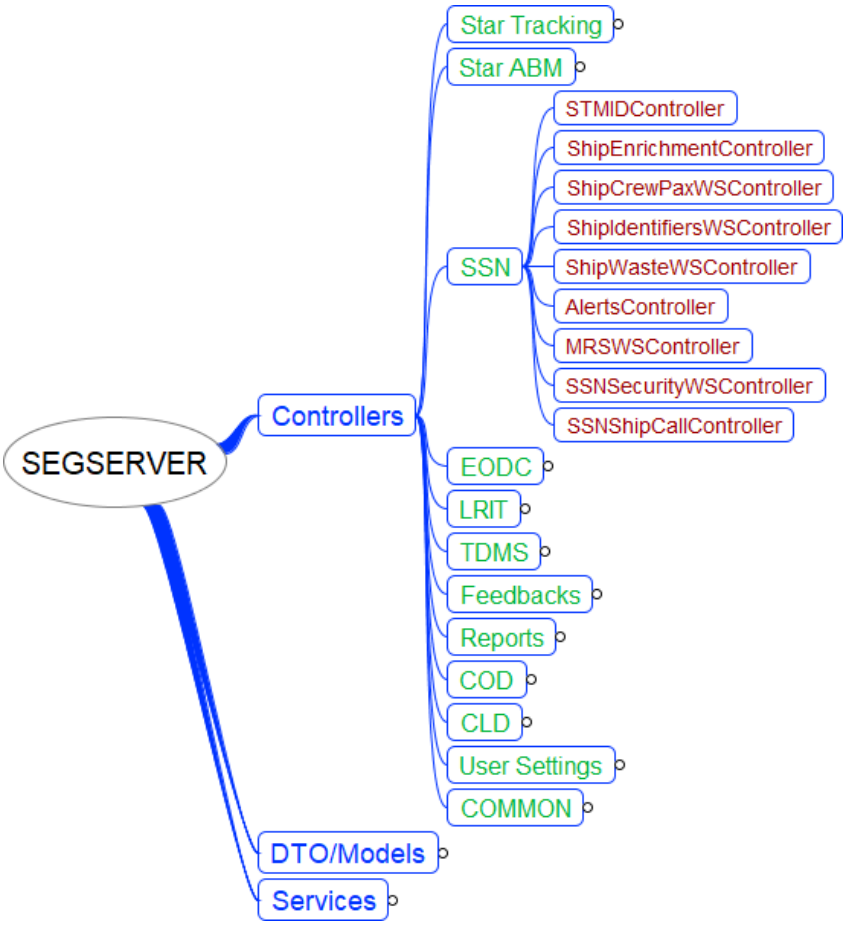


 © GMV 2020; all rights reserved

5.1.2.2. Logical View – Controllers Star ABM

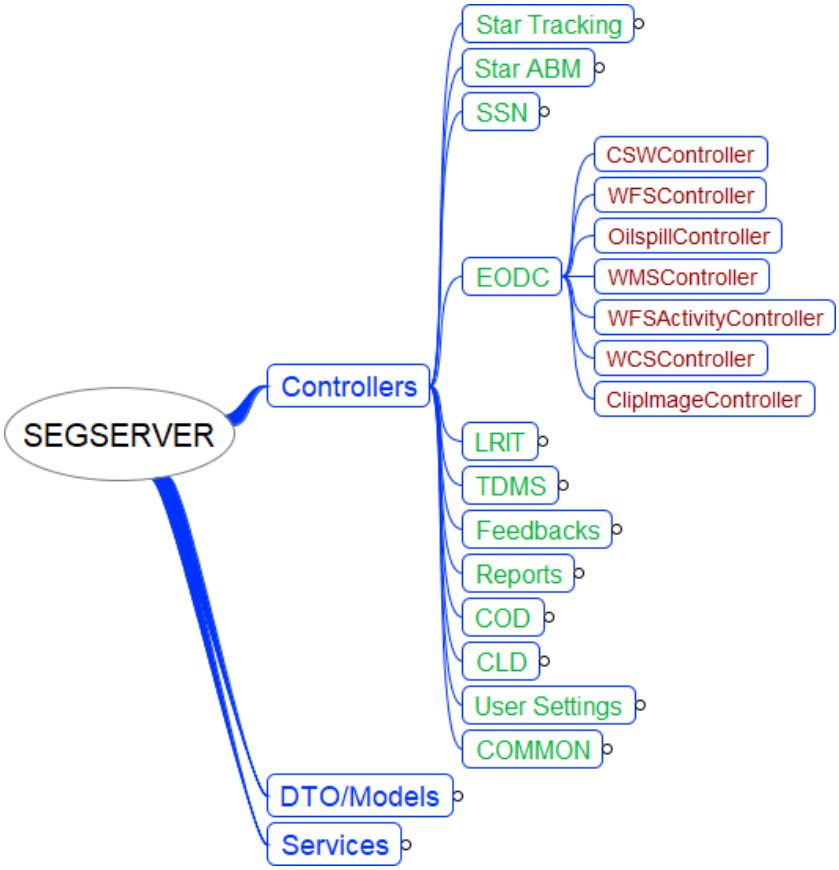


5.1.2.3. Logical View – Controllers SSN

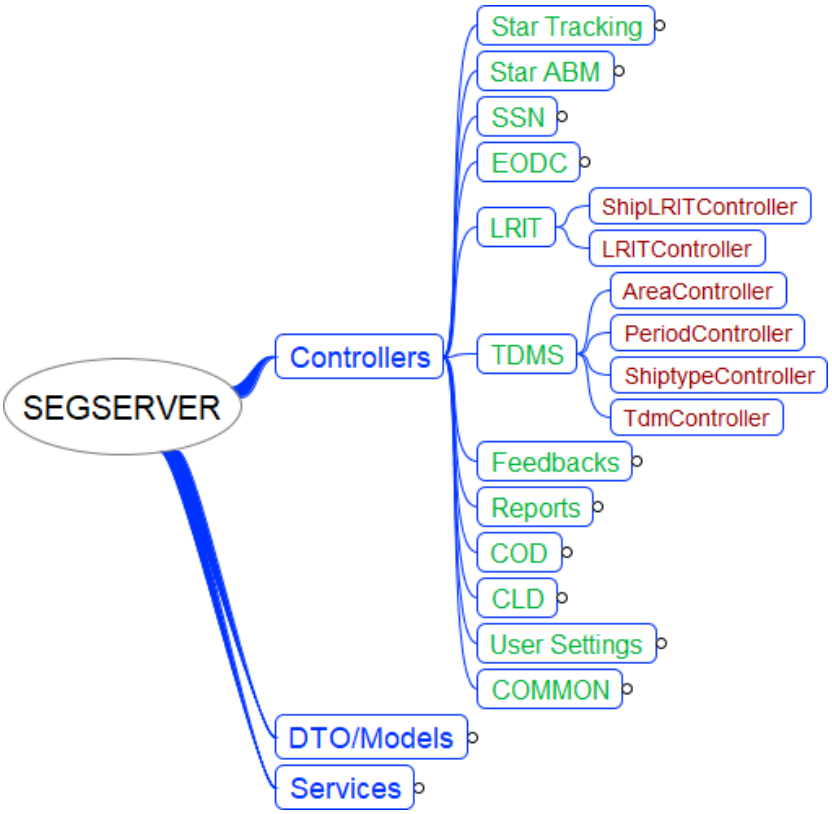


 © GMV 2020; all rights reserved

5.1.2.4. Logical View – Controllers EODC

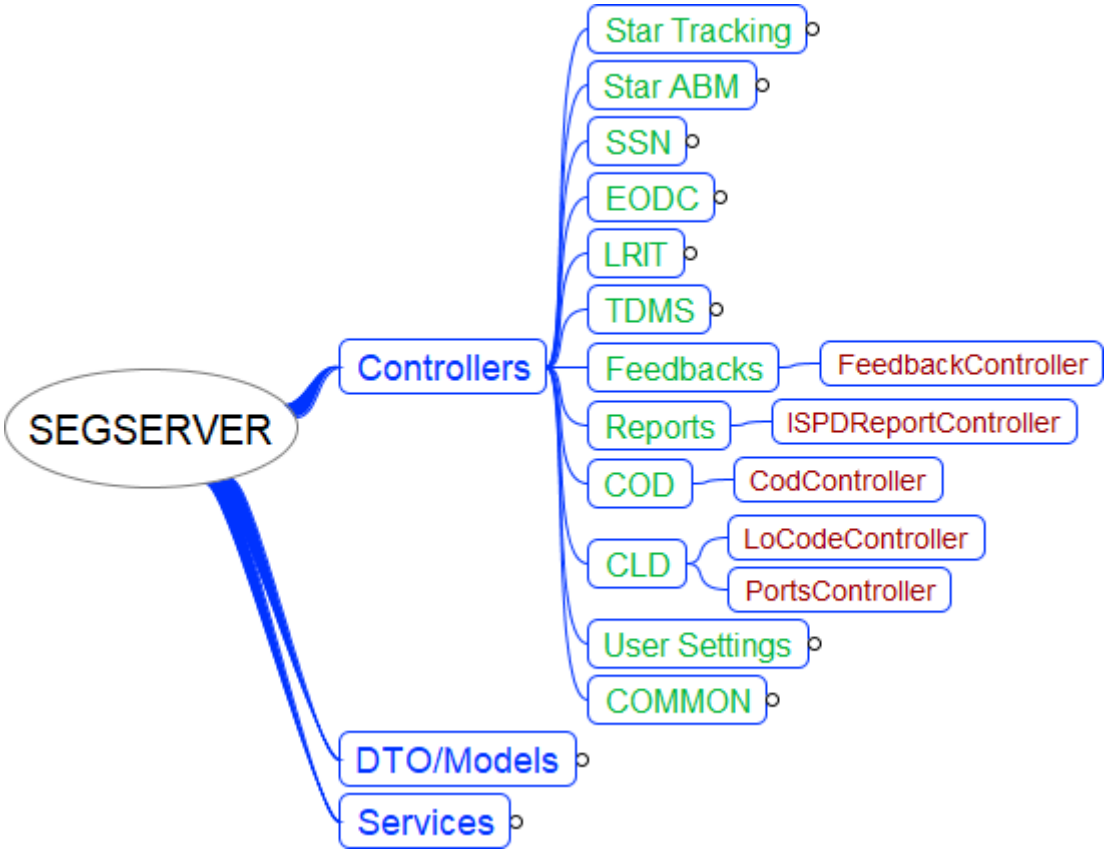


5.1.2.5. Logical View – Controllers LRIT, TDMS

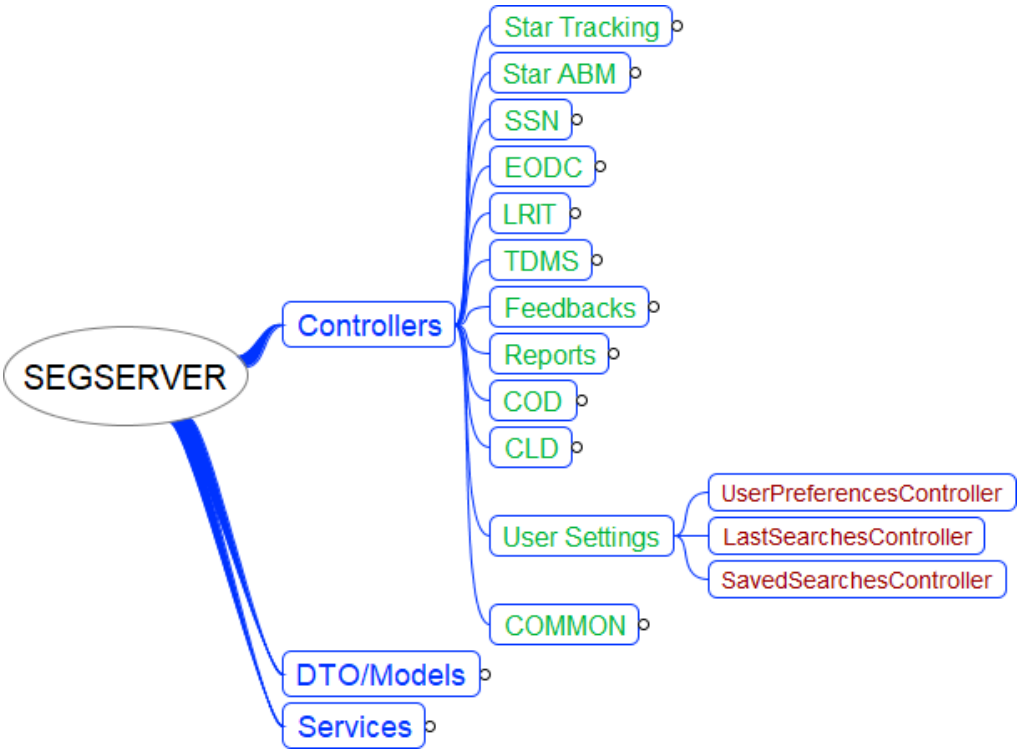


 © GMV 2020; all rights reserved

5.1.2.6. Logical View – Controllers Feedbacks, Reports, COD and CLD

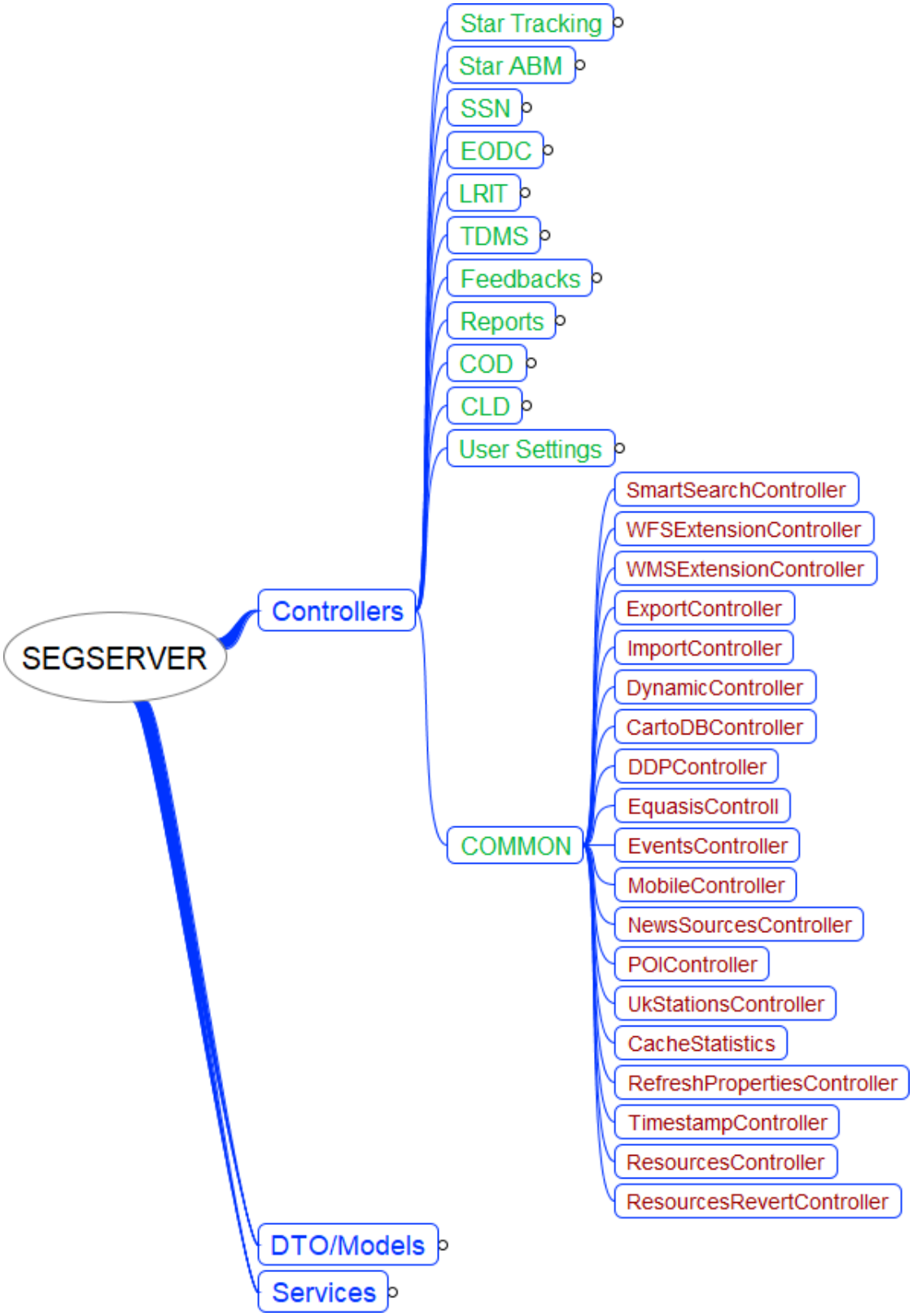


5.1.2.7. Logical View – Controllers User Settings



 © GMV 2020; all rights reserved

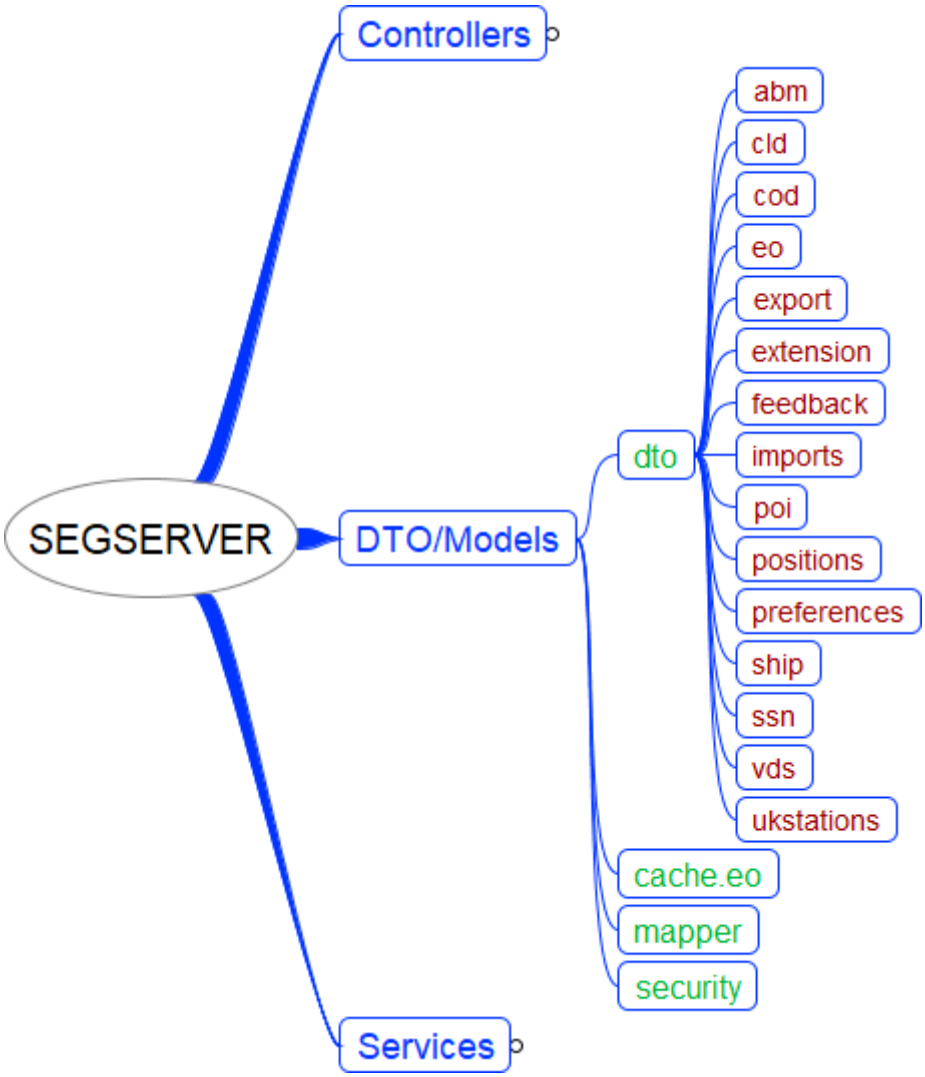
5.1.2.8. Logical View – Controllers COMMON



 © GMV 2020; all rights reserved

5.1.3. BUSINESS VIEW – DTO/MODELS

The following diagram depict the various DTOs/Models that have been developed in the Business Tier:



 © GMV 2020; all rights reserved

5.1.4. CACHING

The figure below shows the flowchart for all the cached requests for the SEG.

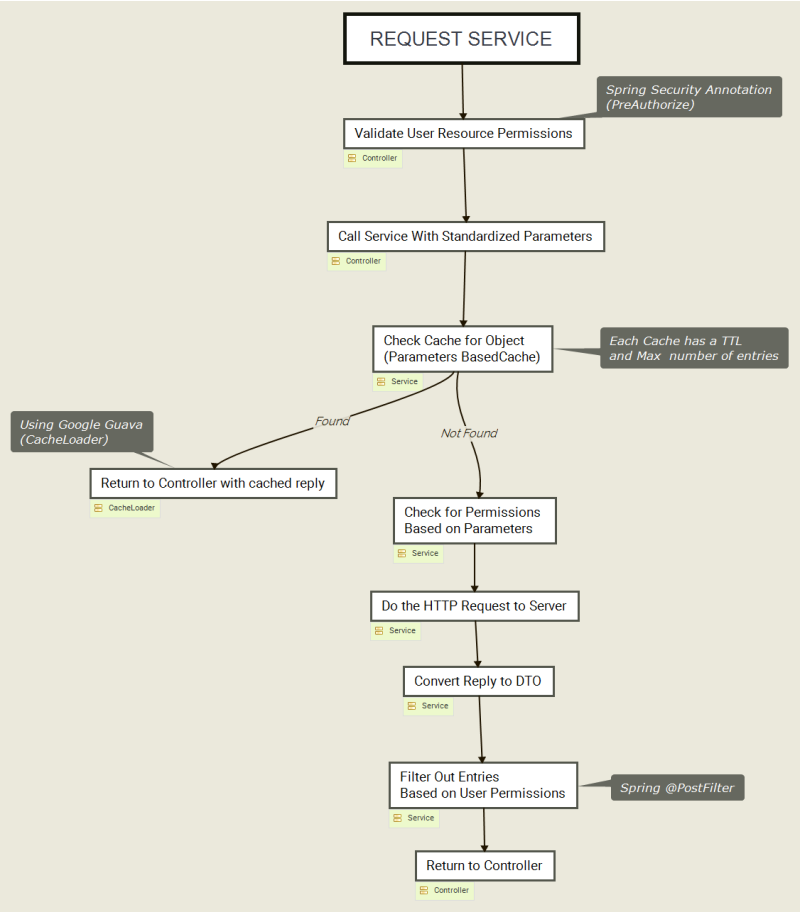


Figure 5-1: Service Catalogue Request flow

The SEG will have a simple approach to caching, where all requests that the reply is not in binary form will be cached based on the request parameters. To achieve a minimal session size the cache will not be replicated between nodes thus the system will require that the load balancer performs stick sessions or equivalent.

Additionally each cache can be configured for time to live, size and invalidated separately.

5.1.5. POSITION ENRICHMENT CACHING

The positions enrichment is based on the SEGs cache, on a periodic timer (currently set for 5 minutes) the request is performed for all vessels for the following services:

- Voyage Information
- Incident Informaion
- MRS Voyages
- Banned Ships
- Detained Ships
- Prevention of Operation/Exemptions
- SHT

This information is then cached. When a positions request is performed it is then enriched with information available from those services.

 © GMV 2020; all rights reserved

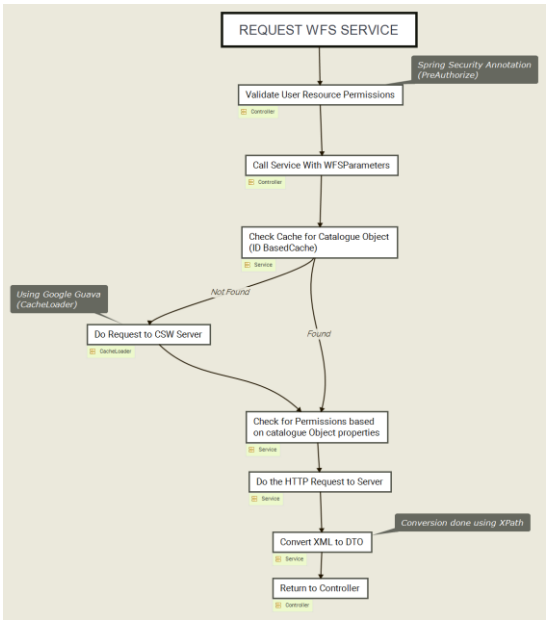
5.1.6. EO-SPECIFIC CACHING

The EO Caching will be disabled for catalogue requests as the Cache on the SEG is designed for simplicity and the current requirements would need a much more complex scenario which is out of scope for the seg as it would replicate the logic on EODC.

However due to operations validation for products the SEG will still cache some information from EODC. Below are the main flowcharts to describe the EO search and retrieval operations:

Figure 5-2: Service Catalogue Request flow

On the figure above it is shown the main steps for the request to get the catalogue information. The SEG will compose the original catalogue information, by adding an additional request to the WCS, providing already the footprints and metadata relative to them. When the list is returned to the user, it is then filtered based on user permissions, by removing the operations to which the user does not have access to.



 © GMV 2020; all rights reserved

Figure 5-3: WFS Request flow

For the WFS Request the system will relay the request performed by the browser to the backend system, however using catalogue information, the operations and additional information relating to the requested products is added to filter the request based on user permissions and product details.

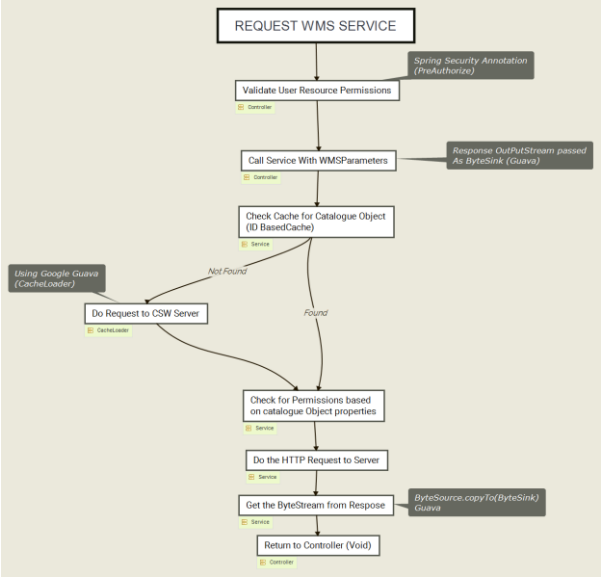


Figure 5-4: WMS Request flow

The WMS is very similar to the WFS request flow, with the exception it handles binary data, and the request is enriched with catalogue information and then filtered based on these parameters.

All information regarding the catalogue is cached on the first request (for the footprints) and that cache is used, if no cached information is available, then a catalogue request will be made for that specific product.

5.2. MANAGEMENT VIEW

The management view deals with the system management and administration. The objective is to identify the system areas that require administration, so that appropriate tools are designed and provided to assist with the system administration.

The following system areas require Administration:

- Application management;
- Management of users, user profiles, and access privileges;
- Management of commercial software modules.

The following figure presents the management view of EUDEMPORT Demonstration System.

 © GMV 2020; all rights reserved

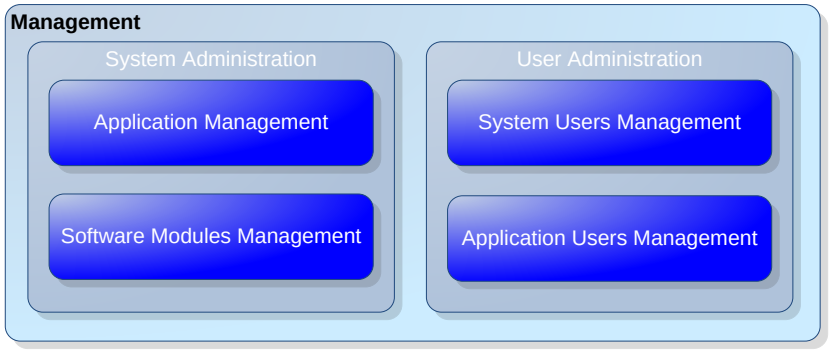


Figure 5-5: SEG Management View Diagram

The management view presents two main components:

- System Administration;
- User Administration.

The System Administration contains two modules:

- Application Management – Deals with the SEG application itself in order to execute management operations like enable/disable specific components (e.g. Interfaces with external systems), monitoring application execution. Status and performance (e.g. Errors and Warnings) and manage the technical configuration parameters.
- Software Modules Management – The SEG application will use and include commercial software that will need to be managed and maintained. The following commercial software will be used:
  - Oracle Database;
  - Weblogic Server.

The User Administration on the SEG is performed using external systems, the Oracle IdM and CARD.

5.3. SECURITY VIEW

SEG System security will be compliant with OWASP (Open Web Application Security Project) Application Security Standards, at least level '2A'. The Open Web Application Security Project (OWASP) is a worldwide free and open community focused on improving the security of application software.

Figure 5-6: OWASP

The Open Web Application Security Project (OWASP) is an open community dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted. All of the OWASP tools, documents, forums, and chapters are free and open to anyone interested in

 © GMV 2020; all rights reserved

improving application security. We advocate approaching application security as a people, process and technology problem because the most effective approaches to application security includes improvements in all of these areas. We can be found at www.owasp.org.

OWASP is a new kind of organization. Our freedom from commercial pressures allows us to provide unbiased, practical, cost-effective information about application security. OWASP is not affiliated with any technology company, although we support the informed use of commercial security technology. Similar to many open-source software projects, OWASP produces many types of materials in a collaborative, open way. The OWASP Foundation is a not-for-profit entity that ensures the project's long-term success.

OWASP Principles

Free & Open
Governed by rough consensus & running code
Abide by a code of ethics
Not-for-profit
Not driven by commercial interests
Risk based approach

The system security view refers to the security considerations and practices for ensuring the data and systems' privacy, integrity and availability.

Security concepts are applied in several layers:

- Physical Security;
- Operating System Security;
- Network Security;
- Transport Security;
- Application Security;

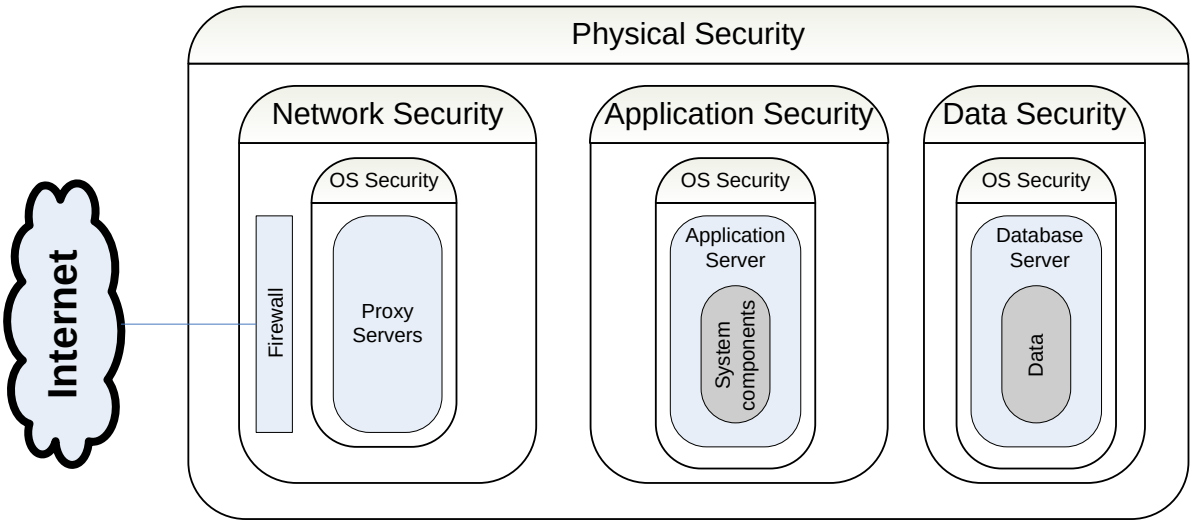


Figure 5-7: **Security View**

5.3.1. PHYSICAL SECURITY

Physical security is the first layer that protects all systems, as such it plays a major role in the security process. It is the physical security that prevents unauthorized users to access the premises of the system. By preventing unauthorized users to access the site where the EUDEMPORT Demonstration System is hosted increases the difficulty in obtaining sensitive data such as passwords and prevents intentional hardware destruction.

 © GMV 2020; all rights reserved

Physical security requirements entail the set-up of rules for physical access to premises where the system is hosted.

5.3.2. OPERATING SYSTEM SECURITY

The operating system security is the second layer after the physical security. A strict password policy will be implemented and unused accounts deleted. System administrators will keep the system updated with the latest security patches and will manage the system following strict security guidelines. The installation of new software releases and patches will be preceded by testing in a development or test environment.

5.3.3. NETWORK SECURITY

The third layer of security on a system is the network security. The network security refers to the use of hardware equipment that ensures the controlled access to the system resources and applications.

For EUDEMPORT Demonstration System the following approach is used, based on the current system configuration at the ports:

- Usage of two firewalls creating a Demilitarized Zone (DMZ)
  - The external firewall filter traffic coming from external users to access the system resources
  - The internal firewall filters traffic coming from services available to the public network and internal users

Additional network security applications such as Intrusion Detection Systems, Virus Detection, and Network Package Sniffers should also be part of the technical environment where EUDEMPORT Demonstration System will be hosted.

The use of an HIDS (Host-based Intrusion Detection System) is encouraged at server level.

5.3.4. TRANSPORT SECURITY

In terms of communication security, the use of Transport Layer Security (TLS) protocol is proposed to access the system from the Internet. TLS provides security and encryption for the data transmitted over the network, making it unreadable without the use of other mechanisms.

HTTPS (HTTP over TLS) will be supported.

5.3.5. APPLICATION SECURITY

Application security is the top-layer for securing user access to applications, authenticating them and providing means of identifying proof-of-origin for messages, transactions, etc.

Every user that is logged into the system is automatically assigned a user profile that is correlated to his/her specific role. Authenticated users are authorized to have access to specific information elements, functions, and reports.

5.3.6. DATA SECURITY

In terms of data security, no direct user access is allowed to the database. All data-related information is provided to the users through the application environment, HTML pages, Mobile application and generated reports. Historical information will be kept for all relevant data blocks. There are two kinds of logs, application logs and functional logs. Functional logs can be accessed through database tools and used to identify specific application state through the data. Application logs can be accessed using a text editor and to fix some application behaviour identified on log data.

The data exchange between the SEG System and other external systems will be performed automatically through internal procedures using EMSA's internal communications links.

5.4. IMPLEMENTATION VIEW

On this development view we present the generic software packages developed. They conform to the Logival View described in Section 5.1 with Figure 5-8 describing some of the logical components in the mentioned section. This reduction was done as depicting all the developed components would render the figure unreadable.

 © GMV 2020; all rights reserved

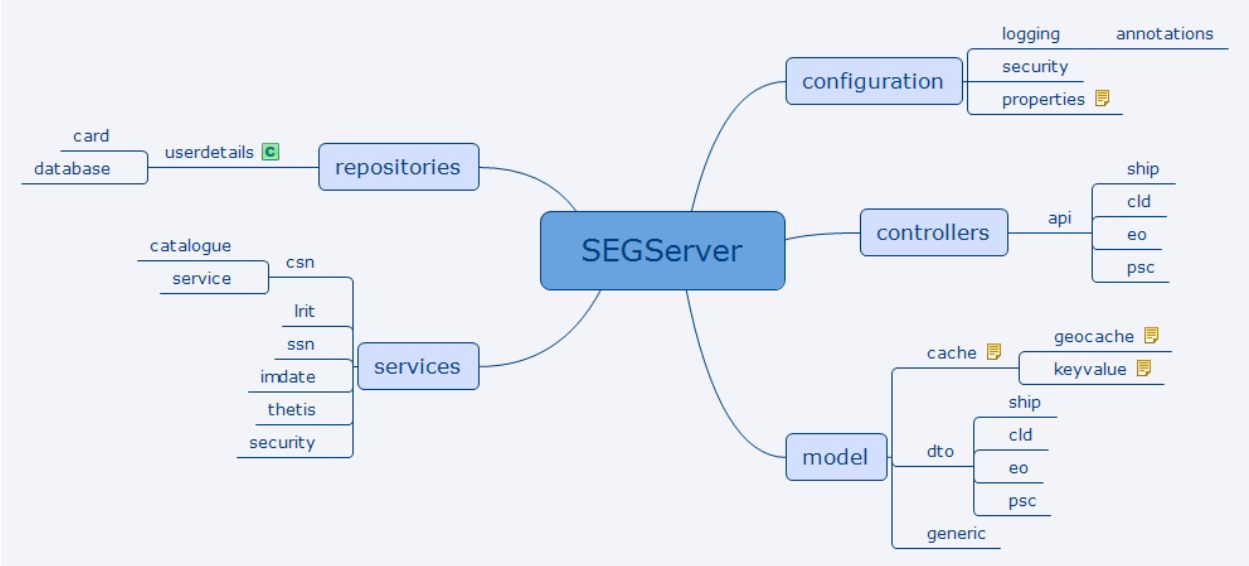


Figure 5-8: **Implementation View**

As shown on the figure above the application is divided in 5 functional packages:

- Configuration
- Controllers
- Model
- Services
- Repositories

Looking into this division and mapping each component into the MVC paradigm, the following split is devised:

- **View**: The view on the SEG is executed fully on the browser via the AngularJS framework and is not represented on the server side components
- **Controller**: The controllers on the SEG are all under the controllers packages, keeping in line with the "thin controllers" the controllers will just map the input/output parameters to the domain objects and select the appropriate service call to execute.
- **Model**: The model on the SEG is composed by the "model", "services" and "repository" packages.
    - **Model**: The model package contains the data transfer objects to be passed between the controllers and services as well as the caches to be used on the project. The key-value cache is a standard cache that can be used to avoid calling the backend services, additionally a geo-referenced cache is added so that a query that requests a sub-area or a subset of data from a previous query, within the cache validity can be answered from the cached values.
    - **Service**: Services on the SEG represent a Business Service Façade, also following the Domain Driven Design definition of "an operation offered as an interface that stands alone in the model, with no encapsulated state". Services thus mark the facades for the OSB and the external services consumed by the SEG.
    - **Repository**: The repositories represent the data stored on the SEG database, they are responsible to fetch and persist data so it is available when the SEG requires.

Outside the MVC model we still have the "**configuration**" these components are the ones responsible to configure the system as a whole as well as marking the auto wiring dependencies, aspects point cuts and other start-up parameters needed for the system to initiate operations.

 © GMV 2020; all rights reserved

6. EXTENSIBILITY

6.1. WMS PROXIED SERVICES

SEG will allow the user to add/remove or edit the existing WMS Service.

For that the SEG will provide a user interface that will allow to add the following parameters for each WMS service:

- Identifier: Uniquely identifies this endpoint
- ResourcePermission: The resource string to be used to check for access permissions
- EndPoint: the WMS endpoint to be used

These new WMS services will be accessible on the SEG via the URI: /api/extenstions/wms/<identifier>

6.2. USER INTERFACE ELEMENTS MANAGEMENT

The SEG will allow the user to dynamically update the user interface elements during runtime. To achieve this all web resources are stored on the database as this will allow automatic distribution between cluster nodes.

The user will access the "/resources" endpoint from this the path following this endpoint will be the full path for the desired resource this will return the latest version of that resource. Additionally the user will have access to the following functionalities:

- List the current resources and respective version;
- Upload a new Version of a specific resource;
- Revert a resource to a previous version;
- Fetch a specific version of a resource.

When a new deployment is done the bundled version of a specific resource is checked against the database version, if higher the database is updated.

 © GMV 2020; all rights reserved

7. DATABASE

For a better and detailed understanding of the overall solution, a detailed view of the data model is provided. This chapter includes the listing of the data model elements (Entity-Relationship Diagram) and a description of the data entities identified.

7.1. DATABASE DEFINITION

This chapter provides a brief description of the technologies used to implement the database system.

7.1.1. ORACLE

The SEG is designed and tested to work using Oracle Exadata.

7.2. DATA MODEL

The SEG Data Model is extremely simple, as only temporary and configuration information is stored on its database.

The tables are described below:

- ACTIVITY_LOG: Contains the user activity
- ALLOWED_PARAMS: Contains the allowed parameters to be used for the dynamic endpoints configuration
- CLIP_IMAGE: Contains the oil spill clip images from EOCD
- CONFIGURATION: Contains the configurations necessary for the SEG
- DATA_DISTRIBUTION_PLAN: Contains information gathered from the LRIT data distribution plan
- DYNAMIC_ENDPOINT: Contains information on exposed endpoints that are simple routing from other EMSA systems
- EVENTS: Stores information on events (fisheries for now)
- FILE_METADATA: Contains metadata on files uploaded to Dropbox
- LAST_SEARCHES: Contains information on the latest searches performed by the users
- NEWS_SOURCES: Contains the configured news sources
- POI: Stores the user POI
- SAVED_SEARCHES: Contains the searches saved by the users
- schema_version: Used to monitor the version of the schema used on automated updates
- SEG_SECURED: Contains the security configurations for the SEG including the expressions to validate if a user has permission to access a resource and the output filtering expression.
- SEG_SECURED_GROUPS_PERMISSION: Associates operations and roles to a resource
- SEG_SECURED_OPERATIONS: Associates IMDaTE Operations to EODC Operations
- SNOOZED_ALERTS: Alerts that have been snoozed by the user
- STMID_DUTIES: Static List of loaded SMID Duties
- TBL_ADDRS: Contains the nodes of this cluster used by Apache Ignite to discover nodes
- UK_STATIONS: Static List of UK AIS Stations
- USER_PERMISSIONS: UNUSED currently, was used on SEG 0.3 to implement the initial user access rignts
- USER_PREFERENCES: Stores the user preferences
- WASTE_CODE_DESCRIPTION: Static list with a mapping to decode the Waste Codes
- WMS_ENDPOINT: Configuration of WMS services proxied by the SEG
- FEEDBACK – Table which holds the data of the feedback of spills.
- FEEDBACK_ATTACHMENT – Information of the attachment files added with the feedback.
- FEEDBACK_TYPE – Holds the feedback types information.
- POLLUTION_TYPE – Holds the pollution types information.
- OTHER_SPILLS – Holds the information regarding connections between spills reported in the feedback.
- POSSIBLE_SOURCES – Information about the possible sources for the spill reported in feedback.
- VDS_PACKAGE_RECORRELATION – Holds the information of re-correlated packages and the time which the re-correlation was requested, including the user.
- VDS_STATUS_MESSAGES – The configured messages for information and error for the vds re-correlation.

 © GMV 2020; all rights reserved

8. SOFTWARE QUALITY ASSURANCE

8.1. CODING STANDARDS

According to GMV's internal procedures, all software code must be written and structured according to the applicable Coding Standards. Adherence to the coding standards is done automatically using the appropriate features of the software development IDE. Additionally coding standards will be validated with FindBugs on GMV's continuous integration Static code analysis Server SonarQube.

8.2. TESTING STANDARDS AND PRACTICES

During development a set of unit tests will be created in order to cover the main functionalities of the system. These unit tests will be performed using the TestNG, a testing framework based on JUnit with added functionalities.

To check the code coverage GMV's continuous integration Static code analysis Server SonarQube will run on a daily basis to ensure proper code coverage.

According to GMV' internal procedures, the verification and validation staff is responsible for:

- Tests design, test cases specification and test procedures specifications which shall be documented in the SVD – Test Plan
- Tests execution and test results are described in the SVD document – Test Results

The validation and verification plan will be performed with the help of the tool TestLink. On this tool, all the user requirements to be implemented will be uploaded.

From the user requirements the test cases will be created and uploaded into TestLink, then for each build to be tested, in the case FAT and SAT a test set will be created and associated with each.

At this phase the SVD document can be created with the following information:

- Test environment description
- Test methodology
- List of tests to be performed
- Requirements coverage matrix

When the system is ready to be tested under FAT or SAT, the test plan provided by TestLink and described on the SVD is followed, and the test results updated on the tool (as Failed, Blocked, Not Run and Passed) , which is then used to retrieve the following metrics:

- Failed tests
- User Requirements affected by failed tests

For the user requirements the following logic is used:

A requirement covered by 3 test cases when only 2 of them were run, the requirement has an overall result of not run, if one failed, then the result will be as Failed. In case all 3 test cases are Passed then the requirement is passed. The priorities for the results are as follows: Failed, Blocked, Not Run and Passed.

The failed tests will then be translated into SPRs which are then created as Problems on GMVMine (Redmine) to be handled.

These metrics will then be used to perform the FAT or SAT report.

8.3. SOFTWARE QUALITY ASSURANCE METRICS

The following table lists the software metrics that will be measured

 © GMV 2020; all rights reserved

Phase / Metric	Coding	Integration	FAT	SAT	Operation
Code quality analysis	×	×	×	×	×
Number of Classes	×	×	×	×	×
Number of Statements	×	×	×	×	×
Unit test Coverage	×	×	×	×	×
Unit Test Failures/Errors	×	×	×	×	×
SPR Trend			×	×	×

Table 8-1: SW metrics to be reported

Where:

- Code quality review results: Static Code analysis performed by FindBugs;
  - Issues are split into: Blocker, Critical, Major, Minor and Info
- Number of classes: total classes implemented;
- Number of statements: per method, distinguishing between executable and comment statements;
  - $\sum code\_sentences$ = LOC
  - $\sum Comments\_lines\_in\_the\_code$
- Unit Test Coverage: Percentage of lines covered by Unit Tests
- Unit Test Failures/Errors: Percentage of Unit tests failing, on error.
- SPR Trend: number of SPRs (total, rejected, open and closed) per phase.

 © GMV 2020; all rights reserved

END OF DOCUMENT

 © GMV 2020; all rights reserved