Deliverable D200.7

Front-End component

WP 200

Project Acronym & Number:	FIspace – 604 123
Project Title:	FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics
Funding Scheme:	Collaborative Project - Large-scale Integrated Project (IP)
Latest version of Annex 1:	2013-10-03
Start date of the project:	01.04.2013
Duration:	24
Status:	Draft
Editor:	Said Rahma (ATOS)
Contributors (to the "R" part of the deliverable[1]; ordered by project partner)	**ATOS**: Said Rahma Rodriguez, Pablo Nuño Lara
Document Identifier:	FIspace-D200.7-FIspace_Integrated_Release_V3-Front-End-v0.4.docx
Date:	27.02.2015
Revision:	004
Project website address:	http://www.FIspace.eu

1 Contributors to FIspace code ("P") include ATB, UDE, IBM, ATOS, KOC, TOG, AST, NKUA, UPM and LimeTri; contributing persons are listed at https://bitbucket.org/fispace/profile/members

The FIspace Project

Leveraging on outcomes of two complementary Phase 1 use case projects (FInest & SmartAgriFood), aim of FIspace is to pioneer towards fundamental changes on how collaborative business networks will work in future. FIspace will develop a multi-domain Business Collaboration Space (short: FIspace) that employs FI technologies for enabling seamless collaboration in open, cross-organizational business networks, establish eight working Experimentation Sites in Europe where Pilot Applications are tested in Early Trials for Agri-Food, Transport & Logistics and prepare for industrial uptake by engaging with players & associations from relevant industry sectors and IT industry.

Project Summary

As a use case project in Phase 2 of the FI PPP, FIspace aims at developing and validating novel Future-Internet-enabled solutions to address the pressing challenges arising in collaborative business networks, focussing on use cases from the Agri-Food, Transport and Logistics industries. FIspace will focus on exploiting, incorporating and validating the Generic Enablers provided by the FI PPP Core Platform with the aim of realising an extensible collaboration service for business networks together with a set of innovative test applications that allow for radical improvements in how networked businesses can work in the future. Those solutions will be demonstrated and tested through early trials on experimentation sites across Europe. The project results will be open to the FI PPP program and the general public, and the pro-active engagement of larger user communities and external solution providers will foster innovation and industrial uptake planned for Phase 3 of the FI PPP.

Project Consortium

- DLO; Netherlands
- ATB Bremen; Germany
- IBM; Israel
- KocSistem; Turkey
- Aston University; United Kingdom
- ENoLL; Belgium
- KTBL; Germany
- NKUA; Greece
- Wageningen University; Netherlands
- PlusFresc; Spain
- FloriCode; Netherlands
- Kverneland; Netherlands
- North Sea Container Line; Norway
- LimeTri; Netherlands
- BO-MO; Slovenia
- MOBICS; Greece
- Fraunhofer IML; Germany
- Q-ray; Netherlands
- FINCONS; Italy
- Kühne + Nagel; Switzerland
- University Duisburg Essen; Germany
- ATOS; Spain
- The Open Group; United Kingdom
- CentMa; Germany
- iMinds; Belgium
- Marintek; Norway
- University Politecnica Madrid; Spain
- Arcelik; Turkey
- EuroPoolSystem; Germany
- GS1 Germany; Germany
- Mieloo & Alexander; Netherlands
- OPEKEPE; Greece
- Innovators; Greece
- CIT; Spain
- SDZ; Germany
- Snoopmedia; Germany
- EECC; Germany
- CBT; Spain

More Information

Harald Sundmaeker (coordinator)	e-mail:	sundmaeker@atb-bremen.de
Bert Vermeer (deputy coordinator)	e-mail:	bert.vermeer@wur.nl
Project Website	Web link:	http://www.fispace.eu/

Dissemination Level

PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	**X**
RE	Restricted to a group specified by the consortium (including the Com-mission Services)	
CO	Confidential, only for members of the consortium (including the Com-mission Services)	

Change History

Version	**Notes**	**Date**
001	Creation of the document Front-End component	09.12.2014
002	Update of the contents, added new technological environment and functionalities, minor changes related to rephrasing	20.01.2015
003	Internal review process, checking URL links to the FIspace Web online documentation Update of the abbreviation table, update of the references table Final version ready for submission of approved document	06.02.2015
004	Update of the coordinator information in the section "*More In-formation*" Added FIspace development repository and documentation references, formatting improvement Final version ready for submission to EC	27.02.2015
005		
006		

Abbreviations

AAA	Authentication, Authorisation, and Accounting
ACSI	Artifact-Centric Service Interoperation
AdvB	Advisory Board
AJAX	Asynchronous JavaScript + XML
API	Application Programming Interface
App	Software Application
B2B	Business-to-business
B2C	Business-to-Consumer
BCM	Business Collaboration Module in FIspace
BCO	Business Collaboration Objects in FIspace
BE	Business Entities
BPPC	Business Process Participant Configuration
BSS	Business Support Systems
CDR	Charging Detailed Records
CEP	Complex Event Processing
CSB	Cloud Service Bus
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
D	Deliverable
DAO	Data Access Object
DB	Database
DoW	Description of Work
EC	European Commission
EDI	Electronic Data Interchange
EE	Experimentation Environment
e.g.	Exempli gratia = for example
EPA	Event Processing Agent
EPM	Event Processing Module in FIspace
ESB	Enterprise Service Bus
EU	European Union
FIA	Future Internet Assembly
FI-PPP	Future Internet Public Private Partnership
FP7	Framework Programme 7
GA	Grant Agreement
GE	Generic Enabler
GUI	Graphical User Interface
HTML	HyperText Markup Language
IaaS	Infrastructure as a Service
ICT	Information and Communication Technology
IDE	Integrated Development Environment
IDM	Identity Management
i.e.	id est = that is to say
IE	Integration Environment
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
I/O	Input / Output
IoT	Internet of Things
IP	Intellectual Property
IP (protocol)	Internet Protocol
IPR	Intellectual Property Rights
IPsec	Internet Protocol Security
IT	Information Technology
ITU	International Telecommunication Union
ISO	International Standardization Organisation
J2SE	Java 2 Platform, Standard Edition
JDK	Java Development Kit
JDT	Related to Eclipse Java Development Tools
JMX	Java Management Extensions
JRE	Java Runtime Environment
JS	JavaScript
JSON	JavaScript Object Notation
JSP	Java Server Page
JVM	Java Virtual Machine
KPI	Key Performance Indicator
LPA	Logistics Planning Application
M	Month
MTBF	Mean Time Between Failures
MVC	Model–View–Controller
OASIS	Organization for the Advancement of Structured Information Standards
OAuth	Open standard Authentication protocol
OMG	Object Management Group
OSS	Operational Support Systems
P2P	Peer-to-peer
PaaS	Platform as a Service
PDE	Related to Eclipse Java Development Tools
PE	Production Environment
PIA	Product Information App

PIE	Preliminary Integration Environment
PKI	Public Key Infrastructure
PM	Person Month
POM	Project Object Model (used by maven tools)
Proton	IBM Proactive Technology Online
QoS	Quality of Service
RBAC	Role-Based Access Control
RCP	Rich Client Platform
REST	Representational State Transfer
RFC	Request for Comments
RSS	Revenue Sharing System
RTD	Research and Technological Development
SaaS	Software as a Service
SDI	System and Data Integration layer in FIspace
SDK	Software Development Kit
SME	Small and Medium Sized Enterprise
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOA-RM	(OASIS) Reference Model for Service Oriented Architecture
SPT	Security, Privacy and Trust Framework
SSH	Secure Shell
SSL	Secure Sockets Layer
SSO	Single Sign On
ST	Sub-Task
SWT	Standard Widget Toolkit
T	Task
TCP	Transmission Control Protocol
TIC	Tailored Information for Consumers
TLS	Transport Layer Security
TPM	Transport Planning Module
UAA	User Management, Authentication and Authorisation
UI	User Interface
UML	Unified Modeling Language
URI	Universal Resource Identifier
URL	Universal Resource Locator
USDL	Unified Service Description Language
VM	Virtual Machine
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WADL	Web Application Description Language
WLAN	Wireless Local Area Network
WP	Work Package
WS	Web Service
WSDL	Web Services Description Language
XLS/XLSX	Microsoft Excel file Format
XML	eXtensible Markup Language
XSD	XML Schema Definition

Table of Contents

List of Figures

List of Tables

1 Introduction

This document aims at describing the third release (V3) of the FIspace, encompassing the implementations along with usage guidance and technical documentation of each FIspace component.

It reports on the description concerning the **Front-End core component**, the description of the development and implementation of the Front-End core components that is part of the FIspace platform.

The User Front-End serves as the main point of access for users of the platform services and Apps. It includes the following main features:

- **Customizable user dashboards**: To ensure our applications are usable, the front-end strives to provide an environment where they feel comfortable, i.e., provide interaction patterns that understand limitations and offer potential opportunities to the users.
- **Social networking and collaboration features** for business partners.
- **Access from anywhere across multiple devices**.

The User Front-End builds the main access point for users of the FIspace platform. Through the integration of external widgets (e.g., from the store, externally developed Apps or other external providers), the User Front-End facilitates an ‘all you need in one place’ user experience and creates a central access point. To support the diversity of FIspace users and devices the User Front-End will be adaptable to specific needs, tasks and roles. Beyond the adaptation to different devices, the User Front-End also supports the configuration of the user interface. This allows the interface personalization in order to address specific user needs or enable custom brandings for companies. The Front-End also enables users to create relations to business partners to facilitate the communication among them (comparable to modern social networks).

Online documentation for User Front-End: http://dev.fispace.eu/doc/wiki/gui

1.1 Scope

The aim of this document is mainly to describe and detail the **FIspace Front-End core component** at development and implementation level, giving detailed and technical information related to the design and the implementation as well as information about the related technologies and standard taken as a reference to build each component.

Along this development activities and tasks, there is a set of resources, online documentation, tutorial and other external resource that refer to the Generic Enablers that can provide more technical information and user guides for the community and people who want to use the FIspace platform for Business collaboration or developers who want to create and develop business application (Apps developer) for a specific domain of application.

Table 1 shows the links to other online resources related to FIspace project and FI-WARE.

Description	Link
FIspace Business collaboration web site	http://www.fispace.eu/
FIspace Developer Documentation web site	http://dev.fispace.eu/doc/wiki/Home
FIspace Deliverables web site	http://www.fispace.eu/deliverable.html
FIspace Tutorial web site	http://www.fispace.eu/tutorials.html
FIWARE web site	http://www.fi-ppp.eu/projects/fi-ware/
FIWARE Catalogue of the Generic Enablers (GEs)	http://catalogue.fi-ware.org/
FIWARE community web site	http://www.fi-ware.org/community/

Table 1: Other FIspace and FIWARE resources

Table 2 shows the links to the Wirecloud online documentation.

Description	Link
FIWARE - Catalogue - Application Mashup - Wirecloud	http://catalogue.fi-ware.org/enablers/application-mashup-wirecloud
FIWARE - Catalogue - Application Mashup - Wirecloud Documentation	http://catalogue.fi-ware.org/enablers/application-mashup-wirecloud/documentation
FIWARE - Application Mashup - Wirecloud - User and Programmer Guide	https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Application_Mashup_-_Wirecloud_-_User_and_Programmer_Guide
Dashboard - Wirecloud home page	http://conwet.fi.upm.es/wirecloud/
Dashboard - The WireCloud Mashup Platform	http://conwet.fi.upm.es/docs/display/wirecloud/The+WireCloud+Mashup+Platform
Dashboard - Welcome to CoNWeT-Wirecloud Confluence	http://conwet.fi.upm.es/docs/dashboard.action
Dashboard - User Guide	http://conwet.fi.upm.es/docs/display/wirecloud/WireCloud+User%27s+Guide
Dashboard - WireCloud Installation and Administration Guide	http://conwet.fi.upm.es/docs/display/wirecloud/Wire-Cloud+Installation+and+Administration+Guide

Table 2: Wirecloud online documentation

Table 3 shows the links to the WStore online documentation.

Description	Link
FIWARE - Catalogue - Store - WStore	http://catalogue.fi-ware.org/enablers/store-wstore
FIWARE - Catalogue - Store - WStore Documentation	http://catalogue.fi-ware.org/enablers/store-wstore/documentation
FIWARE - Store - W-Store - User and Programmer Guide	https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Store_-_W-Store_-_User_and_Programmer_Guide
FIWARE - Store - W-Store - Store - W-Store - Installation and Administration Guide	https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Store_-_W-Store_-_Installation_and_Administration_Guide

Table 3: Store online documentation

Table 4 shows the external development tools references.

Description	Link
Java Environment, JVM, JRE, JDK (Oracle)	http://www.oracle.com/technetwork/java/javase/downloads/index.html
Eclipse IDE (Integrated Development Environment)	https://www.eclipse.org/ , https://www.eclipse.org/downloads/
Maven	http://maven.apache.org/ , http://maven.apache.org/download.cgi

Table 4: External development tools references

Table 5 shows the FIspace development repository and documentation references based on the bitbucket tools for collaborative development.

Bitbucket is a hosting site for the distributed version control systems (DVCS) Git (http://git-scm.com/) and Mercurial (http://mercurial.selenic.com/). The service offering includes an issue tracker and wiki, as well as integration with a number of popular services such as Basecamp, Flowdock, and Twitter.

Description	Link
Bitbucket FIspace repository home page	https://bitbucket.org/fispace

Bitbucket FIspace core component home page	https://bitbucket.org/fispace/core/wiki/Home
Bitbucket FIspace Roadmap page	https://bitbucket.org/fispace/core/wiki/roadmap

Table 5: Bitbucket collaborative environment for FIspace development

1.2 Intended audience

The main interest groups of this deliverable are the participating teams and the responsible partners of FIspace project involved in the development activities, setup and preparation of the development phase. This document is relevant to the software engineer, programmers and developers who are the persons directly involved in the development, participating effectively on the design and implementation of the FIspace platform and the underlying components and sub-systems who want to know more about some technical information intrinsic to the FIspace platform.

At the technical level this document is relevant to: system architects; information systems designers; system developers and application developers; software engineers; other audiences who provide design services and applications using relevant standards and the recommendations of standards bodies like IETF, ITU, ISO, W3C, etc.

Partners involved in the integration tasks include: system integrators; people to test, validate and evaluate the FIspace platform and associated systems; can be also interested.

1.3 General remark

This document follows the ISO/IEC Directives, Part 2: Rules for the structure and drafting of International Standards w.r.t. the usage of the word “shall”. The word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this specification.

This document describes the corresponding core components involved in the FIspace core platform. It presents the development currently done and the corresponding implementation, the main features developed, as well as the related technologies and environment requirements.

In most of the following sections the structure is organized as:

- **Overview**: provides an overall introduction to the component, a description, of the internal architecture and features among other.
- **Interfaces or Application programming interface (API)**: describes the API accessible for the users or entities of the component (typically applications, but a component may also be used by other components).
- **Information model**: describes or specifies the component from an information perspective describing information objects of the component domain.
- **Interaction model**: describes or specifies main usage component “scenarios” associated with the component/GEs, sequence diagrams.
- **High level composite architecture**: describes or shows the main components constituting the set of components (this perspective is optional, since some component consists of only one main component).

Notice that some components only need to describe some of the item above described.

2 Front-End

2.1 Overview

The User Front-End builds the main access point for users of the FIspace platform. Through the integration of external widgets (e.g., from the store, externally developed Apps or other external providers), the User Front-End facilitates an 'all you need in one place' user experience and creates a central access point. To support the diversity of FIspace users and devices the User Front-End will be adaptable to specific needs, tasks and roles. Beyond the adaptation to different devices, the User Front-End also supports some customization of the user interface. The Front-End also enables users to create relations to business partners to facilitate the communication among them (comparable to modern social networks).

User Front-End has been developed as a web application. This web application can be accessed from several devices using its specific URL. All the information related with the User Front-End is stored in a relational database located in the same machine and environment.

In order to let other applications to access to FIspace Front-End database, a Rest API is provided. This API is being used by another User Front-End approach that has been developed as proof of concept, a Java FX application. This application can be executed as desktop application offering FIspace users another user experience.

Figure 1 shows an overview of User Front-End component.

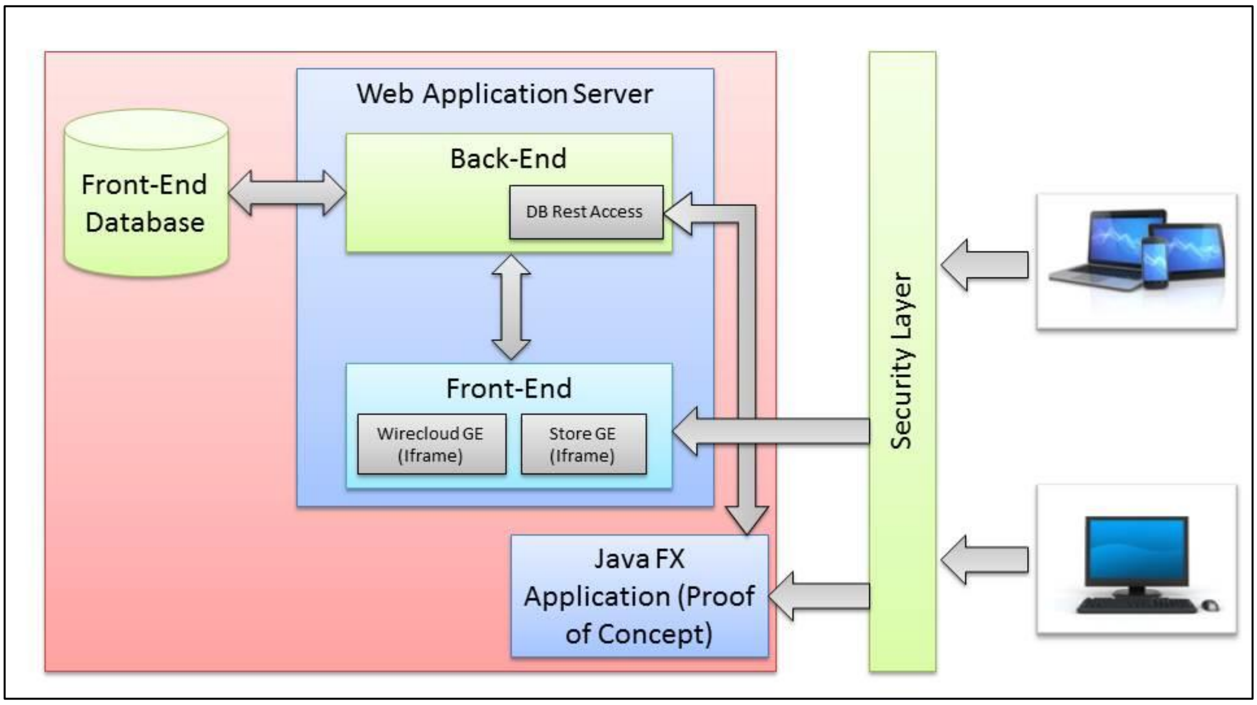


Figure 1: Overview of User Front-End component

2.1.1 Features Provided

The current version of FIspace Front-End provides to the final user the following functionalities.

- **User Profile**: Users can create a Personal Profile. In this profile different personal information can be shown and edited. Also profile picture and cover picture can be uploaded. If you visit another user profile you can add his/her profile as Friend.
- **Company Profile**: Users can create a Company Profile. In this profile different company information can be shown and edited. Also profile picture and cover picture can be uploaded. If you visit a company profile you can follow this company. Who creates the Company Profile is automatically its administrator.
- **Search**: User and Company profiles can be found using search bar. It searches in all the users and companies names the string that you wrote in search field.
- **News**: Home Page where users can post news or information to other FIspace users. Users can post this information as user or as Company (If user is company profile administrator). Each user only sees his friend's news and posts from companies he is following. In each post Edit/Like/Comment/Share actions can be performed.
- **Guided tour**: In welcome page a Tour option that explains each menu option to new users can be started.
- **Chat**: Users can communicate with other FIspace users using chat functionality. Conversations are store in user browser in order to continue a conversation later if browser is closed.
- **Friends/companies suggestions**: In the welcome page Users obtain Friends/Companies suggestions.

  These suggestions are generated with the following criteria:
  - **Friends Suggestion**:
    - If User doesn't have any friend yet he/she gets users from the companies he/she follows.
    - If User has friends he/she gets friends of his/her friends (1st level relationship only).
  - **Companies Suggestion**: Most popular ones. All companies ordered by number of followers in descending order.
- **Help**: All options in FIspace Front End are described in this section in order to help new users to understand the different functionalities.
- **Theme Customization**: Users can select 2 different themes for FIspace Front End (Default Theme or Amelia Theme) to give them the possibility to customize some visual aspects as colours or fonts.
- **Private Profile**: Users can make their profile private. In case he/she actives this option, his/her profile is not going to appear in search results.
- **Email new News**: Users can receive in their email what their friends post in FIspace. An automatic mail is sent containing the new post.
- **User Analytics**: Users can see how many users has visited their profile and who of these visits are already friends or not. Also there is a measurement of profile completed.
- **Internal Messages**: Users can send internal messages to other users. Usual messaging actions are provided (Reply/Forward/Resend/Save as Draft/Mark as Favourite).

- **Different Languages available**: Users can select different languages for FIspace Front-end options and Menus.
- **Advanced Search**: Users can search other users or companies using different filters.
- **More themes**: FIspace Frontend offers more than Default or Amelia theme. A complete list of themes will be available. Also users can download a .css template and modify it in order to customize completely FIspace Frontend.
- **Kanban Board**: Users have a personal Kanban Board in order to manage there all his/her personal task.
- **Business & Capabilities**: Depending on the FIspace user role (Business Architect, App Developer or user) he/she has access to a new section where new Business process or capabilities can defined to be used by new apps.
- **Responsive adaptation**: FIspace can be visualized correctly in different devices.

New features are planned to be completed in future FIspace platform versions.

2.1.2 Related Technologies

From the technical point of view, development team is using a wide range of technologies in order to utilize the most appropriate solution in each use case and functionality. Some of these technologies are:

Back-End:

- Spring MVC 3.2.3: an HTTP- and servlet-based framework providing hooks for extension and customization for web applications and RESTful Web services (http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html).
- Spring Security 3.2.1: is a Java/Java EE framework that provides authentication, authorization and other security features for enterprise applications (http://projects.spring.io/spring-security/)
- Hibernate: object-relational mapping library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database (http://hibernate.org/).

Front-End:

- HTML5 + CSS3
- Jquery: JavaScript library that makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. (http://jquery.com/).
- Bootstrap: Bootstrap is a HTML, CSS, and JS framework for developing responsive, mobile projects on the web (http://getbootstrap.com/).
- Momentjs: A javascript date library for parsing, validating, manipulating, and formatting dates (http://momentjs.com/).
- typeahead.js: A flexible JavaScript library that provides a strong foundation for building robust typeaheads (https://twitter.github.io/typeahead.js/).
- Fontawesome: Font Awesome gives you scalable vector icons (http://fortawesome.github.io/Font-Awesome/).
- Hoganjs: JavaScript templating (http://twitter.github.io/hogan.js/).
- Excanvas: ExplorerCanvas helps Internet Explorer to support the HTML5 canvas tag to allow 2D command-based drawing functionality (http://excanvas.sourceforge.net/)
- Tablesorter: jQuery plugin for turning a standard HTML table with THEAD and TBODY tags into a sortable table without page refreshes (http://tablesorter.com/docs/).
- jquery-pagewalkthrough.js: Creates popups to show web functionality (http://jwarby.github.io/jquery-pagewalkthrough/).
- JQuery Growl: Notification bubble/popup (http://ksylvest.github.io/jquery-growl/).
- jquery.flot: Javascript plotting library (http://www.flotcharts.org/).
- jquery.sequence: Responsive Slider with Advanced CSS3 (http://www.sequencejs.com/).
- Angularjs: Open web application framework (https://angularjs.org/).
- Google Maps API: Google API to manage different maps and localizations (https://developers.google.com/maps/).

- SweetAlert: Replacement for JavaScript's default alerts. (https://github.com/t4t5/sweetalert/).

The Web application approach is deployed in a Wildfly Server 8.2.0.Final (http://wildfly.org/).

The FIspace Front-End database is running in a MySql Server 5.6.19 (http://www.mysql.com).

The Front-End component interacts with Wirecloud and WStore. These interactions are performed using both Generic Enables embedded in an Iframe.

For the JavaFX proof of concept application that has been developed the technology used is JavaFX 2.0 (http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm). A JavaFX based application that can be run as a Desktop application.

2.1.3 Environment Requirements

The main technical requirement for the Front-End are:

- Wildfly Server 8.2.0.Final instance
- Public IP Address
- MySql Server 5.6.19 instance.
- Keycloak Server 1.0.4.Final instance [32].

2.2 Interfaces / API

User Front-End component provides two APIs in order to interact with the application. The first one is an internal API which manages all the operations between the front-end and back-end of our web application. These operations can be directly accessed from the browser (navigation URLs) or can be called by AJAX (Asynchronous JavaScript + XML: http://en.wikipedia.org/wiki/Ajax_%28programming%29) calls from our JavaScript code.

The second one is completely independent from the previous API. The main functionality of this API is to manage an external access to the database providing some operations to obtain some information.

Both API Operations are detailed in the next section.

Function	Brief description
SPRING MVC Controllers	REST API based on the Spring MVC (Model View Controller). Different Spring Controllers provided all the functionality and navigation for the web application JSPs.
DB Rest Module	REST API that provides some operations to interact with the database from an external point.

Table 6: User Front-End API

2.2.1 API operations

2.2.1.1 SPRING MVC Controllers Operations:

Verb	URI	Description
GET	/fispace/application/applicationChannel/	Atmosphere notification Websockets Channel
GET	/fispace/application/getUserNotificationsPage	Redirection to 'Notification' Panel
GET	/fispace/application/getAllUserNotifications	Get all user notifications
POST	/fispace/application/markNotificationAsRead/{id}	Mark a notification as read in the database
DELETE	/fispace/application/deleteNotification/{id}	Delete a notification in the database
GET	/fispace/application/getAllUserUnreadMailboxes	Get all user unread internal mailboxes
GET	/fispace/auth/login	Redirection to 'Login' Page
GET	/fispace/auth/registration	Redirection to 'Registration' Page
POST	/fispace/auth/registration	Validation of 'Registration' Form and creation of a new user
GET	/fispace/chat/chatChannel	Atmosphere chat Websockets Channel
POST	/fispace/chat/chatChannel	Atmosphere chat Websockets Channel
GET	/fispace/chat/getOnlinePartners	Get all online partners of a user
GET	/fispace/chat/sendOnlinePartners	Send all online partners of a user
GET	/fispace/companies/createCompany	Redirection to 'Create New Company' Form

Verb	URI	Description
POST	/fispace/companies/createNewCompany	Validation and creation of a new company profile
GET	/fispace/companies/{companyName:.*}	Redirection to a company profile
POST	/fispace/companies/loadpicture	Update company profile picture
POST	/fispace/companies/loadpictureBackground	Update company profile background picture
POST	/fispace/companies/followCompany	Follow a company profile
POST	/fispace/companies/removeFollower/{companyName:.*}"	Remove a follower in a company profile
POST	/fispace/companies/unfollowCompany	Unfollow a company profile
GET	/fispace/companies/getSuggestedCompanies	Get all suggested companies
GET	/fispace/communities/mycommunities	Redirection to ‘My Communities’ page
GET	/fispace/communities/createCommunity	Redirection to ‘Create New Community Form
POST	/fispace/communities/createNewCommunity	Validation and creation of a new community profile
GET	/fispace/communities/{communityName:.*}	Redirection to a community profile
POST	/fispace/communities/joinCommunity	Join a community profile
POST	/fispace/communities/unjoinCommunity	Unjoin a community profile
POST	/fispace/communities/loadpicture	Update community profile picture
POST	/fispace/communities/loadpictureBackground	Update community profile background picture
GET	/fispace/main/home	Redirection to ‘Home’ page
GET	/fispace/main/{email:.*}	Redirection to a user profile

Verb	URI	Description
POST	/fispace/main/loadpicture	Update user profile picture
POST	/fispace/main/loadpictureBackground	Update user profile background picture
POST	/fispace/main/addPartner	Add another user as partner
POST	/fispace/main/addPartnerRequest	To Accept a partner request
POST	/fispace/main/rejectPartnerRequest	To Reject a partner request
GET	/fispace/main/partnerRequests	Redirection to ‘Partner Request’ page
POST	/fispace/main/removePartner	Remove a partner
GET	/fispace/main/getSuggestedUsers	Get all suggested companies
GET	/fispace/main/wirecloud	Redirection to ‘Applications’ page
GET	/fispace/main/store	Redirection to ‘Store’ page
GET	/fispace/main/search/{name}	Search a username in database
POST	/fispace/main/updateTheme/{theme}	Update Theme in database
GET	/fispace/main/fetchUserIcon	Get user small profile picture
GET	/fispace/kanban/fetchKanbanTasks	Get all Kanban board Tasks
GET	/fispace/kanban/mykanban	Redirection to ‘My Kanban’ page
POST	/fispace/kanban/createNewTask	Create a new Kanban task
POST	/fispace/kanban/editTask/{id}	Edit a Kanban task message by id
POST	/fispace/kanban/updateTaskCategory/{id}	Edit a Kanban task category by id

Verb	URI	Description
DELETE	/fispace/kanban/deleteTask/{id}	Delete a Kanban task by id
GET	/fispace/mail/inbox	Redirection to ‘Inbox’ page
GET	/fispace/mail/sendMessage	Send a message and store it in the database
GET	/fispace/mail/saveDraft	Store a message as draft in the database
GET	/fispace/mail/getMailboxes	Get all messages
GET	/fispace/mail/getMailbox/{id}	Get a message by Id
POST	/fispace/mail/deleteMailbox/{id}	Delete a message by Id in the database
POST	/fispace/mail/markAsRead/{id}	Mark as read a message by Id in the database
POST	/fispace/mail/markAsUnread/{id}	Mark as unread a message by Id in the database
POST	/fispace/mail/changeFavouriteStatus/{id}	Change favourite status of a message by Id in the database
POST	/fispace/mail/archiveMailbox/{id}	Change archive status of a message by Id in the database
POST	/fispace/mail/unarchiveMailbox/{id}	Change archive status of a message by Id in the database
POST	/fispace/mail/selectTag/{id}	Change tag of a message by Id in the database
GET	/fispace/options/help	Redirection to ‘Help’ page
GET	/fispace/options/about	Redirection to ‘About’ page
GET	/fispace/options/terms	Redirection to ‘Terms’ page
GET	/fispace/options/advancedSearch	Redirection to ‘Advanced Search’ page
POST	/fispace/options/executeAdvancedSearchUsers	Execute advanced search of users

Verb	URI	Description
POST	/fispace/options/executeAdvancedSearchCompanies	Execute advanced search of companies
POST	/fispace/options/executeAdvancedSearchCompanies	Execute advanced search of companies
POST	/fispace/options/executeAdvancedSearchCommunities	Execute advanced search of communities
GET	/fispace/options/account	Redirection to ‘Account’ page
POST	/fispace/options/updateUser	Update User information in the database
POST	/fispace/options/updateAccount	Update Account information in the database
POST	/fispace/options/updateProfileInfo	Update Profile Info information in the database
POST	/fispace/options/updateCompanyProfile	Update company profile information in the database
GET	/fispace/options/visitNumberProfile	Get number of visits of a profile
GET	/fispace/options/unreadMessagesNumber	Get unread messages number
GET	/fispace/options/selectCompany/{name}	Select a company as user company
GET	/fispace/options/visitsProfile	Get visits details of a profile
DELETE	/fispace/options/removeAccount	Remove a FIspace account
DELETE	/fispace/options/removeCompany/{name}	Remove a company in FIspace
DELETE	/fispace/options/removeCommunity/{name}	Remove a community in FIspace
GET	/fispace/options/businessCapabilities	Redirection to ‘Business&Capabilities’ page
POST	/fispace/options/createCapabilityType	Create a new Capability Type
POST	/fispace/options/createBusinessProcessTemplate	Create a new Business Process Template

Verb	URI	Description
POST	/fispace/options/createCapability	Create a new Capability
POST	/fispace/options/createBusinessProcess	Create a new Business Process
DELETE	/fispace/options/removeBusinessProcess	Remove a Business Process
DELETE	/fispace/options/removeBusinessProcessTemplate	Remove a Business Process Template
DELETE	/fispace/options/removeCapability	Remove a Capability
DELETE	/fispace/options/removeCapabilityType	Remove a Capability Type
GET	/fispace/news/fetchNews	Get all news
POST	/fispace/news/postNews	Post a new News
POST	/fispace/news/postNewsCompany	Post a new company News
DELETE	/fispace/news /deleteNews/{id}	Delete a post by ID
DELETE	/fispace/news/deleteCompanyNews/{id}/{name}	Delete a company post by ID
POST	/fispace/news/likeNews",	Store a ‘Like’ from one user to a post in database
POST	/fispace/news/commentNews	Store a ‘Comment’ from one user to a post in database
POST	/fispace/news/editNews	Edit a post in the database
GET	/fispace/news/shareNews/{id}	Share a post in personal profile

Table 7: Front-End – SPRING MVC Controllers Operations

2.2.1.2 DB Rest Module:

Verb	URI	Description
GET	/dbaccess/db/readNotif	Get all user notifications from the database
GET	/dbaccess/db/companiesList	Get all companies in FIspace from the data-base
GET	/dbaccess/db/loginUser?email={email}&password={password}	Validate user credentials to login
GET	/dbaccess/db/getCompany?name={CompanyName}	Get company information from the database
GET	/dbaccess/db/getUser?email={email}	Get user information from the database
POST	/dbaccess/db/updateAccountForm?email={email}	Update account information in the database
POST	/dbaccess/db/updateUserForm?email={email}	Update user information in the database
POST	/dbaccess/db/updateProfileInfoForm?email={email}	Update profile information in the database
POST	/dbaccess/db/updateUserImage?email={email}	Update user profile picture in the database
POST	/dbaccess/db/updateCompanyImage?companyName={companyName}	Update company profile picture in the data-base

Table 8: Front-End – DB Rest Module

2.3 Information model

User Front-End is based in Spring MVC framework so its architecture is distributed in different layer. The following diagram contains a High-level diagram of the different layer inside the application.

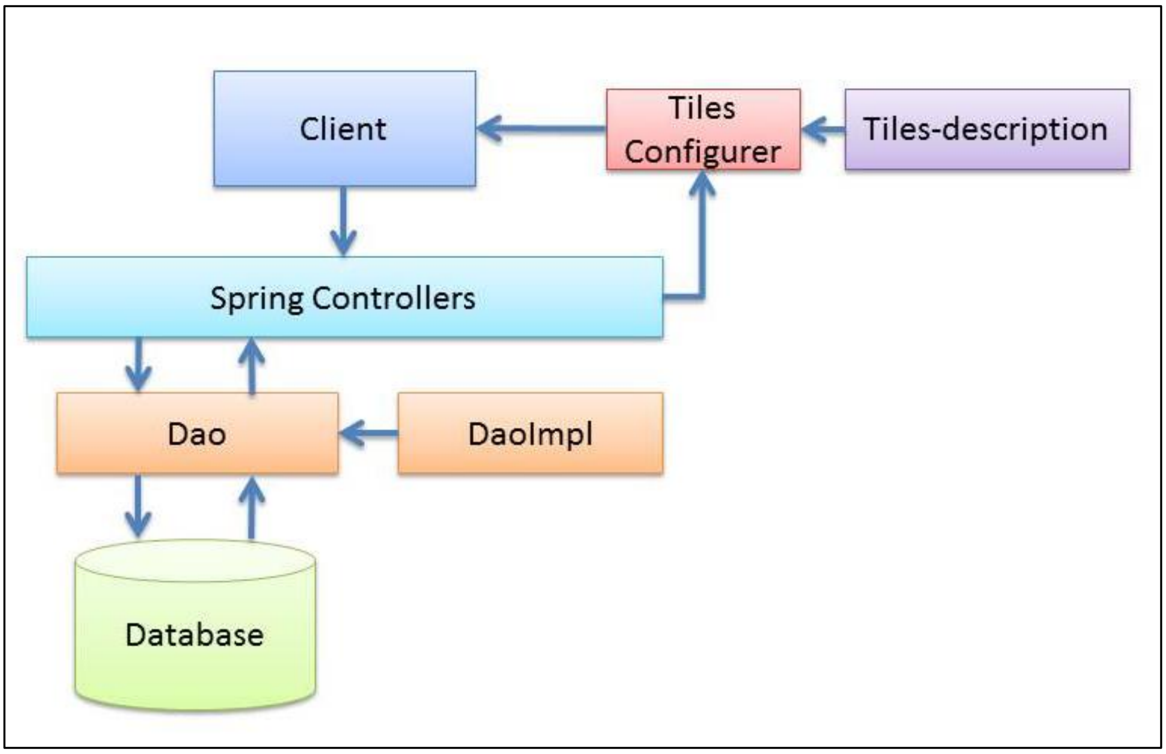


Figure 2: Front-End – High-level diagram of the different layer

Also this web application uses different context files in order to describe different aspects and beans that are used in different classes or services. The main context XML files are:

- **mvc-context.xml**: Contains some beans such as internationalization or Apache Tiles.
- **db-context.xml**: All the information related with the database access is located in this file.
- **root-context.xml**: This file imports all previous mentioned files.

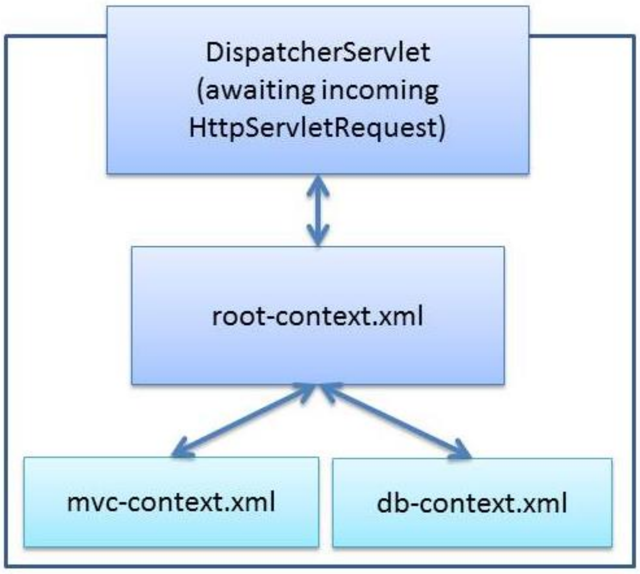


Figure 3: Front-End –Context files

2.3.1 Class and object model

Due to the fact this component is based in many classes. Only the main packages are going to be explained in details in this chapter.

Note: At the moment of writing this deliverable, the following figure represents the complete class diagram and the different relations between all of them.

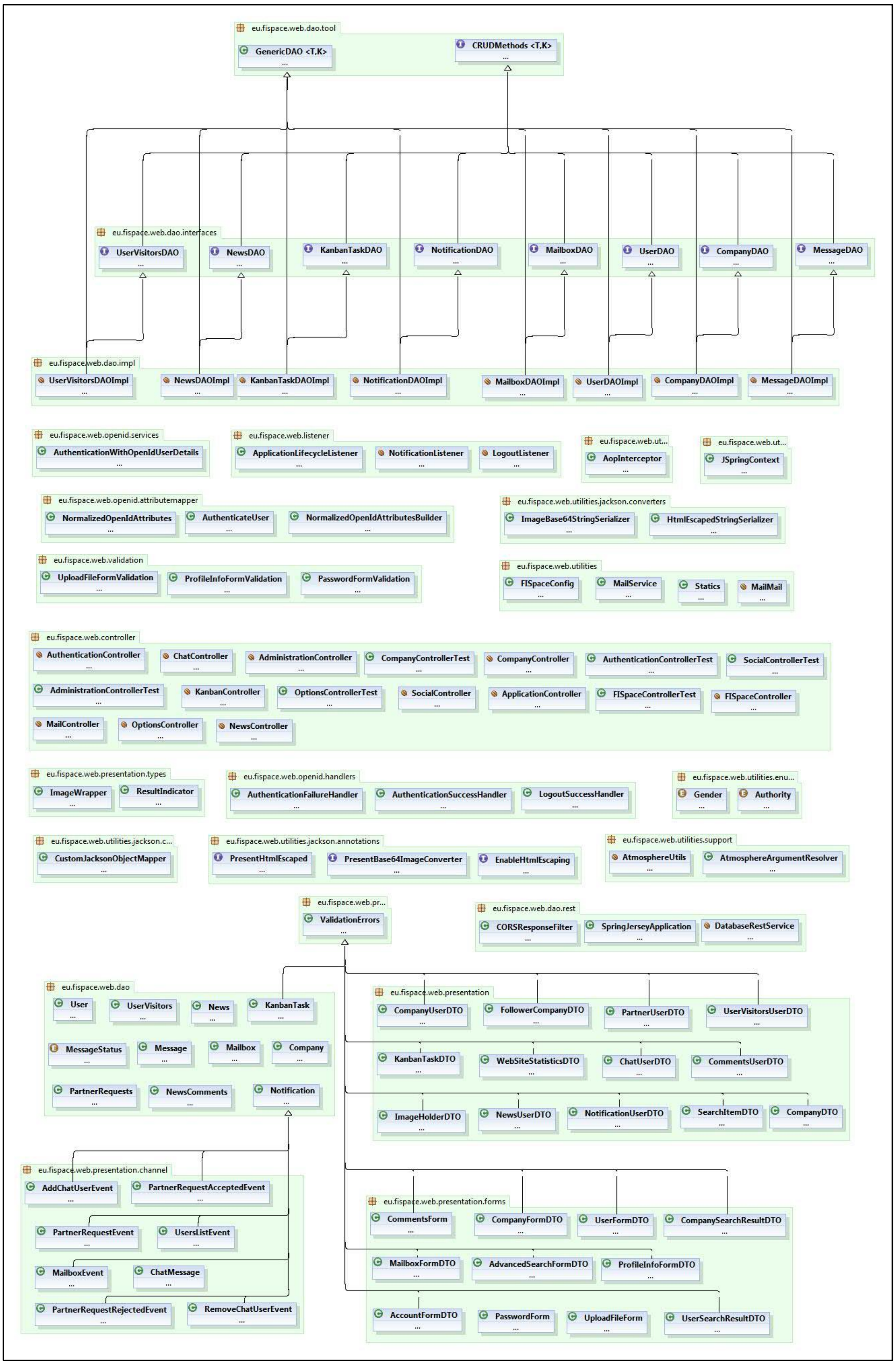
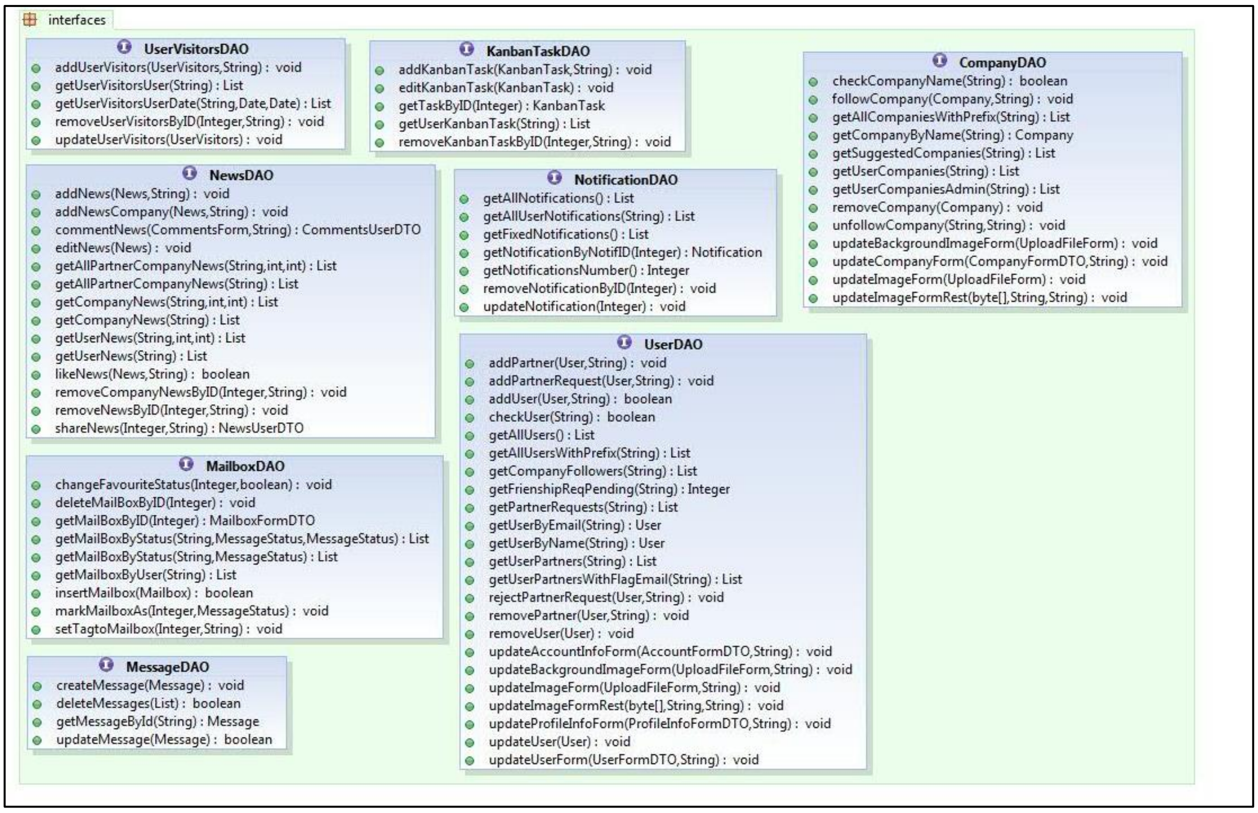


Figure 4: Front-End Class Diagram

The package *eu.fispace.web.dao.interfaces* contains all the interfaces that define the methods to interact with the database. These classes are part of the DAO layer from the architectural point of view. One class has been created for each object type that is store in the database.



```
interfaces

UserVisitorsDAO
  addUserVisitors(UserVisitors,String) : void
  getUserVisitorsUser(String) : List
  getUserVisitorsUserDate(String,Date,Date) : List
  removeUserVisitorsByID(Integer,String) : void
  updateUserVisitors(UserVisitors) : void

KanbanTaskDAO
  addKanbanTask(KanbanTask,String) : void
  editKanbanTask(KanbanTask) : void
  getTaskByID(Integer) : KanbanTask
  getUserKanbanTask(String) : List
  removeKanbanTaskByID(Integer,String) : void

CompanyDAO
  checkCompanyName(String) : boolean
  followCompany(Company,String) : void
  getAllCompaniesWithPrefix(String) : List
  getCompanyByName(String) : Company
  getSuggestedCompanies(String) : List
  getUserCompanies(String) : List
  getUserCompaniesAdmin(String) : List
  removeCompany(Company) : void
  unfollowCompany(String,String) : void
  updateBackgroundImageForm(UploadFileForm) : void
  updateCompanyForm(CompanyFormDTO,String) : void
  updateImageForm(UploadFileForm) : void
  updateImageFormRest(byte[],String,String) : void

NewsDAO
  addNews(News,String) : void
  addNewsCompany(News,String) : void
  commentNews(CommentsForm,String) : CommentsUserDTO
  editNews(News) : void
  getAllPartnerCompanyNews(String,int,int) : List
  getAllPartnerCompanyNews(String) : List
  getCompanyNews(String,int,int) : List
  getCompanyNews(String) : List
  getUserNews(String,int,int) : List
  getUserNews(String) : List
  likeNews(News,String) : boolean
  removeCompanyNewsByID(Integer,String) : void
  removeNewsByID(Integer,String) : void
  shareNews(Integer,String) : NewsUserDTO

NotificationDAO
  getAllNotifications() : List
  getAllUserNotifications(String) : List
  getFixedNotifications() : List
  getNotificationByNotifID(Integer) : Notification
  getNotificationsNumber() : Integer
  removeNotificationByID(Integer) : void
  updateNotification(Integer) : void

UserDAO
  addPartner(User,String) : void
  addPartnerRequest(User,String) : void
  addUser(User,String) : boolean
  checkUser(String) : boolean
  getAllUsers() : List
  getAllUsersWithPrefix(String) : List
  getCompanyFollowers(String) : List
  getFrienshipReqPending(String) : Integer
  getPartnerRequests(String) : List
  getUserByEmail(String) : User
  getUserByName(String) : User
  getUserPartners(String) : List
  getUserPartnersWithFlagEmail(String) : List
  rejectPartnerRequest(User,String) : void
  removePartner(User,String) : void
  removeUser(User) : void
  updateAccountInfoForm(AccountFormDTO,String) : void
  updateBackgroundImageForm(UploadFileForm,String) : void
  updateImageForm(UploadFileForm,String) : void
  updateImageFormRest(byte[],String,String) : void
  updateProfileInfoForm(ProfileInfoFormDTO,String) : void
  updateUser(User) : void
  updateUserForm(UserFormDTO,String) : void

MailboxDAO
  changeFavouriteStatus(Integer,boolean) : void
  deleteMailBoxByID(Integer) : void
  getMailBoxByID(Integer) : MailboxFormDTO
  getMailBoxByStatus(String,MessageStatus,MessageStatus) : List
  getMailBoxByStatus(String,MessageStatus) : List
  getMailboxByUser(String) : List
  insertMailbox(Mailbox) : boolean
  markMailboxAs(Integer,MessageStatus) : void
  setTagtoMailbox(Integer,String) : void

MessageDAO
  createMessage(Message) : void
  deleteMessages(List) : boolean
  getMessageById(String) : Message
  updateMessage(Message) : boolean
```

Figure 5: Front-End DAO interfaces

All these classes have their respective implementation in the package *eu.fispace.web.dao.impl.* Hibernate is used to execute the different connections with the internal database.

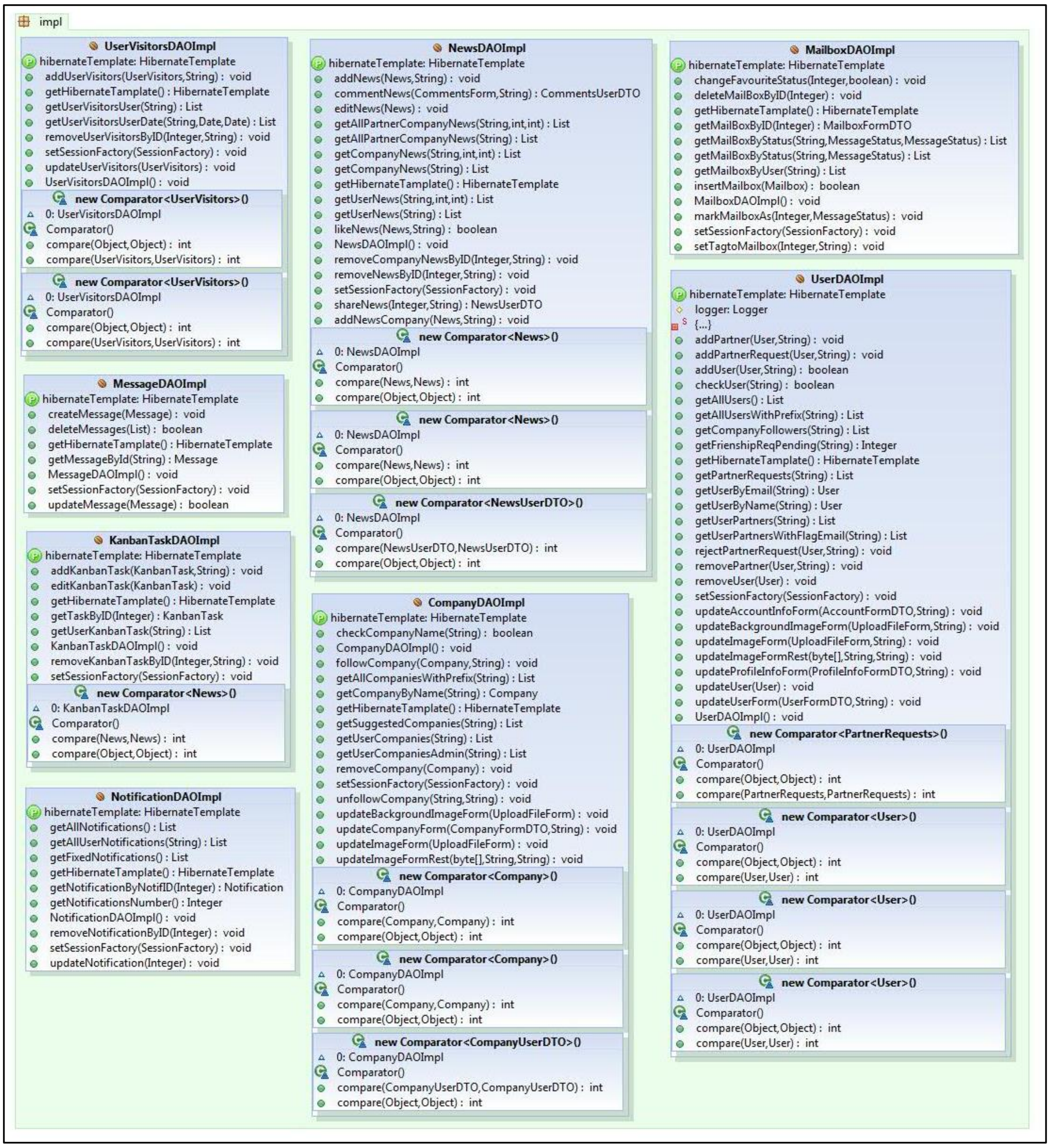

impl
UserVisitorsDAOImpl
hibernateTemplate: HibernateTemplate
addUserVisitors(UserVisitors,String) : void
getHibernateTamplate() : HibernateTemplate
getUserVisitorsUser(String) : List
getUserVisitorsUserDate(String,Date,Date) : List
removeUserVisitorsByID(Integer,String) : void
setSessionFactory(SessionFactory) : void
updateUserVisitors(UserVisitors) : void
UserVisitorsDAOImpl() : void
new Comparator<UserVisitors>()
0: UserVisitorsDAOImpl
Comparator()
compare(Object,Object) : int
compare(UserVisitors,UserVisitors) : int
new Comparator<UserVisitors>()
0: UserVisitorsDAOImpl
Comparator()
compare(Object,Object) : int
compare(UserVisitors,UserVisitors) : int
MessageDAOImpl
hibernateTemplate: HibernateTemplate
createMessage(Message) : void
deleteMessages(List) : boolean
getHibernateTamplate() : HibernateTemplate
getMessageById(String) : Message
MessageDAOImpl() : void
setSessionFactory(SessionFactory) : void
updateMessage(Message) : boolean
KanbanTaskDAOImpl
hibernateTemplate: HibernateTemplate
addKanbanTask(KanbanTask,String) : void
editKanbanTask(KanbanTask) : void
getHibernateTamplate() : HibernateTemplate
getTaskByID(Integer) : KanbanTask
getUserKanbanTask(String) : List
KanbanTaskDAOImpl() : void
removeKanbanTaskByID(Integer,String) : void
setSessionFactory(SessionFactory) : void
new Comparator<News>()
0: KanbanTaskDAOImpl
Comparator()
compare(News,News) : int
compare(Object,Object) : int
NotificationDAOImpl
hibernateTemplate: HibernateTemplate
getAllNotifications() : List
getAllUserNotifications(String) : List
getFixedNotifications() : List
getHibernateTamplate() : HibernateTemplate
getNotificationByNotifID(Integer) : Notification
getNotificationsNumber() : Integer
NotificationDAOImpl() : void
removeNotificationByID(Integer) : void
setSessionFactory(SessionFactory) : void
updateNotification(Integer) : void
NewsDAOImpl
hibernateTemplate: HibernateTemplate
addNews(News,String) : void
commentNews(CommentsForm,String) : CommentsUserDTO
editNews(News) : void
getAllPartnerCompanyNews(String,int,int) : List
getAllPartnerCompanyNews(String) : List
getCompanyNews(String,int,int) : List
getCompanyNews(String) : List
getHibernateTamplate() : HibernateTemplate
getUserNews(String,int,int) : List
getUserNews(String) : List
likeNews(News,String) : boolean
NewsDAOImpl() : void
removeCompanyNewsByID(Integer,String) : void
removeNewsByID(Integer,String) : void
setSessionFactory(SessionFactory) : void
shareNews(Integer,String) : NewsUserDTO
addNewsCompany(News,String) : void
new Comparator<News>()
0: NewsDAOImpl
Comparator()
compare(News,News) : int
compare(Object,Object) : int
new Comparator<News>()
0: NewsDAOImpl
Comparator()
compare(News,News) : int
compare(Object,Object) : int
new Comparator<NewsUserDTO>()
0: NewsDAOImpl
Comparator()
compare(NewsUserDTO,NewsUserDTO) : int
compare(Object,Object) : int
CompanyDAOImpl
hibernateTemplate: HibernateTemplate
checkCompanyName(String) : boolean
CompanyDAOImpl() : void
followCompany(Company,String) : void
getAllCompaniesWithPrefix(String) : List
getCompanyByName(String) : Company
getHibernateTamplate() : HibernateTemplate
getSuggestedCompanies(String) : List
getUserCompanies(String) : List
getUserCompaniesAdmin(String) : List
removeCompany(Company) : void
setSessionFactory(SessionFactory) : void
unfollowCompany(String,String) : void
updateBackgroundImageForm(UploadFileForm) : void
updateCompanyForm(CompanyFormDTO,String) : void
updateImageForm(UploadFileForm) : void
updateImageFormRest(byte[],String,String) : void
new Comparator<Company>()
0: CompanyDAOImpl
Comparator()
compare(Company,Company) : int
compare(Object,Object) : int
new Comparator<Company>()
0: CompanyDAOImpl
Comparator()
compare(Company,Company) : int
compare(Object,Object) : int
new Comparator<CompanyUserDTO>()
0: CompanyDAOImpl
Comparator()
compare(CompanyUserDTO,CompanyUserDTO) : int
compare(Object,Object) : int
MailboxDAOImpl
hibernateTemplate: HibernateTemplate
changeFavouriteStatus(Integer,boolean) : void
deleteMailBoxByID(Integer) : void
getHibernateTamplate() : HibernateTemplate
getMailBoxByID(Integer) : MailboxFormDTO
getMailBoxByStatus(String,MessageStatus,MessageStatus) : List
getMailBoxByStatus(String,MessageStatus) : List
getMailboxByUser(String) : List
insertMailbox(Mailbox) : boolean
MailboxDAOImpl() : void
markMailboxAs(Integer,MessageStatus) : void
setSessionFactory(SessionFactory) : void
setTagtoMailbox(Integer,String) : void
UserDAOImpl
hibernateTemplate: HibernateTemplate
logger: Logger
{...}
addPartner(User,String) : void
addPartnerRequest(User,String) : void
addUser(User,String) : boolean
checkUser(String) : boolean
getAllUsers() : List
getAllUsersWithPrefix(String) : List
getCompanyFollowers(String) : List
getFrienshipReqPending(String) : Integer
getHibernateTamplate() : HibernateTemplate
getPartnerRequests(String) : List
getUserByEmail(String) : User
getUserByName(String) : User
getUserPartners(String) : List
getUserPartnersWithFlagEmail(String) : List
rejectPartnerRequest(User,String) : void
removePartner(User,String) : void
removeUser(User) : void
setSessionFactory(SessionFactory) : void
updateAccountInfoForm(AccountFormDTO,String) : void
updateBackgroundImageForm(UploadFileForm,String) : void
updateImageForm(UploadFileForm,String) : void
updateImageFormRest(byte[],String,String) : void
updateProfileInfoForm(ProfileInfoFormDTO,String) : void
updateUser(User) : void
updateUserForm(UserFormDTO,String) : void
UserDAOImpl() : void
new Comparator<PartnerRequests>()
0: UserDAOImpl
Comparator()
compare(Object,Object) : int
compare(PartnerRequests,PartnerRequests) : int
new Comparator<User>()
0: UserDAOImpl
Comparator()
compare(Object,Object) : int
compare(User,User) : int
new Comparator<User>()
0: UserDAOImpl
Comparator()
compare(Object,Object) : int
compare(User,User) : int
new Comparator<User>()
0: UserDAOImpl
Comparator()
compare(Object,Object) : int
compare(User,User) : int

Figure 6: Front-End DAO implementation classes

One of the most important packages in this component is *eu.fispace.web.controller.*It contains all the spring Controllers. These classes manage all the redirections inside the application and all the interactions between View layer and Service layer. A different controller class has been created in order to group functionalities that are located in similar part of the View layer or manage similar objects.

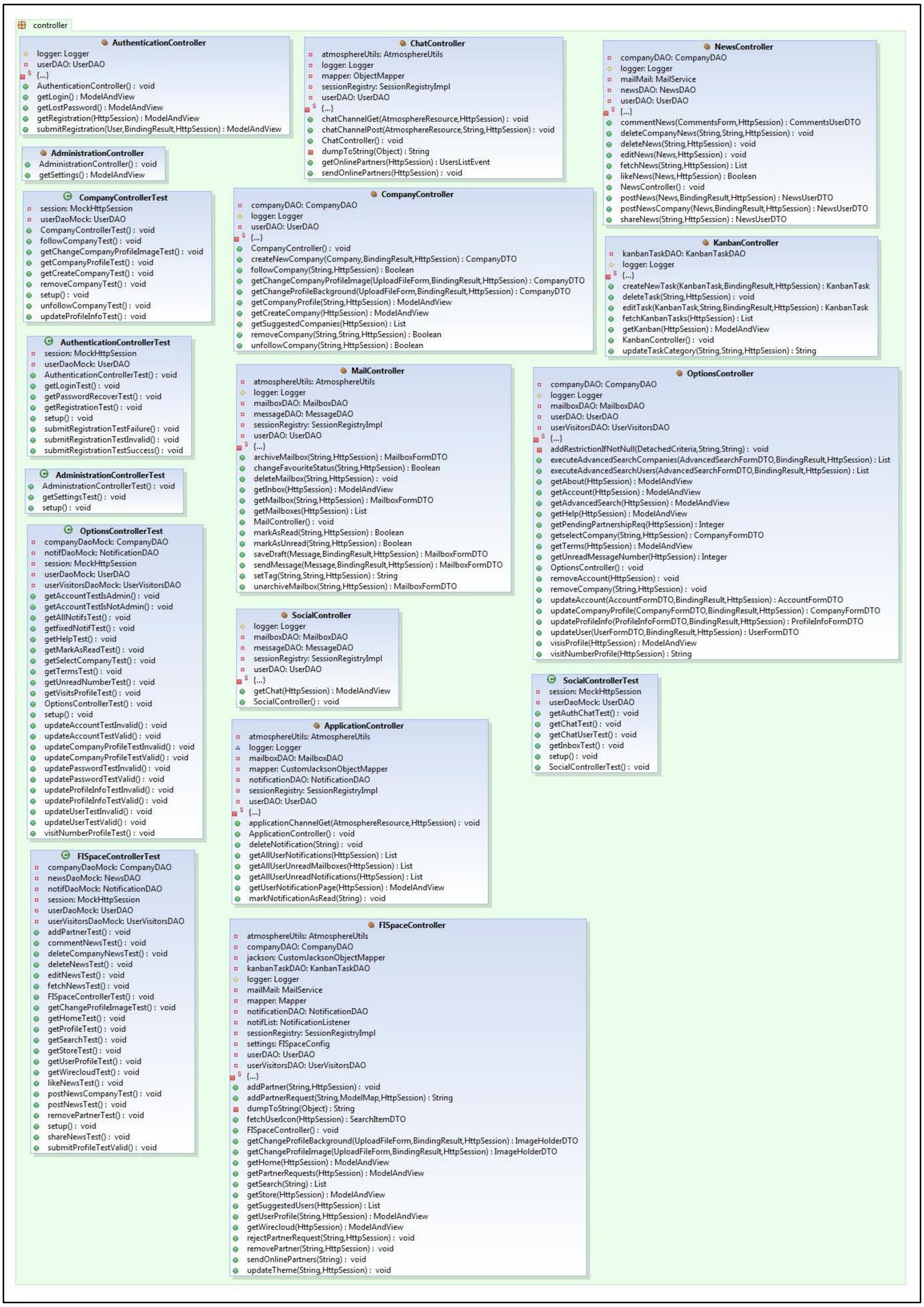

controller
AuthenticationController
logger: Logger
userDAO: UserDAO
{...}
AuthenticationController() : void
getLogin() : ModelAndView
getLostPassword() : ModelAndView
getRegistration(HttpSession) : ModelAndView
submitRegistration(User,BindingResult,HttpSession) : ModelAndView
ChatController
atmosphereUtils: AtmosphereUtils
logger: Logger
mapper: ObjectMapper
sessionRegistry: SessionRegistryImpl
userDAO: UserDAO
{...}
chatChannelGet(AtmosphereResource,HttpSession) : void
chatChannelPost(AtmosphereResource,String,HttpSession) : void
ChatController() : void
dumpToString(Object) : String
getOnlinePartners(HttpSession) : UsersListEvent
sendOnlinePartners(HttpSession) : void
NewsController
companyDAO: CompanyDAO
logger: Logger
mailMail: MailService
newsDAO: NewsDAO
userDAO: UserDAO
{...}
commentNews(CommentsForm,HttpSession) : CommentsUserDTO
deleteCompanyNews(String,String,HttpSession) : void
deleteNews(String,HttpSession) : void
editNews(News,HttpSession) : void
fetchNews(String,HttpSession) : List
likeNews(News,HttpSession) : Boolean
NewsController() : void
postNews(News,BindingResult,HttpSession) : NewsUserDTO
postNewsCompany(News,BindingResult,HttpSession) : NewsUserDTO
shareNews(String,HttpSession) : NewsUserDTO
AdministrationController
AdministrationController() : void
getSettings() : ModelAndView
CompanyControllerTest
session: MockHttpSession
userDaoMock: UserDAO
CompanyControllerTest() : void
followCompanyTest() : void
getChangeCompanyProfileImageTest() : void
getCompanyProfileTest() : void
getCreateCompanyTest() : void
removeCompanyTest() : void
setup() : void
unfollowCompanyTest() : void
updateProfileInfoTest() : void
CompanyController
companyDAO: CompanyDAO
logger: Logger
userDAO: UserDAO
{...}
CompanyController() : void
createNewCompany(Company,BindingResult,HttpSession) : CompanyDTO
followCompany(String,HttpSession) : Boolean
getChangeCompanyProfileImage(UploadFileForm,BindingResult,HttpSession) : CompanyDTO
getChangeProfileBackground(UploadFileForm,BindingResult,HttpSession) : CompanyDTO
getCompanyProfile(String,HttpSession) : ModelAndView
getCreateCompany(HttpSession) : ModelAndView
getSuggestedCompanies(HttpSession) : List
removeCompany(String,String,HttpSession) : Boolean
unfollowCompany(String,HttpSession) : Boolean
KanbanController
kanbanTaskDAO: KanbanTaskDAO
logger: Logger
{...}
createNewTask(KanbanTask,BindingResult,HttpSession) : KanbanTask
deleteTask(String,HttpSession) : void
editTask(KanbanTask,String,BindingResult,HttpSession) : KanbanTask
fetchKanbanTasks(HttpSession) : List
getKanban(HttpSession) : ModelAndView
KanbanController() : void
updateTaskCategory(String,String,HttpSession) : String
AuthenticationControllerTest
session: MockHttpSession
userDaoMock: UserDAO
AuthenticationControllerTest() : void
getLoginTest() : void
getPasswordRecoverTest() : void
getRegistrationTest() : void
setup() : void
submitRegistrationTestFailure() : void
submitRegistrationTestInvalid() : void
submitRegistrationTestSuccess() : void
MailController
atmosphereUtils: AtmosphereUtils
logger: Logger
mailboxDAO: MailboxDAO
messageDAO: MessageDAO
sessionRegistry: SessionRegistryImpl
userDAO: UserDAO
{...}
archiveMailbox(String,HttpSession) : MailboxFormDTO
changeFavouriteStatus(String,HttpSession) : Boolean
deleteMailbox(String,HttpSession) : void
getInbox(HttpSession) : ModelAndView
getMailbox(String,HttpSession) : MailboxFormDTO
getMailboxes(HttpSession) : List
MailController() : void
markAsRead(String,HttpSession) : Boolean
markAsUnread(String,HttpSession) : Boolean
saveDraft(Message,BindingResult,HttpSession) : MailboxFormDTO
sendMessage(Message,BindingResult,HttpSession) : MailboxFormDTO
setTag(String,String,HttpSession) : String
unarchiveMailbox(String,HttpSession) : MailboxFormDTO
OptionsController
companyDAO: CompanyDAO
logger: Logger
mailboxDAO: MailboxDAO
userDAO: UserDAO
userVisitorsDAO: UserVisitorsDAO
{...}
addRestrictionIfNotNull(DetachedCriteria,String,String) : void
executeAdvancedSearchCompanies(AdvancedSearchFormDTO,BindingResult,HttpSession) : List
executeAdvancedSearchUsers(AdvancedSearchFormDTO,BindingResult,HttpSession) : List
getAbout(HttpSession) : ModelAndView
getAccount(HttpSession) : ModelAndView
getAdvancedSearch(HttpSession) : ModelAndView
getHelp(HttpSession) : ModelAndView
getPendingPartnershipReq(HttpSession) : Integer
getselectCompany(String,HttpSession) : CompanyFormDTO
getTerms(HttpSession) : ModelAndView
getUnreadMessageNumber(HttpSession) : Integer
OptionsController() : void
removeAccount(HttpSession) : void
removeCompany(String,HttpSession) : void
updateAccount(AccountFormDTO,BindingResult,HttpSession) : AccountFormDTO
updateCompanyProfile(CompanyFormDTO,BindingResult,HttpSession) : CompanyFormDTO
updateProfileInfo(ProfileInfoFormDTO,BindingResult,HttpSession) : ProfileInfoFormDTO
updateUser(UserFormDTO,BindingResult,HttpSession) : UserFormDTO
visisProfile(HttpSession) : ModelAndView
visitNumberProfile(HttpSession) : String
AdministrationControllerTest
AdministrationControllerTest() : void
getSettingsTest() : void
setup() : void
OptionsControllerTest
companyDaoMock: CompanyDAO
notifDaoMock: NotificationDAO
session: MockHttpSession
userDaoMock: UserDAO
userVisitorsDaoMock: UserVisitorsDAO
getAccountTestIsAdmin() : void
getAccountTestIsNotAdmin() : void
getAllNotifsTest() : void
getfixedNotifTest() : void
getHelpTest() : void
getMarkAsReadTest() : void
getSelectCompanyTest() : void
getTermsTest() : void
getUnreadNumberTest() : void
getVisitsProfileTest() : void
OptionsControllerTest() : void
setup() : void
updateAccountTestInvalid() : void
updateAccountTestValid() : void
updateCompanyProfileTestInvalid() : void
updateCompanyProfileTestValid() : void
updatePasswordTestInvalid() : void
updatePasswordTestValid() : void
updateProfileInfoTestInvalid() : void
updateProfileInfoTestValid() : void
updateUserTestInvalid() : void
updateUserTestValid() : void
visitNumberProfileTest() : void
SocialController
logger: Logger
mailboxDAO: MailboxDAO
messageDAO: MessageDAO
sessionRegistry: SessionRegistryImpl
userDAO: UserDAO
{...}
getChat(HttpSession) : ModelAndView
SocialController() : void
SocialControllerTest
session: MockHttpSession
userDaoMock: UserDAO
getAuthChatTest() : void
getChatTest() : void
getChatUserTest() : void
getInboxTest() : void
setup() : void
SocialControllerTest() : void
ApplicationController
atmosphereUtils: AtmosphereUtils
logger: Logger
mailboxDAO: MailboxDAO
mapper: CustomJacksonObjectMapper
notificationDAO: NotificationDAO
sessionRegistry: SessionRegistryImpl
userDAO: UserDAO
{...}
applicationChannelGet(AtmosphereResource,HttpSession) : void
ApplicationController() : void
deleteNotification(String) : void
getAllUserNotifications(HttpSession) : List
getAllUserUnreadMailboxes(HttpSession) : List
getAllUserUnreadNotifications(HttpSession) : List
getUserNotificationPage(HttpSession) : ModelAndView
markNotificationAsRead(String) : void
FISpaceControllerTest
companyDaoMock: CompanyDAO
newsDaoMock: NewsDAO
notifDaoMock: NotificationDAO
session: MockHttpSession
userDaoMock: UserDAO
userVisitorsDaoMock: UserVisitorsDAO
addPartnerTest() : void
commentNewsTest() : void
deleteCompanyNewsTest() : void
deleteNewsTest() : void
editNewsTest() : void
fetchNewsTest() : void
FISpaceControllerTest() : void
getChangeProfileImageTest() : void
getHomeTest() : void
getProfileTest() : void
getSearchTest() : void
getStoreTest() : void
getUserProfileTest() : void
getWirecloudTest() : void
likeNewsTest() : void
postNewsCompanyTest() : void
postNewsTest() : void
removePartnerTest() : void
setup() : void
shareNewsTest() : void
submitProfileTestValid() : void
FISpaceController
atmosphereUtils: AtmosphereUtils
companyDAO: CompanyDAO
jackson: CustomJacksonObjectMapper
kanbanTaskDAO: KanbanTaskDAO
logger: Logger
mailMail: MailService
mapper: Mapper
notificationDAO: NotificationDAO
notifList: NotificationListener
sessionRegistry: SessionRegistryImpl
settings: FISpaceConfig
userDAO: UserDAO
userVisitorsDAO: UserVisitorsDAO
{...}
addPartner(String,HttpSession) : void
addPartnerRequest(String,ModelMap,HttpSession) : String
dumpToString(Object) : String
fetchUserIcon(HttpSession) : SearchItemDTO
FISpaceController() : void
getChangeProfileBackground(UploadFileForm,BindingResult,HttpSession) : ImageHolderDTO
getChangeProfileImage(UploadFileForm,BindingResult,HttpSession) : ImageHolderDTO
getHome(HttpSession) : ModelAndView
getPartnerRequests(HttpSession) : ModelAndView
getSearch(String) : List
getStore(HttpSession) : ModelAndView
getSuggestedUsers(HttpSession) : List
getUserProfile(String,HttpSession) : ModelAndView
getWirecloud(HttpSession) : ModelAndView
rejectPartnerRequest(String,HttpSession) : void
removePartner(String,HttpSession) : void
sendOnlinePartners(String) : void
updateTheme(String,HttpSession) : void

Figure 7: Front-End Spring MVC controller classes

2.3.2 Data base Model

User Front-End has its own database in order to store all the information related with Front-End functionality. This database stores only information as profile information, FIspace Front-End messages, news published, etc.

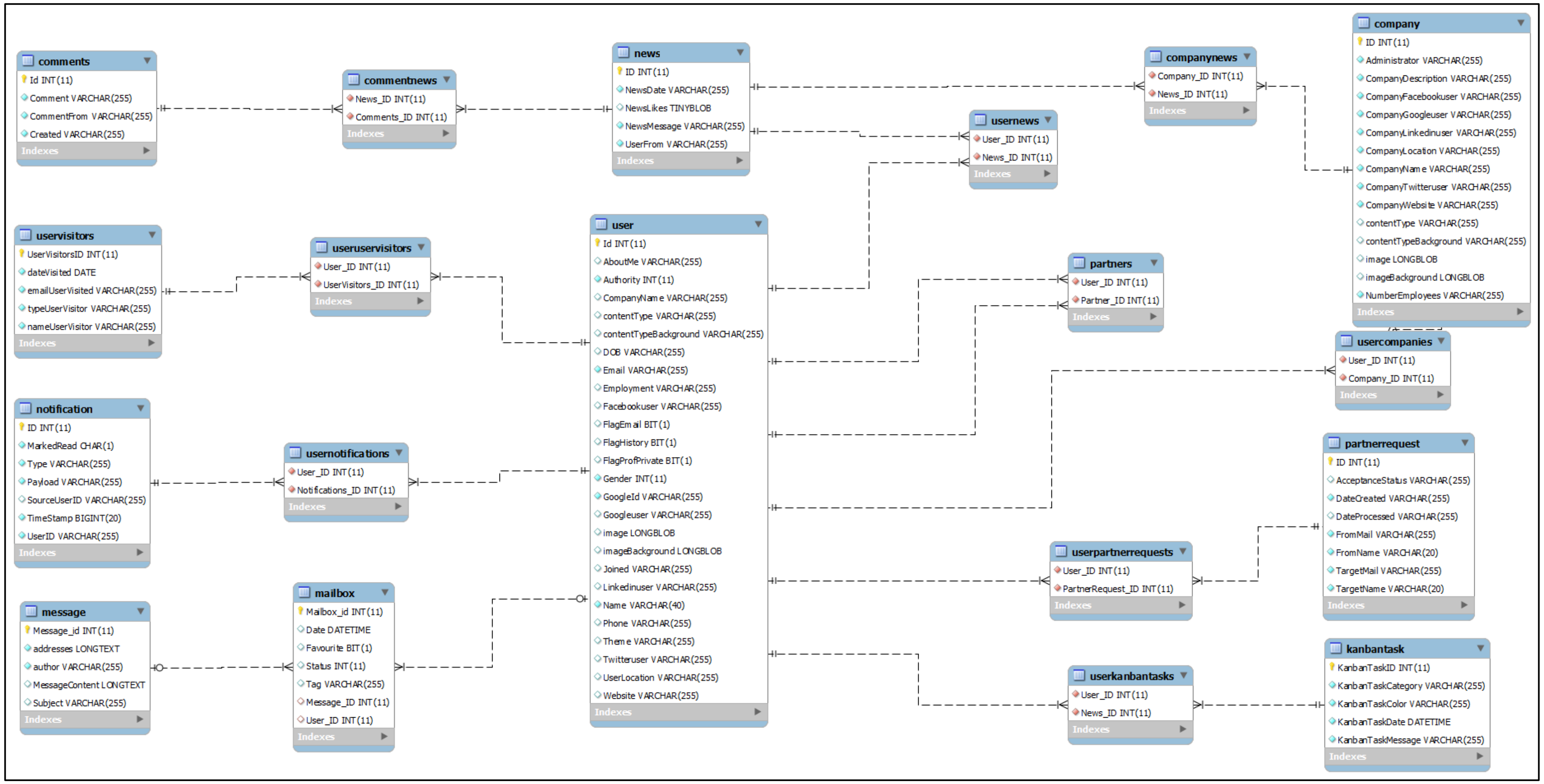


Figure 8: Front-End – Data model

2.4 Interaction model

In this section some interactions are shown in order to provide a high level idea of how some methods interact with other.

The first sequence diagram represents the *getHome* method in FIispaceController class. This method obtains different information from the session object about the user logged. Also it gets information from the database about some user features. All this information is stored in a ModelMap object that is sent to the view layer.

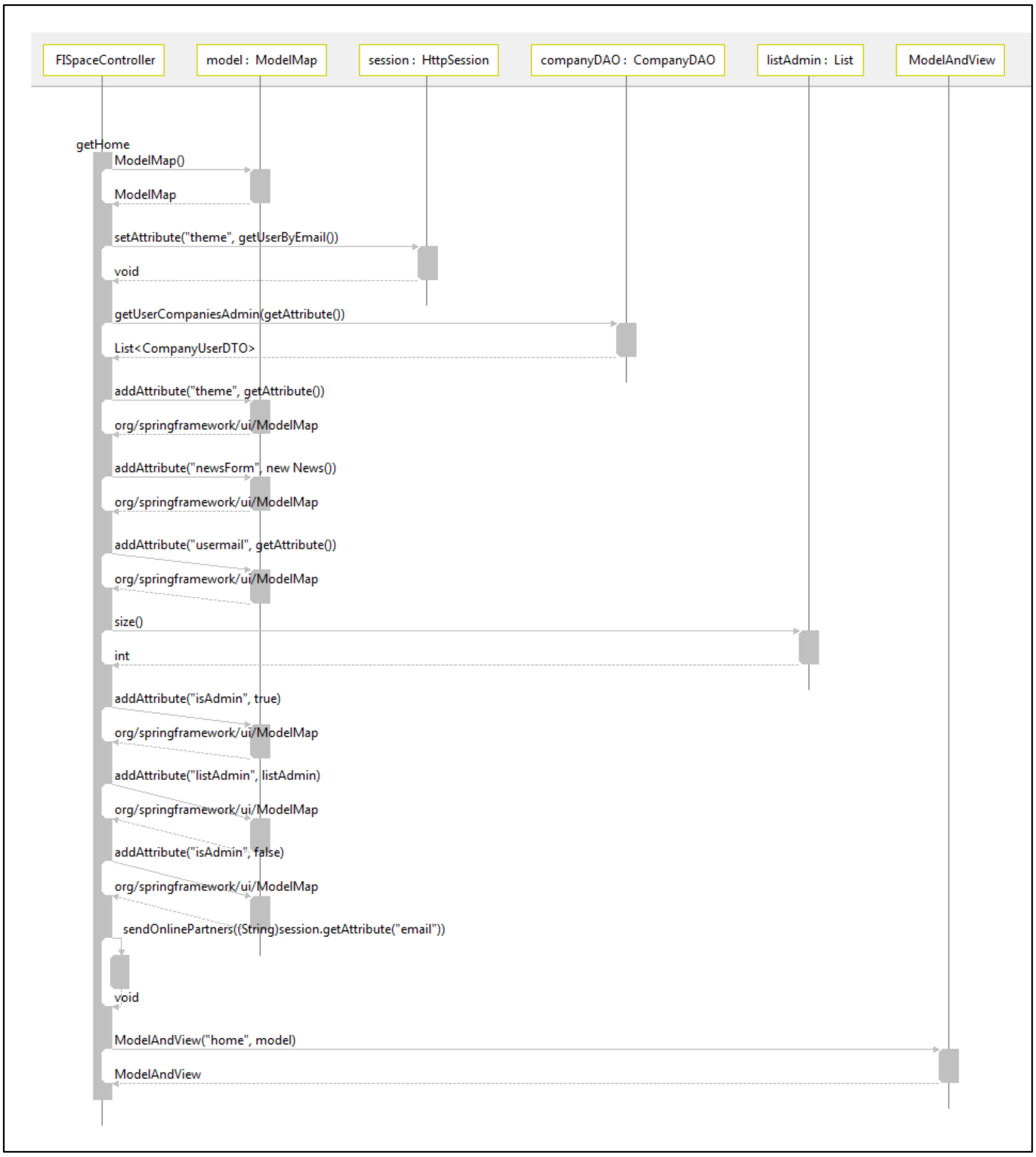

FISpaceController
model : ModelMap
session : HttpSession
companyDAO : CompanyDAO
listAdmin : List
ModelAndView
getHome
ModelMap()
ModelMap
setAttribute("theme", getUserByEmail())
void
getUserCompaniesAdmin(getAttribute())
List<CompanyUserDTO>
addAttribute("theme", getAttribute())
org/springframework/ui/ModelMap
addAttribute("newsForm", new News())
org/springframework/ui/ModelMap
addAttribute("usermail", getAttribute())
org/springframework/ui/ModelMap
size()
int
addAttribute("isAdmin", true)
org/springframework/ui/ModelMap
addAttribute("listAdmin", listAdmin)
org/springframework/ui/ModelMap
addAttribute("isAdmin", false)
org/springframework/ui/ModelMap
sendOnlinePartners((String)session.getAttribute("email"))
void
ModelAndView("home", model)
ModelAndView

Figure 9: Front-End – Sequence diagram 1

The second sequence diagram represents the *updateProfile* method in OptionController class. This method receives all the data introduced by the used in a form from the View layer. Data is validated and if it doesn't contain errors it is managed and updated in the database

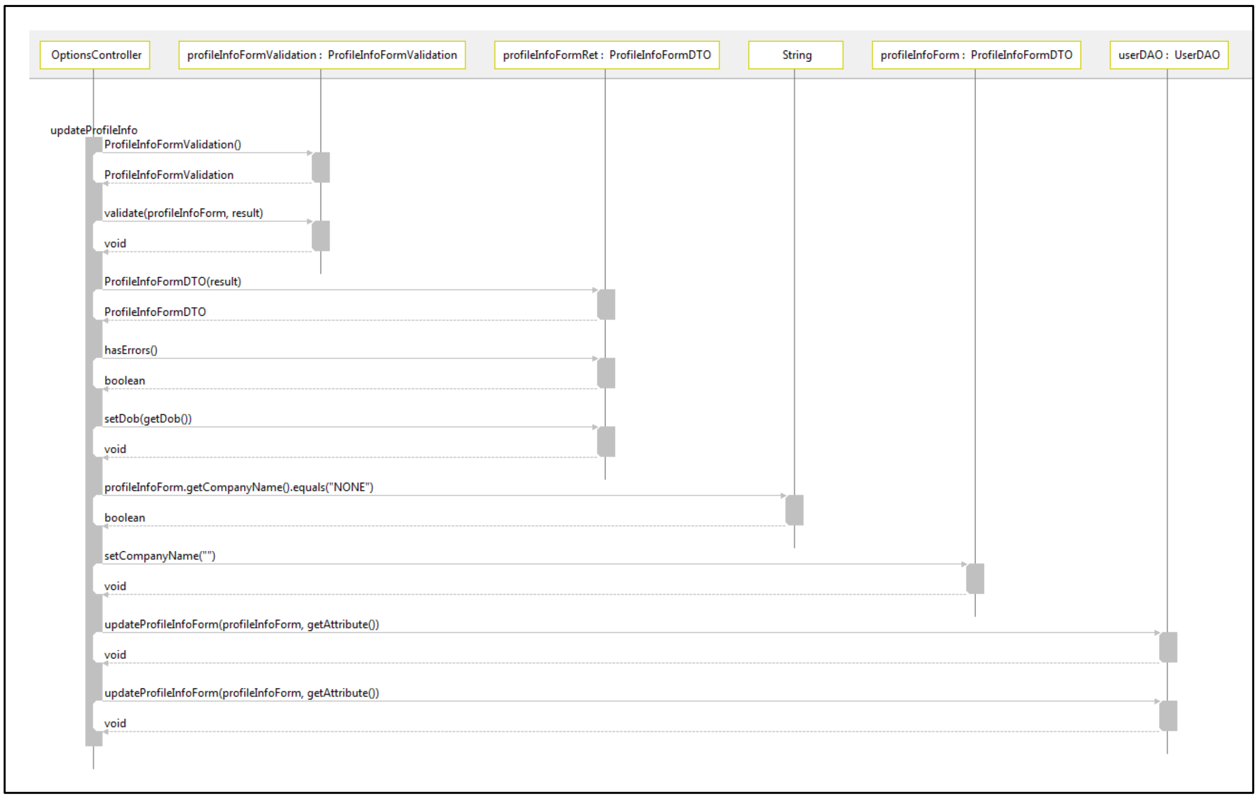

OptionsController
profileInfoFormValidation : ProfileInfoFormValidation
profileInfoFormRet : ProfileInfoFormDTO
String
profileInfoForm : ProfileInfoFormDTO
userDAO : UserDAO
updateProfileInfo
ProfileInfoFormValidation()
ProfileInfoFormValidation
validate(profileInfoForm, result)
void
ProfileInfoFormDTO(result)
ProfileInfoFormDTO
hasErrors()
boolean
setDob(getDob())
void
profileInfoForm.getCompanyName().equals("NONE")
boolean
setCompanyName("")
void
updateProfileInfoForm(profileInfoForm, getAttribute())
void
updateProfileInfoForm(profileInfoForm, getAttribute())
void

Figure 10: Front-End – Sequence diagram 2

To conclude, the following interaction model shows the different steps that a user performs to acces to FIspace platform. In particular, this process corresponds to a first time login in FIspace.

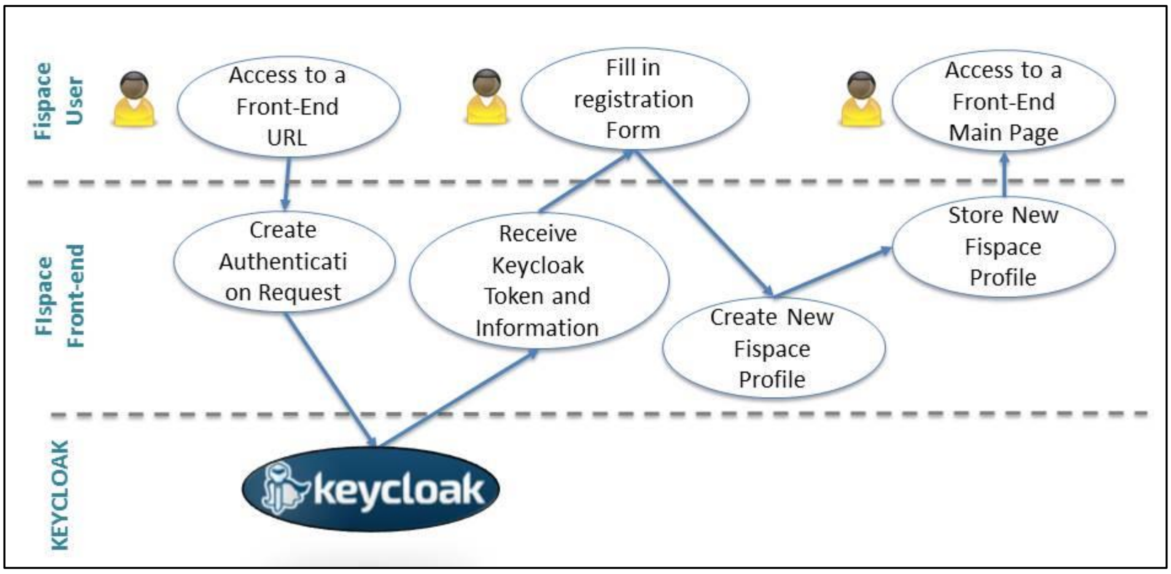

Fispace User
Access to a Front-End URL
Fill in registration Form
Access to a Front-End Main Page
FIspace Front-end
Create Authenticati on Request
Receive Keycloak Token and Information
Create New Fispace Profile
Store New Fispace Profile
KEYCLOAK
keycloak

Figure 11: Front-End – Interaction model

2.4.1 WireCloud GE

Wirecloud is a reference implementation of the FIWARE “Application Mashup“ Generic Enabler. It provides a mashup engine based on the W3C Widget specification. In addition to functionalities for personalized dashboards, WireCloud is able to interface with different app stores, from which apps can be obtained. It also comes with its own graphical wiring editor that allows for creating mashup applications out of two or more apps by connecting the output parameters of one app with input parameters of another app.

2.5 High level composite architecture

The following figure shows the high level composite architecture for the Front-End component.

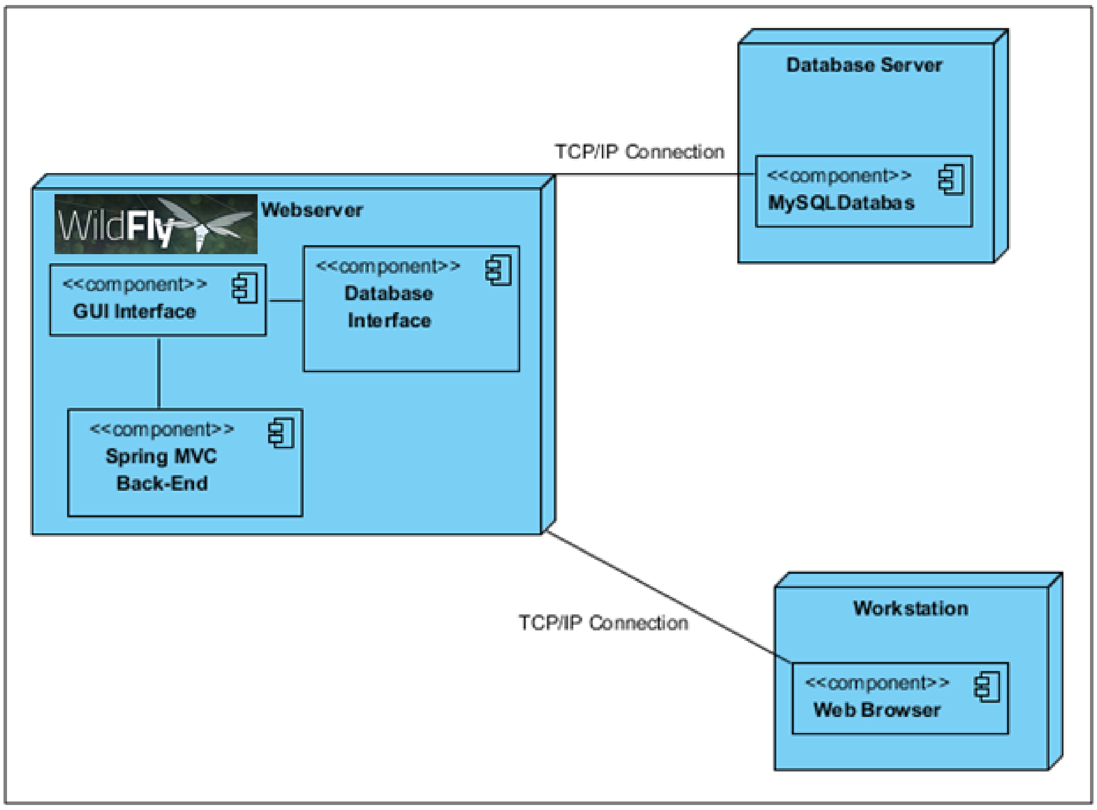


Figure 12: Front-End – High level composite architecture

2.6 User Front-End Guidance and resources

There is an online documentation as a tutorial at user level available (i.e. not developer) at http://dev.fispace.eu/doc/wiki/gui and http://dev.fispace.eu/doc/wiki/gui/gui-guide

In addition the document D200.7 Annex *“FIspace Front-End User Guide”* [10] describes the main features and functionalities of the Front-End component presented as a user guideline related to the business collaboration activities. It describes the graphical user interface (GUI) and the user interaction to login, manage its account and profile among other features.

The online documentation for the User Front-End Guide is available in the following web site: http://dev.fispace.eu/doc/wiki/gui/gui-guide

3 Glossary

The glossary provides the coherent terminological framework used in this document.

3.1 Terms and definitions

This section provides definitions of any terms that may be needed in order for the reader to understand the terminology used in the document. The author should define any definition/acronym or technical term used in the document that may be unfamiliar to the reader, and it is best to err on the side of too many rather than too few definitions. This also allows the author to frame a word within a specific context, which provides the reader with a common understanding of the author's definition.

Access control

Authorisation (or denegation) for performing a certain action (based on privileges management). The access control is carried out once the Identification and Authentication procedures have been performed.

Accounting

Process of gathering information about the usage of resources by subjects.

Acceptance and trust

Acceptability indicates the degree of approval of a technology by the users. It depends on whether the technology can satisfy the needs and expectations of its users and potential stakeholders. Within the framework of introducing new technologies, acceptability relates to social and individual aspects as well.

Application

Use of capabilities, including hardware, software and data, provided by an information system specific to the satisfaction of a set of user requirements in a given application domain.

Application Domain

Integrated set of problems, terms, information and tasks of a specific thematic domain that an application (e.g. an information system or a set of information systems) has to cope with.

Application Schema [ISO/FDIS 19109:2003]

Conceptual schema for data required by one or more applications.

Architecture (of a system) [ISO/IEC 10746-2:1996]

Set of rules to define the structure of a system and the interrelationships between its parts.

Architecture (of a system) [ISO/IEC 10746-2:1996]

Set of rules to define the structure of a system and the interrelationships between its parts.

Authentication

Process of verifying the identity of a certain subject. In other words authentication indicates whether a subject is who/what it seems to be.

Generally speaking, this proof can depend on a secret that can be, e.g. what somebody has (key, smart card, …), what somebody knows (password, …), what somebody is (biometrical data, …)

Authorisation

Process of determining whether a subject is allowed to have the specified types of access to a particular resource. This is done by evaluating applicable access control information contained in a so called authorisation context. Usually, authorisation is carried out after the identification and authentication. Once a subject is identified and authenticated, it may be authorized (or not) to perform different types of access.

Availability

Availability refers to the degree to which a system, subsystem, or equipment is in a specified operable and committable state at the start of a mission, when the mission is called for at an unknown, i.e., a random time. So, availability is the proportion of time that a system is in operating condition.

Capability

Capabilities are a set of functionalities, through a combination of software and hardware, used to provide services and data. They can reside in a system or for example in a terminal itself as embedded capabilities or they can be available through the network services and infrastructure and others communication technologies as external capabilities.

Catalogue [derived from http://www.opengeospatial.org/resources/?page=glossary]

Collection of entries, each of which describes and points to a feature collection. Catalogues include indexed listings of feature collections, their contents, their coverages, and of meta-information. A catalogue registers the existence, location, and description of feature collections held by an Information Community. Catalogues provide the capability to add and delete entries. A minimum Catalogue will include the name for the feature collection and the locational handle that specifies where these data may be found. Each catalogue is unique to its Information Community.

Certificate Authority

A Trusted Third Party, responsible for ensuring the binding between the public keys and the personal data of their respective owners.

Component

Hardware component (device) or Software Component.

Conceptual model [ISO/FDIS 19109:2003(E); ISO 19101]

Model that defines concepts of a universe of discourse.

Conceptual schema [ISO/FDIS 19109:2003(E); ISO 19101]

Formal description of a conceptual model.

Coverage [ISO 19123]

Function from a spatial, temporal or spatiotemporal domain to an attribute range. A coverage associates a position within its domain to a record of values of defined data types. Thus, a coverage is a feature with multiple values for each attribute type, where each direct position within the geometric representation of the feature has a single value for each attribute type.

Data acquisition

Methods of data acquisition include methods to collect background data, digitally acquire data from sensors, and subjective data (such as data acquired from questionnaires). In addition, data in the form of manually or automatically transcribed data and reductions of collected data is also considered sensor acquired data (but with a manual sensor – the analyst).

Description Logics

Family of logic based knowledge representation languages that are a decidable subset of first order logic with well-defined semantics and inferencing (problem decision procedures). In Description Logics, a distinction is made between the terminological knowledge and the assertional knowledge. This distinction is useful for knowledge base modelling and engineering: for modelling it is just natural to distinguish between concepts and individuals; for engineering it helps by separating key inference problems.

Digital Certificate

A kind of digital document that contains structured information about the identity of its owner along with her/his public key, signed all together with a Certificate Authority's private key.

Digital Signature

The encrypted form of a message with the private key of the owner, indicating in a secure way the creator of the message, as well as the identity of a signed data.

Encryption

The act of modifying the contents of a message in an algorithmic and secure way, so that it can not be observed or altered in while in transit.

End-User

All users that are involved in an application domain and that use the applications, the services built by the system users according to the system and service Architecture.

Feature [derived from ISO 19101]

Abstraction of a real world phenomenon [ISO 19101] perceived in the context of an Application. In this general sense, a feature corresponds to an "object" in analysis and design models.

Framework [http://www.opengeospatial.org/resources/?page=glossary]

An information architecture that comprises, in terms of software design, a reusable software template, or skeleton, from which key enabling and supporting services can be selected, configured and integrated with application code.

Generic

A service is generic, if it is independent of the application domain. A service infrastructure is generic, if it is independent of the application domain and if it can adapt to different organisational structures at different sites, without programming (ideally).

Identification

The identification process allows relating a person/device with the service environment. The “electronic identity” is something like a credential or a “business card”, suitable to be verified throughout the authentication process.

Implementation [http://www.opengeospatial.org/resources/?page=glossary]

Software package that conforms to a standard or specification. A specific instance of a more generally defined system.

Info-structure Service

Service that is required to operate a system oriented service in the sense that it plays an indispensable role in the operation of an architecture or system oriented service.

Interface [ISO 19119:2005; http://www.opengis.org/docs/02-112.pdf]

Named set of operations that characterize the behaviour of an entity.

The aggregation of operations in an interface, and the definition of interface, shall be for the purpose of software reusability. The specification of an interface shall include a static portion that includes definition of the operations. The specification of an interface shall include a dynamic portion that includes any restrictions on the order of invoking the operations.

Interoperability [ISO 19119:2005 or OGC; http://www.opengeospatial.org/resources/?page=glossary]

Capability to communicate, execute programs, or transfer data among various functional units in a manner that require the user to have little or no knowledge of the unique characteristics of those units [ISO 2382-1]. (http://www.opengeospatial.org/ogc/glossary/i)

Loose coupling [W3C; http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#loosecoupling]

Coupling is the dependency between interacting systems. This dependency can be decomposed into real dependency and artificial dependency: Real dependency is the set of features or services that a system consumes from other systems. The real dependency always exists and cannot be reduced. Artificial dependency is the set of factors that a system has to comply with in order to consume the features or services provided by other systems. Typical artificial dependency factors are language dependency, platform dependency, API dependency, etc. Artificial dependency always exists, but it or its cost can be reduced. Loose coupling describes the configuration in which artificial dependency has been reduced to the minimum.

Middleware [http://www.opengeospatial.org/resources/?page=glossary]

Software in a distributed computing environment that mediates between clients and servers.

Open Architecture [based on (Powell 1991)] [33]

Architecture whose specifications are published and made freely available to interested vendors and users with a view of widespread adoption of the architecture. An open architecture makes use of existing standards where appropriate and possible and otherwise contributes to the evolution of relevant new standards.

Operation [ISO 19119:2005; http://www.opengis.org/docs/02-112.pdf]

Specification of a transformation or query that an object may be called to execute. An operation has a name and a list of parameters.

Performance indicators definition (PI)

PIs are quantitative or qualitative measurements, agreed on beforehand, expressed as a percentage, index, rate or other value, which is monitored at regular or irregular intervals and can be compared with one or more criteria.

Platform (Service)

Set of infrastructural means and rules that describe how to specify service interfaces and related information and how to invoke services in a distributed system.

Reference Model [ISO Archiving Standards; http://ssdoo.gsfc.nasa.gov/nost/isoas/us04/defn.html]

A reference model is a framework for understanding significant relationships among the entities of some environment, and for the development of consistent standards or specifications supporting that environment. A reference model is based on a small number of unifying concepts and may be used as a basis for education and explaining standards to a non-specialist.

Reliability

Reliability is the ability of a system or component to perform its required functions in routine circumstances, as well as hostile or unexpected circumstances, under stated conditions for a specified period of time.

Resource

Functions (possibly provided through services) or data objects.

Service [ISO 19119:2005; ISO/IEC TR 14252; http://www.opengis.org/docs/02-112.pdf]

Distinct part of the functionality that is provided by an entity through interfaces.

REST

Representational state transfer (REST) is an abstraction of the architecture of the World Wide Web; more precisely, REST is an architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements.

Service [ISO 19119:2005; ISO/IEC TR 14252; http://www.opengis.org/docs/02-112.pdf]

Distinct part of the functionality that is provided by an entity through interfaces.

Session

Temporary association between a subject and a principal as a result of an authentication process initiated by the subject. Information about a session is stored in authentication session information.

SOAP

Simple Object Access protocol is a protocol specification for exchanging structured information in the implementation of web services in computer networks. It uses XML Information Set for its message format, and relies on other application layer protocols, most notably Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.

Software Component [derived from component definition of http://www.opengeospatial.org/resources/?page=glossary]

Software program unit that performs one or more functions and that communicates and interoperates with other components through common interfaces.

Source System

Container of unstructured, semi-structured or structured data and/or a provider of functions in terms of services. The source systems are of very heterogeneous nature and contain information in a variety of types and formats.

Support Service

Service that facilitates the operation of an architecture or system oriented service, e.g. providing an added value by combining the usage of Info-Structure Services.

System [ISO/IEC 10746-2:1996]

Something of interest as a whole or as comprised of parts. Therefore a system may be referred to as an entity. A component of a system may itself be a system, in which case it may be called a sub-system.

Note: For modelling purposes, the concept of system is understood in its general, system theoretic sense. The term "system" can refer to an information processing system but can also be applied more generally.

System User

Provider of services that are used for an application domain as well as IT architects, system developers, integrators and administrators that conceive, develop, deploy and run applications for an application domain.

Terminal

Terminals are a mobile device that is capable of running mobile services and/or mobile applications.

Use case

A common definition of use cases is the one described by Jacobson (Jacobson et al (1995) [34]): "*When a user uses the system, she or he will perform a behaviourally related sequence of transactions in a dialogue with the system. We call such a special sequence a use case*". In Other words, a use case is a textual presentation or a story about the usage of the system told from an end user's perspective.

The use cases provide some tools for people, with different skills (e.g. software developers and non-technology oriented people), to communicate with each other. The use cases are general descriptions of needs or situations that often are related to basic scenarios and that are independent of the technologies and implementations of the underlying system.

User

Human acting in the role of a system user or end user of the service and system.

WADL

The Web Application Description Language is a machine-readable XML description of HTTP-based web applications (typically REST web services) WADL models the resources provided by a service and the relationships between them. WADL is intended to simplify the reuse of web services that are based on the existing HTTP architecture of the Web. It is platform and language independent and aims to promote reuse of applications beyond the basic use in a web browser.

Web Service

Self-contained, self-describing, modular service that can be published, located, and invoked across the Web. A Web service performs functions, which can be anything from simple requests to complicated business processes. Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service.

W3C Web Service [W3C, http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice]

Software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

4 References

The following references are used as background documents for the preparation of this document. References are categorized standards (i.e. standards and specifications from the consortium working groups or alliances and specifications or drafts standardization bodies) and other documents, publications and technical or scientific books.

[1]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.1 *“FIspace Design and Release Plan”*, 2014.
[2]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.2 *“FIspace Technical Architecture and Specification”*, 2014
[3]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.3 *“FIspace Integrated Release V1”*, 2014
[4]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.4 *“FIspace Development Progress Report and V1 Updates”*, *2014*
[5]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.5 *“FIspace Integrated Release V2”*, 2014.
[6]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.6 *“FIspace Development Progress Report and V2 Updates”*, *2014*
[7]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. D200.5 Annex *“FIspace Front-End User Guide”*
[8]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. D200.5 Annex *“FIspace SDK User and Developer Guide”*
[9]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.7 *“FIspace Integrated Release V3”*, 2014.
[10]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. D200.7 Annex *“FIspace Front-End User Guide”*
[11]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. D200.7 Annex *“FIspace SDK User and Developer Guide”*

[12]	FIspace Business collaboration web site. http://www.fispace.eu/
[13]	FIspace Developer Documentation web site. http://dev.fispace.eu/doc/wiki/Home
[14]	FIspace Deliverables web site. http://www.fispace.eu/deliverable.html
[15]	FIspace Tutorial web site. http://www.fispace.eu/tutorials.html
[16]	FIspace Front-End Users Information web site. http://dev.fispace.eu/doc/wiki/gui
[17]	FIspace Front-End User Guide web site. http://dev.fispace.eu/doc/wiki/gui/gui-guide
[18]	FIspace App Developer Intro web site. http://dev.fispace.eu/doc/wiki/App%20Developer%20Intro
[19]	FIspace SDK Guide web site. http://dev.fispace.eu/doc/wiki/sdk
[20]	FIspace apps for newbies web site. https://bitbucket.org/fispace/apps/wiki/FIspace%20apps%20for%20newbies
[21]	FIWARE web site. http://www.fi-ppp.eu/projects/fi-ware/
[22]	FIWARE Catalogue of the Generic Enablers (GEs). http://catalogue.fi-ware.org/
[23]	FIWARE community web site. http://www.fi-ware.org/community/
[24]	FIWARE - Catalogue - Application Mashup – Wirecloud web site. http://catalogue.fi-ware.org/enablers/application-mashup-wirecloud
[25]	FIWARE - Catalogue - Store – Wstore web site. http://catalogue.fi-ware.org/enablers/store-wstore
[26]	Eclipse web site. https://www.eclipse.org/ , https://www.eclipse.org/downloads/
[27]	Maven web site. http://maven.apache.org/ , http://maven.apache.org/download.cgi
[28]	RabbitMQ web site. http://www.rabbitmq.com/
[29]	Express web site. http://expressjs.com/
[30]	ACSI EU-project web site. http://www.acsi-project.eu/
[31]	Proton - CEP recorded webinar and tutorial. http://edu.fi-ware.eu/course/view.php?id=58 Proton - CEP user and programmers guide. http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/CEP_-

	_User_and_Programmer_Guide
[32]	Keycloak JBoss project. http://keycloak.jboss.org/ Keycloak JBoss documentation. http://keycloak.jboss.org/docs
[33]	Powell, D. (Ed.) (1991). Delta-4: A Generic Architecture for Dependable Distributed Computing. Re-search Reports ESPRIT. Project 818/2252 Delta-4 Vol.1. ISBN 3-540-54985-4 Springer-Verlag 1991.
[34]	Jacobson, I., Bylund, S., Jonsson, P., and Ehneboom, S. (1995), “Mod-eling with Use Cases: Using contracts and use cases to build plugable architectures”. Journal of Object Oriented Programming, Vol. 8, No. 2, pp. 18-24.

FIspace

Business Collaboration