

# DEVELOPER DOCUMENTATION

ONLINE DECISION SUPPORT TOOLKIT FOR CLIMATE RESILIENT SEAPORTS

**RMIT University**  
Prepared by: **Guillaume Prevost**  
Date: **30/05/2013**  
ANDS Project Code: **AP35**  
Document Version: **1.0**

<b>ANDS Project Code</b>	AP35
<b>Project Title</b>	Online decision support toolkit for climate resilient seaports
<b>ANDS Program</b>	Applications
<b>Organisation responsible for the project (Subcontractor)</b>	RMIT University
<b>Organisation that will undertake the work (Sub-Subcontractor)</b>	RMIT University
<b>Name of Contact Person</b>	Professor Darryn McEvoy
<b>Address and contact details of Contact Person</b>	Address: GPO Box 2476, Melbourne VIC 3001 Telephone: 03 99251943 Email: <a href="mailto:Darryn.mcevoy@rmit.edu.au">Darryn.mcevoy@rmit.edu.au</a>
<b>Names and affiliations of collaborators if any</b>	Climate Change Adaptation Program and eResearch at RMIT University, Ports Australia, National Transport Commission

## LICENSE

*Copyright (c) 2013, RMIT University, Australia.  
All rights reserved.*

*This project's code and documentation are under the BSD license. See 'license.txt' for details.  
Project hosted at: <https://code.google.com/p/climate-smart-seaports>*

## TABLE OF CONTENTS

License.....	1
Document Revision History.....	3
1 Introduction .....	5
1.1 Project summary .....	5
1.2 Purpose of the document .....	5
1.3 Intended Audience and Reading Suggestions .....	5
1.4 References .....	5
2 Architecture .....	6
2.1 General Architecture.....	6
2.2 Application Actors.....	6
2.3 Workflow.....	9
3 Web Application.....	10
3.1 Project file system.....	10
3.2 Dependencies.....	11
3.1.1 Dependencies list.....	11
3.1.2 Maven .....	12
3.1.3 Jetty.....	13
3.1.4 Spring .....	13
3.1.5 Hibernate .....	13
3.3 Frameworks Configuration .....	13
3.3.1 Maven .....	13
3.3.2 Jetty Configuration.....	15
3.3.3 Spring Configuration .....	15
3.3.4 Hibernate Configuration .....	15
3.4 MVC, Tiles and Views .....	17
3.5 Sitemap .....	19
3.6 Database Architecture .....	20
3.6.1 Database Schema.....	20
3.6.2 Identifiers .....	20

3.6.3	Loading the Database .....	20
3.6.4	Changing the Database .....	21
4	Application .....	21
4.1	Main concepts of the architecture .....	21
4.2	Seaports and Regions.....	22
4.3	Workboards and User Stories .....	23
4.4	User Story Access level .....	24
4.5	Data Elements .....	24
4.6	Data Sources .....	27
4.6.1	ABS Data.....	27
4.6.2	BITRE Data (Ports Australia).....	30
4.6.3	Custom Files .....	33
4.6.4	Past Data (BoM and CSIRO trends) .....	33
4.6.5	ACORN-SAT Data .....	35
4.6.6	CSIRO Data .....	38
4.6.7	CMAR Data .....	41
4.6.8	Engineering Model Data (Concrete deterioration).....	43
4.6.9	Vulnerability Assessment Data .....	48
4.7	Security .....	51
4.7.1	Authentication rules .....	52
4.7.2	User Login Service .....	52
4.7.3	Password Encoder.....	52
5	Publishing to ANDS RIF-CS .....	54
5.1	What is RIF-CS publishing?.....	54
5.2	Formatting for RIC-CS .....	55
5.3	RIF-CS Useful links.....	56
6	Annex 1: Full Database Diagram .....	58
7	Annex 3: Glossary.....	59

## DOCUMENT REVISION HISTORY

Author	Date	Modifications	Version
Guillaume Prevost	22/01/2013	Draft of documentation	0.1
Guillaume Prevost	03/06/2013	Full developer documentation	1.0

## 1 INTRODUCTION

### 1.1 PROJECT SUMMARY

The potential impact of climate change on ports differs according to their location, function and business model of the ports. To be 'Climate Smart', ports in Australia need to understand the relevant climate impacts and risks for their particular operation; only then can they determine what adaptation measures may be appropriate.

The Climate Smart Seaports web-tool is designed primarily for port personnel who make (or influence) decisions around long-term port planning for infrastructure, assets and management systems. However, it will also be of value to port owners and related businesses, government departments and local authorities concerned with ports and infrastructure; and for application by academic researchers.

The Climate Smart Seaports Tool enables interested users to begin the process of a climate risk assessment. It assists them to identify current and historical climate trends and variability, as well as future climate projections under a variety of scenarios.

Population and trade data is included, and users can add port-specific information to round out their analysis.

### 1.2 PURPOSE OF THE DOCUMENT

The purpose of this document is to detail the technical choices, architecture and implementation of the Climate Smart Seaports application, as well as elements about its configuration. This document provides useful insights about the application code, to get started on working on Climate smart Seaports.

### 1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

This document is aimed to developers working – or willing to start working – on the project. Although it is recommended for a new developer to read this document's sections in order, one can also look for an explanation for a specific part of the application.

### 1.4 REFERENCES

For documentation about installation of the development environment and deployment of the project, consult the *Developer Installation Guide* document.

## 2 ARCHITECTURE

### 2.1 GENERAL ARCHITECTURE

This section details the general architecture and purpose of the Climate Smart Seaports application.

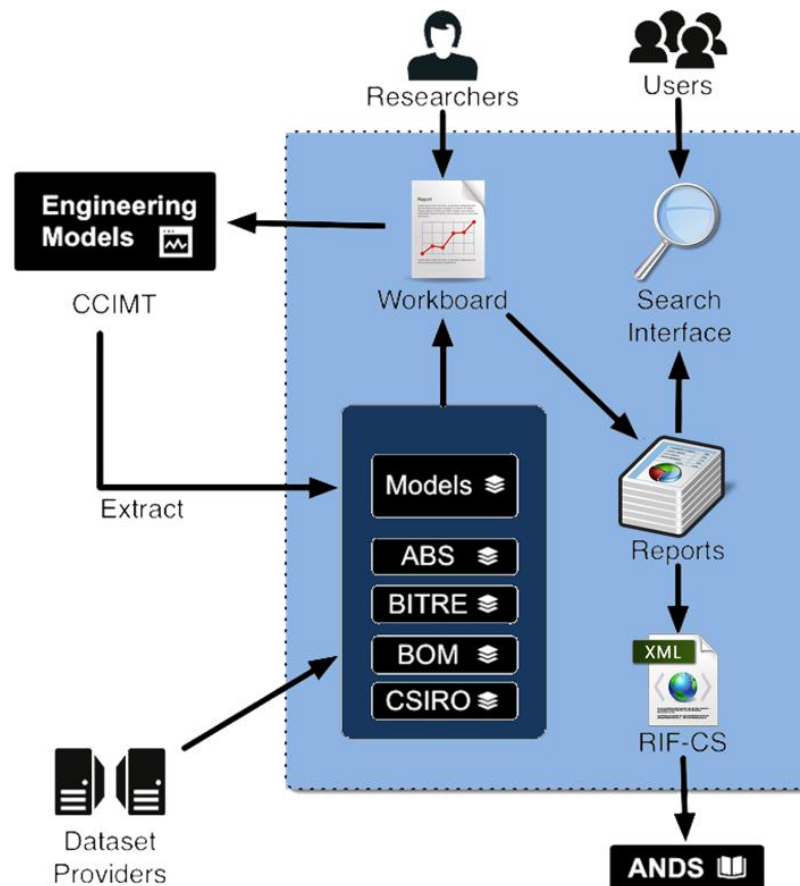


Figure 1: Climate Smart Seaports General architecture diagram

Researchers can use the application to create a “workboard” and add data originating from various sources. They can also work with an external engineering tool which generates more customized data and import this data to their workboard. After gathering the piece of data they need, researchers can then create reports and publish them, both on the CSS portal to make them accessible and searchable by all and to Research Data Australia, a service from ANDS which facilitates search and reuse of research data.

### 2.2 APPLICATION ACTORS

This section details the different entities interacting with the Seaports application and the way they do so.

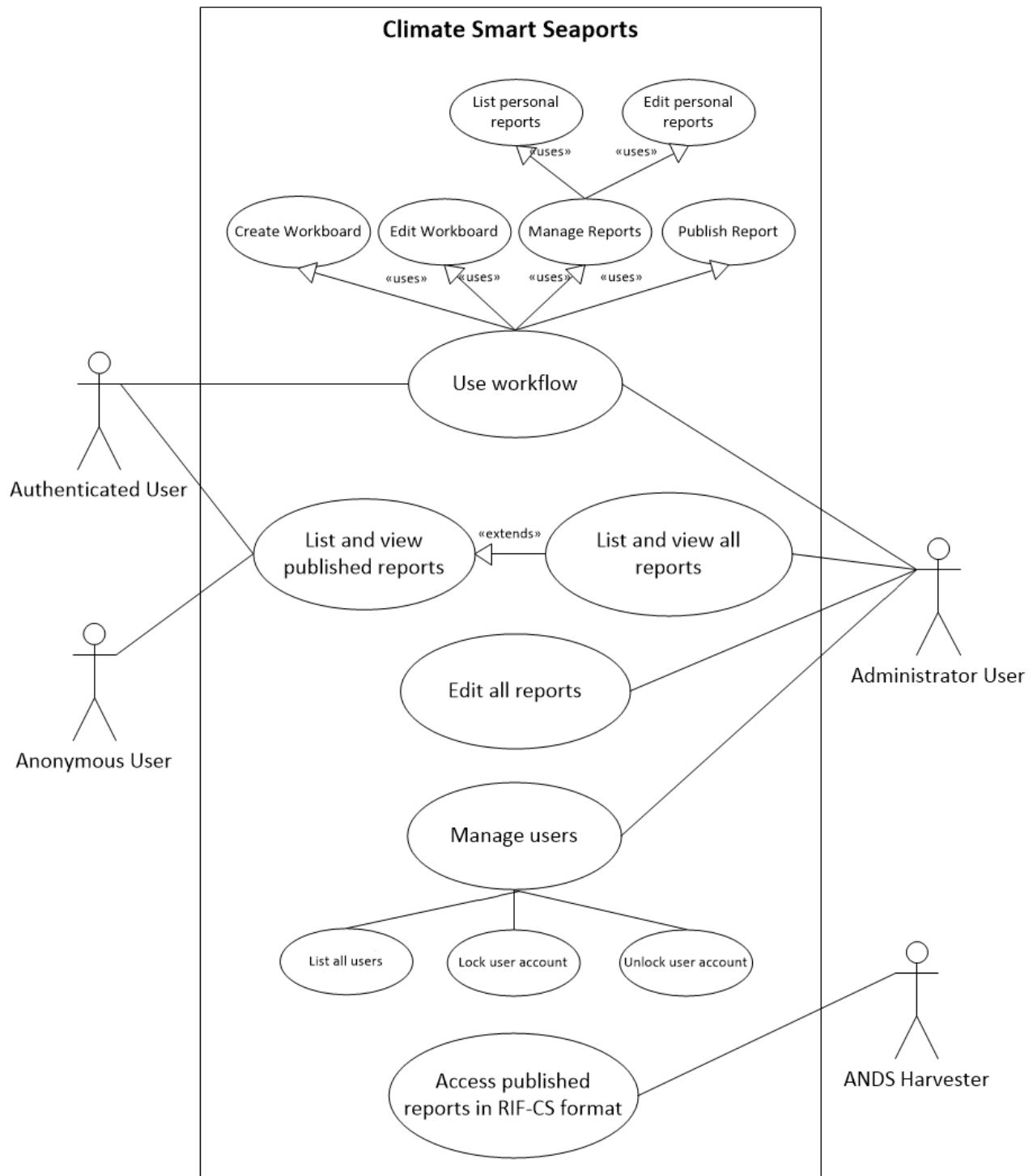


Figure 2: Use Case diagram of the Climate Smart Seaport application



**Anonymous users** can only access the home page, browse and view the reports published by users.

**Authenticated users**, also referred as “users” can do all what the anonymous users can do, but can also use the workflow of the application (create a workboard, add data to a Workboard, list and edit their personal reports, and publish reports).

**Administrator users** have the ability to do all of the above plus browse, view and edit workboard and reports of any other user. They can also manage users (list all users, lock and unlock user accounts).

The **ANDS Harvester** isn’t an actual user, but software interacting with the CSS application to collect RIF-CS records. It can only access the published reports under the RIF-CS format. (see section 5 “Publishing to ANDS RIF-CS” for more details).

## 2.3 WORKFLOW

This section aims to describe the workflow of the application from a user point of view. This should help getting familiar with the both the application concepts and the Sitemap.

There are four main stages for the creation of a report, each composed of up to four steps (see Figure 3: User Workflow).

### Climate Smart Seaports Tool - User Workflow

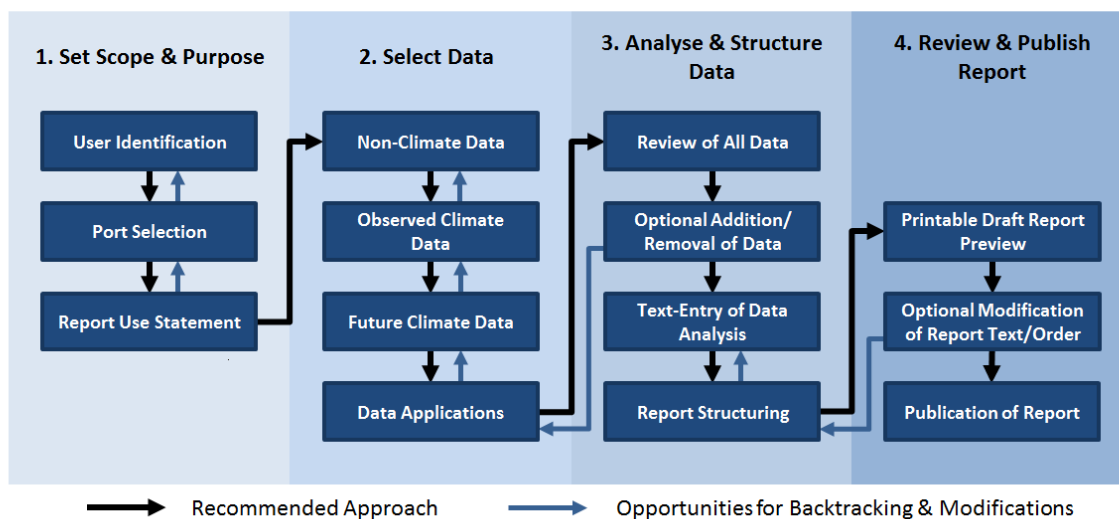


Figure 3: User Workflow

Once logged in, the user the first stage consists in setting the scope and purpose of the report that is going to be created: this is achieved in the “*Workboard creation*” page.

After creating the Workboard, the user can add several types of data grouped into 4 categories: non climate data, observed climate data, future climate data, and data applications (see 4.6 Data Sources for details about each type of data).

The next stage consists in analyzing and structuring the data gathered in the previous stage. Before leaving the “*Workboard*” page, the user can review the all data at once, and then continue to the “*Report Edition*” page. There, it is still possible to decide to exclude some data that were added, or to re-include them into the final report. It’s also possible to add and edit some analysis text about the data, and re-order the data and texts. At any time, the user can preview the report to see how the final version will look like.

When the report is ready, the user can decide to publish it (see 5 Publishing to ANDS RIF-CS).

### 3 WEB APPLICATION

This section details the Climate Smart Seaports web application. It helps understanding the project file system, configuration files, the main frameworks used, and the sitemap.

#### 3.1 PROJECT FILE SYSTEM

- src/main/java
      - database
      - helpers
      - security
      - war.controller
      - war.dao
      - war.model
    - src/main/resources
    - src/test/java
      - war.controller
      - war.dao
    - src/test/resources
    - Maven Dependencies
    - JRE System Library [JavaSE-1.7]
    - bin
    - libs
    - src
      - main
        - webapp
          - META-INF
          - resources
            - css
            - docs
            - img
            - js
          - WEB-INF
            - spring
            - views
              - jetty-web.xml 606 22/03/13 14:23 guillaume
              - tiles.xml 665 20/05/13 17:11 guillaume
              - web.xml 606 22/03/13 14:23 guillaume
              - index.html 590 04/03/13 12:47 guillaume
        - test
    - target
      - jetty-env.xml 605 21/03/13 14:56 guillaume
      - pom.xml 661 17/05/13 15:27 guillaume

The seaports project is organized as shown on the picture on the left:

**src/main/java:** this is where the main JAVA classes are located, and most of the source code belongs here.

**src/main/resources:** hibernate and log4j configuration files for the main classes.

**src/test/java:** location of the unit tests classes. The unit tests are in a package corresponding to the name of the classes they test.

**src/test/resources:** hibernate and log4j configuration files for the test classes.

**bin:** compiled files. nothing to see here

**libs:** local project's libraries.

**src/main/webapp/resources:** the web application resources: pictures, documents, Javascript and CSS files.

**src/main/webapp/WEB-INF:** the web application and Spring framework configuration files are located here.

**src/main/webapp/WEB-INF/views:** the JSP files responsible of the layout and display of the page.

**target:** compiled files. nothing to see here

**src/pom.xml:** Maven file containing the project's dependencies.

## 3.2 DEPENDENCIES

### 3.1.1 DEPENDENCIES LIST

The following diagram shows the different dependencies that have been used in Climate Smart Seaports.

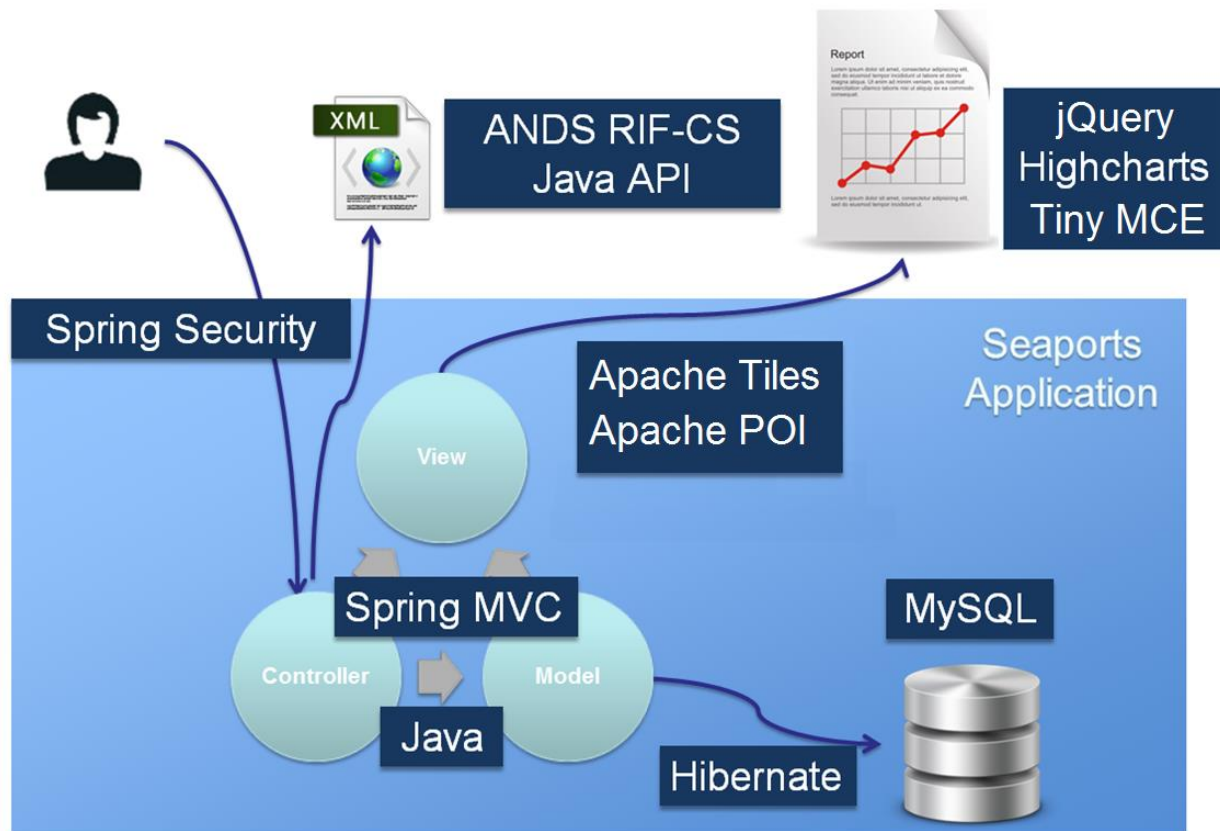


Figure 4: Technologies used in Climate Smart Seaports application

Java and MySQL are mentioned in the above diagram (Figure 4) only to give a picture of the programming language and database language used in the project.

The following tables detail the rest of these dependencies' websites and licenses:

### Java dependencies

Dependency	URL	License	License URL
<b>Spring</b>	<a href="http://www.springframework.org/">http://www.springframework.org/</a>	Apache 2.0 license	<a href="http://www.apache.org/licenses/LICENSE-2.0.html">http://www.apache.org/licenses/LICENSE-2.0.html</a>
<b>Spring Security</b>	<a href="http://www.springframework.org/spring-security">http://www.springframework.org/spring-security</a>	Apache 2.0 license	<a href="http://www.apache.org/licenses/LICENSE-2.0.html">http://www.apache.org/licenses/LICENSE-2.0.html</a>
<b>Hibernate</b>	<a href="http://www.hibernate.org/g/license">http://www.hibernate.org/g/license</a>	LGPL v2.1	<a href="http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html">http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html</a>
<b>Apache Tiles</b>	<a href="http://tiles.apache.org/">http://tiles.apache.org/</a>	Apache 2.0 license	<a href="http://www.apache.org/licenses/LICENSE-2.0.html">http://www.apache.org/licenses/LICENSE-2.0.html</a>
<b>Apache POI</b>	<a href="http://poi.apache.org/">http://poi.apache.org/</a>	Apache 2.0 license	<a href="http://www.apache.org/licenses/LICENSE-2.0.html">http://www.apache.org/licenses/LICENSE-2.0.html</a>
<b>RIF-CS Java API</b>	<a href="http://ands.org.au/resource/downloads.html">http://ands.org.au/resource/downloads.html</a>	CC Australia Attribution 3.0 Licence	<a href="http://creativecommons.org/licenses/by/3.0/au/deed.en">http://creativecommons.org/licenses/by/3.0/au/deed.en</a>

### JavaScript dependencies:

Dependency	URL	License	License URL
<b>jQuery</b>	<a href="http://jquery.com">http://jquery.com</a>	MIT License	<a href="http://opensource.org/licenses/MIT">http://opensource.org/licenses/MIT</a>
<b>jQuery Zebra Tooltips plugin</b>	<a href="http://stefangabos.ro/jquery/zebra-tooltips">http://stefangabos.ro/jquery/zebra-tooltips</a>	LGPL v3.0	<a href="http://www.gnu.org/licenses/lgpl-3.0.txt">http://www.gnu.org/licenses/lgpl-3.0.txt</a>
<b>jQuery MapHighLights plugin</b>	<a href="https://github.com/kemayo/maphilight">https://github.com/kemayo/maphilight</a>	MIT License	<a href="http://davidlynch.org/projects/maphilight/MIT-LICENSE.txt">http://davidlynch.org/projects/maphilight/MIT-LICENSE.txt</a>
<b>jQuery datatables plugin</b>	<a href="http://www.datatables.net/">http://www.datatables.net/</a>	LGPL v2	<a href="http://www.datatables.net/license_gpl2">http://www.datatables.net/license_gpl2</a>
<b>Highcharts</b>	<a href="http://www.highcharts.com/">http://www.highcharts.com/</a>	CC BY-NC 3.0	<a href="http://creativecommons.org/licenses/by-nc/3.0/">http://creativecommons.org/licenses/by-nc/3.0/</a>
<b>TinyMCE</b>	<a href="http://www.tinymce.com/">http://www.tinymce.com/</a>	LGPL	<a href="http://www.gnu.org/licenses/gpl.html">http://www.gnu.org/licenses/gpl.html</a>

#### 3.1.2 MAVEN

Maven is used to take care of the project's dependencies and deployment. It is based on a POM (project object model) file which is located at the root of the project's file system. See the Frameworks Configuration

section below for more details.

Maven home page: <http://maven.apache.org/>

### 3.1.3 JETTY

---

Jetty provides a Web Server and *javax.servlet* container.

Jetty home page: <http://www.eclipse.org/jetty/>

Jetty documentation: <http://www.eclipse.org/jetty/documentation/current/>

### 3.1.4 SPRING

---

The Climate Smart Seaports project uses Spring Model-View-Controller (Spring MVC) as its main architecture. This separates the data access, the application logic and the display. Spring annotations allow identifying some classes as controllers which handle clients' requests

Spring Framework documentation: <http://www.springsource.org/documentation>

Spring Framework tutorials: <http://www.springsource.org/tutorials>

### 3.1.5 HIBERNATE

---

Hibernate is a persistence framework, which allows a complete abstraction of the database concepts and let programming by dealing with objects only. Hibernate annotations are used to specify which properties of the *model* classes should be saved in the database. The *data access objects* (DAO) are responsible of saving the model objects.

Hibernate documentation: <http://www.hibernate.org/docs>

Hibernate tutorials: <http://docs.jboss.org/hibernate/orm/4.2/quickstart/en-US/html/>

## 3.3 FRAMEWORKS CONFIGURATION

### 3.3.1 MAVEN

---

The Maven project object model, or POM file (*pom.xml*) holds the references to all the dependencies and to the repositories where to find them. At the top of the file are also specified the application's name and version, and in which package the java classes are located:

```
<packaging>war</packaging>
<version>1.0.0</version>
<name>Climate Smart Seaports</name>
```

Here, the application Climate Smart Seaports in its version 1.0.0 specifies that the classes are located in the "war" package.

After this, a list of all the project's dependencies (the libraries used by the Climate Smart Seaports application) follows. When listed here, Maven will automatically download them and add them to the project when invoking the command *"mvn install"* (see the *Developer Installation Guide* document for more details).

Almost all the dependencies used by the application are standard, and can be found on online repositories which are specified at the bottom of the POM file.

The only exception is the **RIF-CS API from ANDS**. Because of this, the JAR file for this library is contained within the project, under the *libs* folder. This is the code added to the POM file to reference this local library:

```
<dependencies>

    . . .

    <dependency>
        <groupId>ands</groupId>
        <artifactId>rifcs-api</artifactId>
        <version>1.3.0</version>
    </dependency>
</dependencies>

<repositories>
    <repository>
        <id>libs</id>
        <releases>
            <enabled>true</enabled>
            <checksumPolicy>ignore</checksumPolicy>
        </releases>
        <snapshots>
            <enabled>false</enabled>
        </snapshots>
        <url>file://${project.basedir}/libs</url>
    </repository>

    . . .
</repositories>
```

Maven searches the library at the following path:

*[project repository]/[groupid]/[artifactId]/[version]/[artifactId]-[version].jar*

Based on the above configuration, the following path will be constructed:

*file://\${project.basedir}/libs/ands/rifcs-api/1.3.0/rifcs-api-1.3.0.jar*

Where *\${project.basedir}* is the location of the project in the file system.

This corresponds to the existing file system's tree in the project:

```
└─ libs
  └─ ands
    └─ rifcs-api
      └─ 1.3.0
        └─ rifcs-api-1.3.0.jar
```

### 3.3.2 JETTY CONFIGURATION

---

```
<Configure id="webAppCtx" class="org.mortbay.jetty.webapp.WebAppContext">
<New id="beanManager" class="org.mortbay.jetty.plus.naming.Resource">
  <Arg><Ref id="webAppCtx" /></Arg>
  <Arg>BeanManager</Arg>
  <Arg>
    <New class="javax.naming.Reference">
      <Arg>javax.enterprise.inject.spi.BeanManager</Arg>
      <Arg>org.jboss.weld.resources.ManagerObjectFactory</Arg>
      <Arg />
    </New>
  </Arg>
</New>
</Configure>
```

```
<Configure id="WebAppContext" class="org.eclipse.jetty.webapp.WebAppContext">
  <Set name="maxFormContentSize" type="int">600000000</Set>
</Configure>
```

For more information: [http://wiki.eclipse.org/Jetty/Howto/Configure\\_Jetty](http://wiki.eclipse.org/Jetty/Howto/Configure_Jetty)

### 3.3.3 SPRING CONFIGURATION

---

#### Spring config

### 3.3.4 HIBERNATE CONFIGURATION

---

There are several files for the configuration of the database. The Spring configuration file **root-context.xml** (*src/main/webapp/WEB-INF/spring/root-context.xml*) contains the following line to specifies that a specific file should be used.

*extract of root-context.xml (src/main/webapp/WEB-INF/spring/root-context.xml)*

```
...
<import resource="db.xml" />
...
```



**db.xml** defines in which file the database credentials and settings are located (*db.properties*), sets Hibernate's entity manager factory settings, and sets the Hibernate's data source credentials based on the credentials found in *db.properties*. It also sets the JPA transaction manager.

extract of *db.xml* (*src/main/webapp/WEB-INF/spring/db.xml*):

```
...
<bean id="placeholderConfig"
class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
  <property name="location" value="classpath:db.properties" />
</bean>

<bean id="entityManagerFactory"
class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
  <property name="jpaVendorAdapter">
    <bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
      <property name="showSql" value="true" />
      <property name="generateDdl" value="true" />
      <property name="databasePlatform" value="${db.dialect}" />
    </bean>
  </property>
</bean>

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-
method="close">
  <property name="driverClassName" value="${db.driver}" />
  <property name="url" value="${db.url}" />
  <property name="username" value="${db.username}" />
  <property name="password" value="${db.password}" />
</bean>

<bean id="transactionManager"
class="org.springframework.orm.jpa.JpaTransactionManager">
</bean>

<bean
class="org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor" />
...
```

The database credentials are in the **db.properties** files (*/src/main/resources/db.properties* for the application database, and */src/test/resources/db.properties* for the test database).

*db.properties* (*src/main/resources/db.properties*):

```
#MySQL Options
db.password=YOUR_DATABASE_PASSWORD
db.username=YOUR_DATABASE_USER
db.url=jdbc:mysql://YOUR_SERVER:3306/YOUR_DATABASE_NAME
db.dialect=org.hibernate.dialect.MySQL5Dialect
db.driver=com.mysql.jdbc.Driver
```

Some additional Hibernate-specific settings are located in the file **persistence.xml**.

extract of '*src/main/resources/META-INF/persistence.xml*':

```
...
<persistence-unit name="acme" transaction-type="RESOURCE_LOCAL">
```

```
<properties>
<property name="hibernate.hbm2ddl.auto" value="create-drop" />
<property name="hibernate.show_sql" value="true" />
<property name="hibernate.transaction.flush_before_completion" value="true" />
<property name="hibernate.cache.provider_class"
value="org.hibernate.cache.HashtableCacheProvider" />
<property name="hibernate.ejb.naming_strategy"
value="org.hibernate.cfg.ImprovedNamingStrategy" />
</properties>
</persistence-unit>
...
```

### 3.4 MVC, TILES AND VIEWS

The web application is organized with around Spring MVC. The typical way a client's request is handled is as follows:

1. The client's request is intercepted by the Spring MVC framework's servlet.
2. The servlet performs the security and authentication checks (see 4.7 Security).
3. The servlet consults the mapping between requested addresses and controllers' methods to handle them (this mapping is defined using Spring annotations above each of the controllers' methods). It redirects the request to the right controller's method.
4. The controller method retrieves, saves, and edits the required data using the relevant Data Access Objects (DAO), and performs any other relevant action.
5. The controller returns either a *String* or a *ModelAndView* object.
6. The servlet handles that return value and transmits it to the Spring *Tiles* plugin. The return value from the controller is retrieved, and a specific view is selected based on that value and the configuration files *tiles.xml* file (*src/main/webapp/WB-INF/tiles.xml*). See the below for a concrete example.

Let's take the example of a client's request at the root of the application "/" corresponding to the URL "serveraddress:8080/CSS/". The controller *PublicController.java* specifies that it handles this request in the `@RequestMapping` annotation (see the code extract below).

*extract of src/main/java/war/controllers/PublicControllers.java:*

```
...
@RequestMapping(value = {"/", "/public"}, method = RequestMethod.GET)
public ModelAndView home(Model model) {
    tryGetLoggedInUser(model);
    return new ModelAndView("home");
}
...
```

After doing the necessary operations (not much on the home page, but usually a lot more in the "WorkboardController" and "UserStoryController"), the method returns a *ModelAndView* object containing the view name: "home".

The Tiles configuration file has a definition for the “*home*” view. This view inherits from the “*public.base.definition*” view. This defines the view file “*layout.jsp*” as the file to use, and also defines several attributes that are going to be used in the view.

*extract of src/main/webapp/WEB-INF/tiles.xml:*

```
...
<definition name="public.base.definition" template="/WEB-INF/views/layout.jsp">
  <put-attribute name="title" value="Seaport Project" />
  <put-attribute name="imports" value="/WEB-INF/views/imports.jsp" />
  <put-attribute name="header" value="/WEB-INF/views/header.jsp" />
  <put-attribute name="menu" value="/WEB-INF/views/menu.jsp" />
  <put-attribute name="userInfo" value="/WEB-INF/views/userInfo.jsp" />
  <put-attribute name="footer" value="/WEB-INF/views/footer.jsp" />
</definition>
...
<definition name="home" extends="public.base.definition">
  <put-attribute name="title" value="Home - Climate Smart Seaports" />
  <put-attribute name="body" value="/WEB-INF/views/home.jsp" />
</definition>
...
```

In “*layout.jsp*”, the different attributes defined in “*tiles.xml*” are used. For our example, the “*body*” attribute has been set to “*/WEB-INF/views/home.jsp*”.

*extract of src/main/webapp/WEB-INF/views/layout.jsp:*

```
...
<div id="page-body" style="min-height:500px">
  <tiles:insertAttribute name="body" ignore="true" />
</div>
...
```

The file “*home.jsp*” will be included in place of the line `<tiles:insertAttribute name="body" ignore="true" />`.

The same principles are applied all across the application for every client’s request that the application handles. The controllers can use call a method from the *Model* object passed to it as a parameter in order to add attributes that can be used in the views:

*example of addition of an attribute in the controller (Java file):*

```
model.addAttribute("userProfile", user);
```

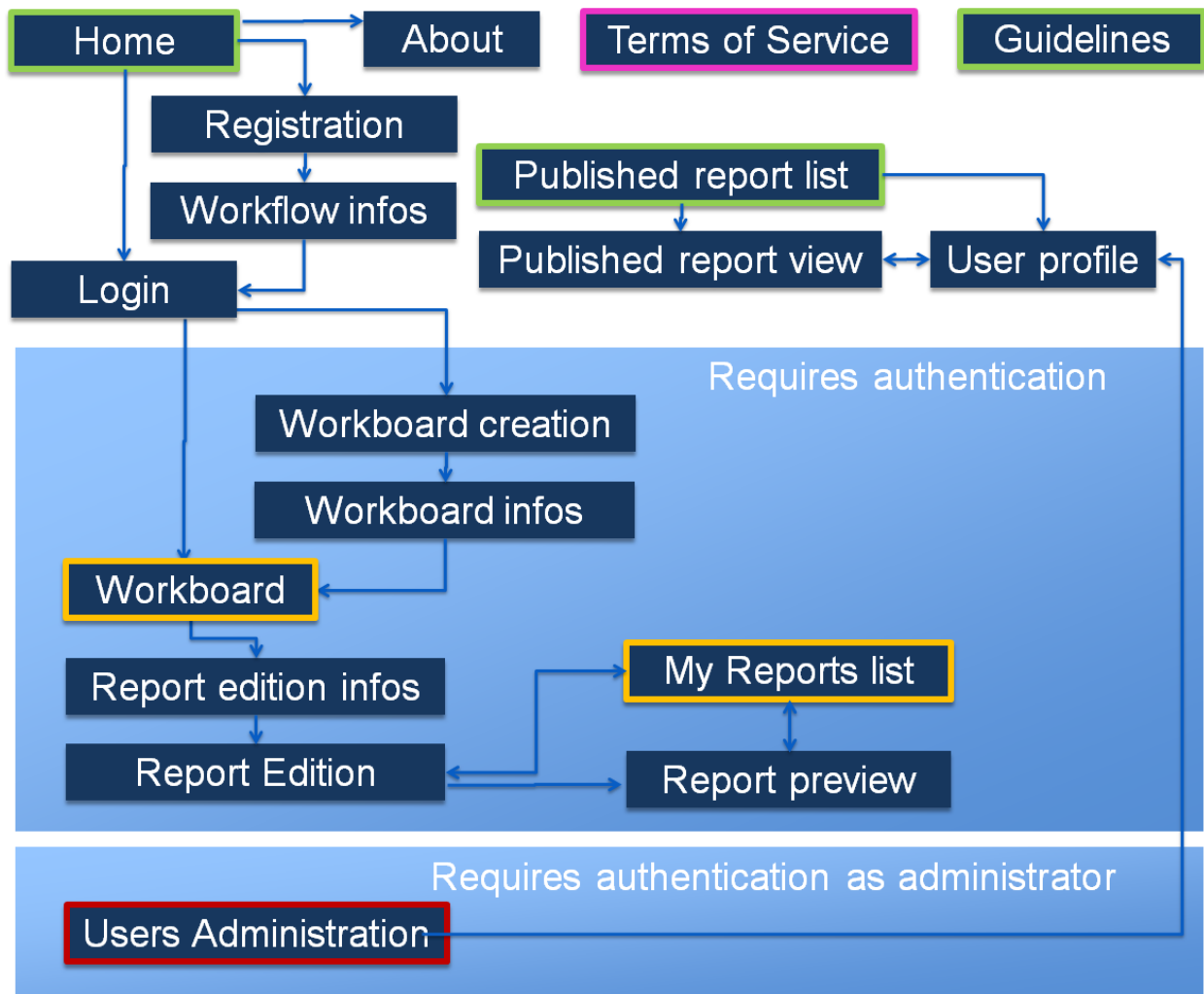
*example of the use of an attribute in the view (JSP file):*

```
...
<c:if test="${not empty userProfile}">
  <h2>${userProfile.firstname} ${userProfile.lastname}'s profile</h2>
  ...
</c:if>
...
```

### 3.5 SITEMAP

The following sitemap shows the relationships between pages, and which page requires authentication or specific rights. The outline around some pages represents their accessibility through the main menu. Note that it is incremental: all main menu items available for public users are still available for authenticated users, and all main menu items available for authenticated users are still available for administrator users.

Figure 5: Sitemap of Climate Smart Seaports application



## 3.6 DATABASE ARCHITECTURE

### 3.6.1 DATABASE SCHEMA

---

The full database schema diagram can be found in the Annex 1: Full Database Diagram.

### 3.6.2 IDENTIFIERS

---

Except for a few exceptions, all tables have a unique identifier column “*id*” of type Integer which contains automatically generated identifier. The exceptions are the following tables:

- The table *user* has a “*username*” column of type String which contains the unique username used by the user to log in.
- The table “*acorn\_sat\_station*” has a “*number*” column of type *Double* which holds the unique ACORN-SAT station number set by Bureau of Meteorology.
- The table “*seaports*” has a “*code*” column of type *String* representing the unique international code of the seaport.

### 3.6.3 LOADING THE DATABASE

---

There is a Java package called “*database*” under *src/main/java/database* which contains several classes used to create the various data sets for the initial load of the database.

The advantage of using this method is that Hibernate annotations are used to automatically create the database schema corresponding to the model.

The two principal classes in this package are: *DatabaseLoader* and *TestDatabaseLoader*. They each contain a “*main*” method which can be run within Eclipse (Right click on the class, “*Run As...*”, “*Java Application*”). They respectively load the application database and the test database. These classes respectively use *hibernate.xml* and *hibernate-test.xml* configuration files to define the database credentials.

The other classes of this package contain methods which each load data for a specific dataset. They are used by the two principal classes to load all the data, but they can also be run independently using their “*main*” methods.

The Engineering Model example data (see 4.6.8 Engineering Model Data (Concrete deterioration)) can’t be automatically loaded in Java like the other datasets since they come from Excel files. To load them, there is an SQL file “*seaports\_2regions\_engineering\_examples.sql*” that needs to be loaded directly using SQL in the database after its creation.

SQL dumps of the database initially loaded with datasets but empty of any user data are also available for the application database (*seaports\_dump.sql*) and for the test database (*seaports\_test\_dump.sql*).

#### 3.6.4 CHANGING THE DATABASE

---

If the Java classes in the model are modified, the database has to be updated in consequence. Minor changes can eventually be performed directly using SQL, which avoids having to reset the data.

However, for major changes, it is strongly recommended to re-create an empty database and re-run the main methods of the *DatabaseLoader* and *TestDatabaseLoader*, in order to re-create a schema matching perfectly the object model.

The steps to follow:

1. Modify the model objects (Hibernate annotations change, name change, etc.)
2. Delete the database and re-create it (SQL: *DROP SCHEMA seaports; CREATE SCHEMA seaports;*).
3. Run the main method of the class *DatabaseLoader*.

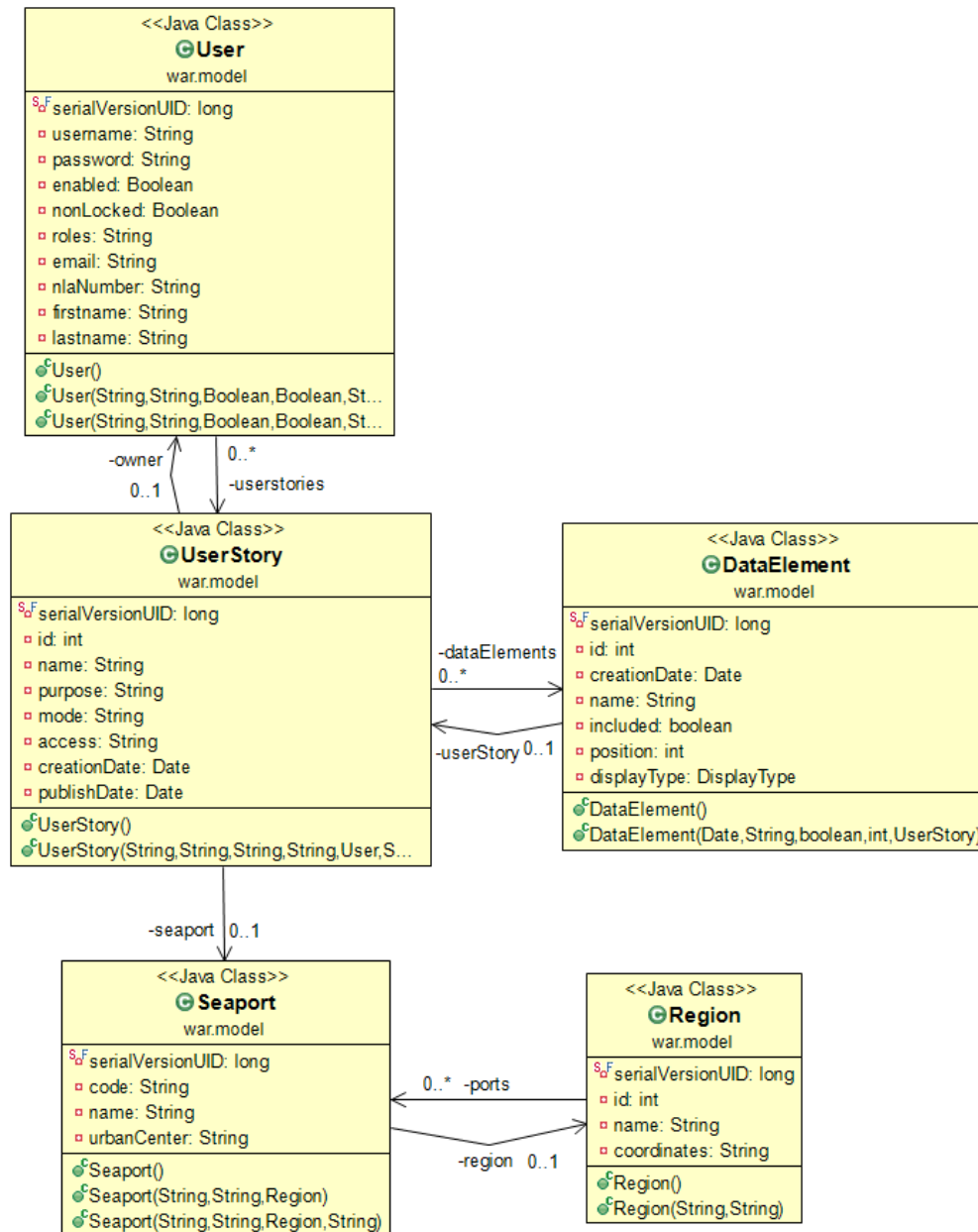
After performing changes in the database, don't forget to replace the SQL dumps files "*seaports\_dump.sql*" and "*seaports\_test\_dump.sql*" after reloading initial data (this can be done using MySQL).

## 4 APPLICATION

### 4.1 MAIN CONCEPTS OF THE ARCHITECTURE

To make it simple, the basics is that each user of the CSS application can have several user stories, and each user story can contain several data elements.

The Data Elements can eventually be related to datasets held in the database as well, which will be detailed later (see 4.6 Data Sources). A User Story is related to a particular Seaport, which belongs to a Region. A Region contains many Seaports.



## 4.2 SEAPORTS AND REGIONS

The regions correspond to the NRM (Natural Resource Management) regions defined for Australia. For this proof of concept of the application, only 3 of these regions have been integrated. However, the architecture was conceived to make it easy to add new regions and seaports easily later on.

The following table shows the region and which seaports they contain:

East Coast South	Southern Slopes Vic East	Southern and Southwestern Flatlands West
------------------	--------------------------	--

Port of Yamba	Port Kembla	Albany Port
Port of Newcastle	Port of Eden	Port of Bunbury
Sydney Ports	Port of Mallacoota	Esperance Ports
	Lakes Entrance	Fremantle Ports
	Port Welshpool	Port of Geraldton

#### 4.3 WORKBOARDS AND USER STORIES

Each user can have many user stories. The user stories can be in 3 different modes: “active”, “passive” and “published”.

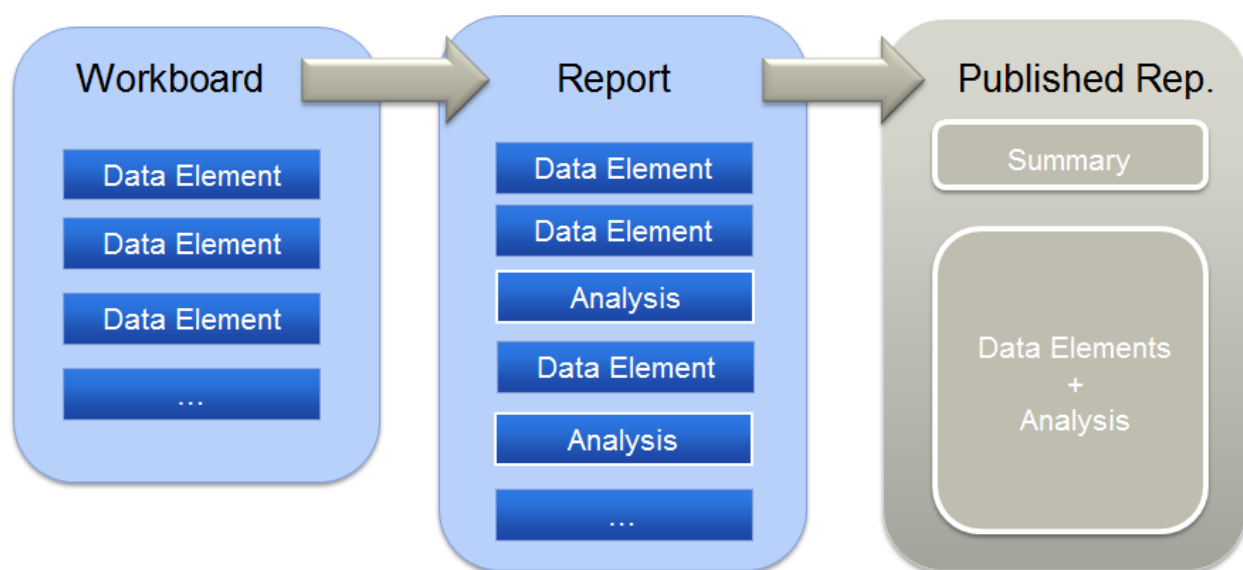


Figure 6: Simple diagram of the application workflow

**Active:** an active user story is called the **workboard**. Each user can have only one workboard at a time. Technically, the workboard of a user is stored as a User Story in the database, with the “mode” property set to “active”.

**Passive:** once a user is done adding data elements to his workboard, he can create a user story based on it. From that point on, the “mode” property is set to “passive”. From the user point of view, **his workboard becomes a report**. He can work on organizing and writing text in the story, or he can create a new workboard.

**Published:** When a user decides a report is complete and ready to be officially published, he can publish it to ANDS on Research Data Australia. Once the user decides to publish a report, the User Story “mode” property is set to “published”. **Once a report is published, a user can’t edit it, delete it, or set it to private access** (cf. 4.4 User Story Access level).



#### 4.4 USER STORY ACCESS LEVEL

**Private:** a User Story with the “access” property set to “private” is not listed in the results of the researches, and can only be viewed by the owner of this User Story. He can access it via the page “My Stories”. A Workboard is always private (because it is not a finished user story).

**Public:** A user can decide to set any of his stories as public. The “access” property of the User Story is set to “public”. Public User Stories are listed in the search results and can be viewed by any user, even not logged in. When publishing a User Story, it automatically makes it public, but a story can be made public on the Seaports portal but not published to ANDS.

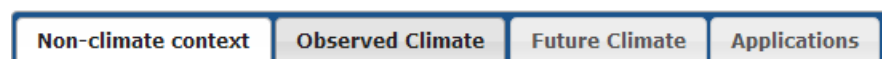
#### 4.5 DATA ELEMENTS

**A User Story is composed of data elements.** A data element is a piece of data that can originate from many different sources. It can be extracted from the application database (example: CSIRO data), be uploaded by the user as a custom file, etc.

Most of the data elements are added during the “**Workboard**” stage (see Workflow). When it becomes passive, the user can only add hand written text to add his/her analysis to the report. What the application technically does at this stage is adding more data elements to the User Story, but the end user sees it as redacting text comment.

Each data element content can be either populated by the user or correspond to a subset of datasets pre-existing in the database.

During the “Workboard” stage (see Workflow), the **data elements are separated into 4 logical categories** appearing as 4 tabs in the user interface. This is to help the user following the workflow, but it isn’t reflected in the database:



The following table is a summary of all the data element types, their corresponding name in the user interface, and their categories:

Logical category	Display name	Data Element subclass	Linked to dataset
Non-climate context	ABS Data	DataElementABS	X
	Ports Australia Data	BitreDataElement	X
	Custom data	DataElementFile	
Observed climate	Past Data	DataElementPast	X
	ACORN-SAT data	DataElementAcornSat	X

Future climate	CSIRO data	DataElementCsiro	X
	CMAR data	DataElementCmar	X
Applications	Concrete deterioration Model	DataElementEngineeringModel	(X) <sup>1</sup>
	Vulnerability Assessment	DataElementVulnerability	

---

<sup>1</sup> The Concrete Deterioration Model has a (X) because it is supposed to be composed of data uploaded by the user, but the application also offers to use pre-defined examples stored in a dataset. See 4.6.8 Engineering Model Data (Concrete deterioration) for details.

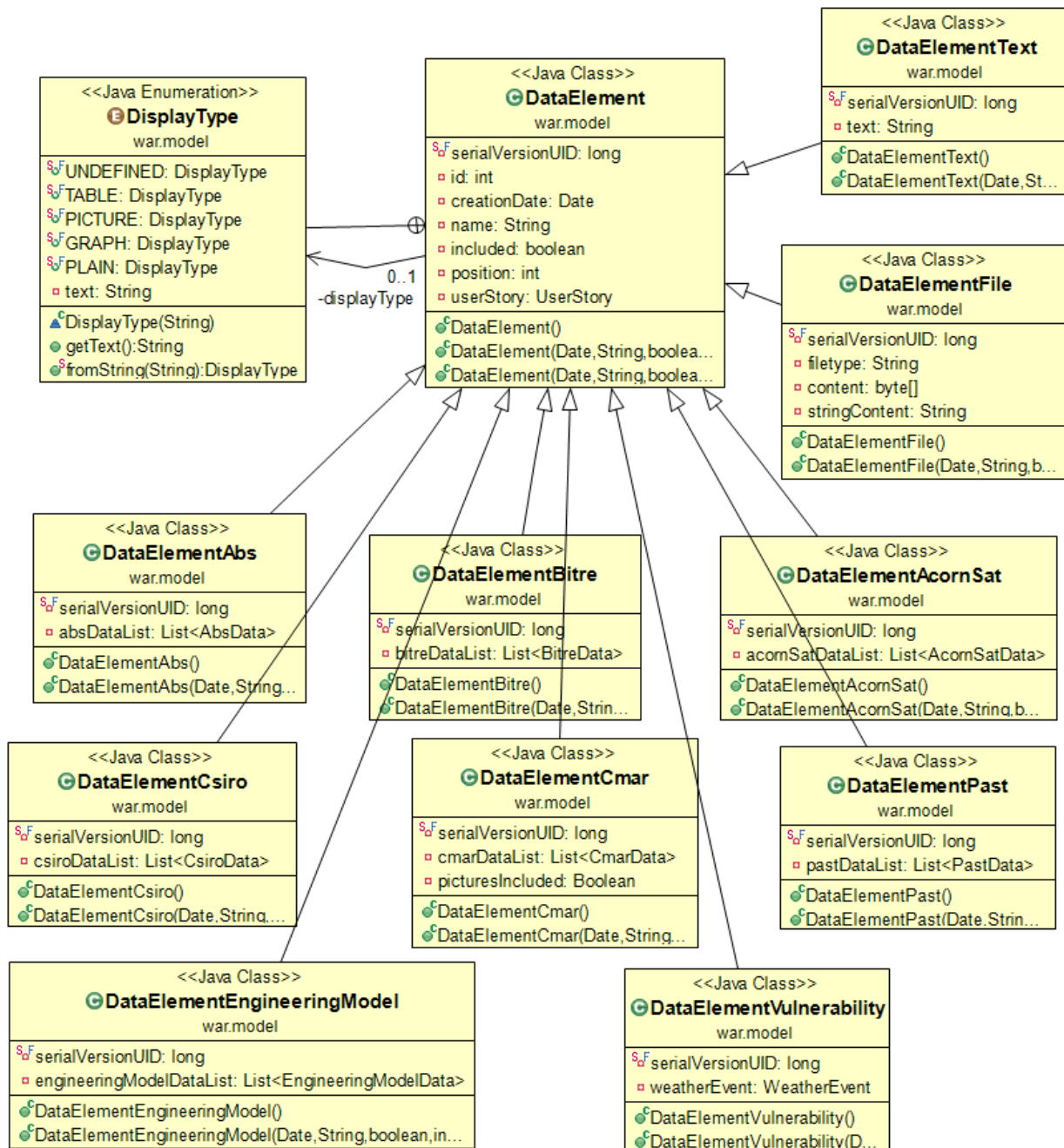


Figure 7: Data Element class's inheritance

The diagram below shows the inheritance of data elements. A class **DataElement** holds all the properties common to every data element: a unique ID, the date and time when the data element was created, the name of the data element, the parent user story containing the data element, whether the data element is included in its parent, and its position within its parent.

The *displayType* property is selected by the user upon creation of the data element, and only used in to display the same data in different ways in the views.

Each specific type of data element has a corresponding class which inherits all these properties, and adds new properties specific to its type. Most of them provide a list of data of a specific type: this is the way a data element references a subset of a dataset.

User selects variable "green"  
and other variable "triangle".

The Data Element references a list of  
all the data matching "green triangle".

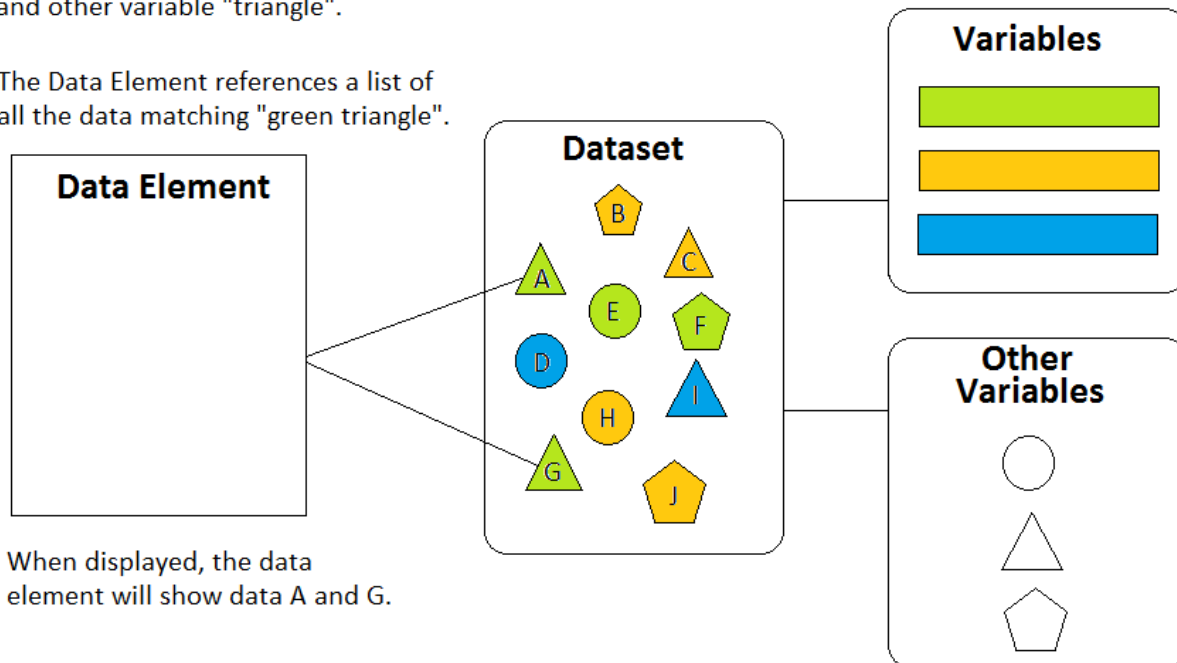


Figure 8: simple schema of how data elements reference datasets

## 4.6 DATA SOURCES

This section presents the different data sources available for the users, and explains how they are modeled and the important details to know in order to work on them or add new sources.

### 4.6.1 ABS DATA

The ABS data represents data from the Australian Bureau of Statistics. The data is related to one seaport which has a urban center nearby (example: Port Kembla and Wollongong), and to a measured variable. So far, data for only one variable "Population Change" has been added in the Seaports Application. The following diagram shows how this is modeled in term of classes:

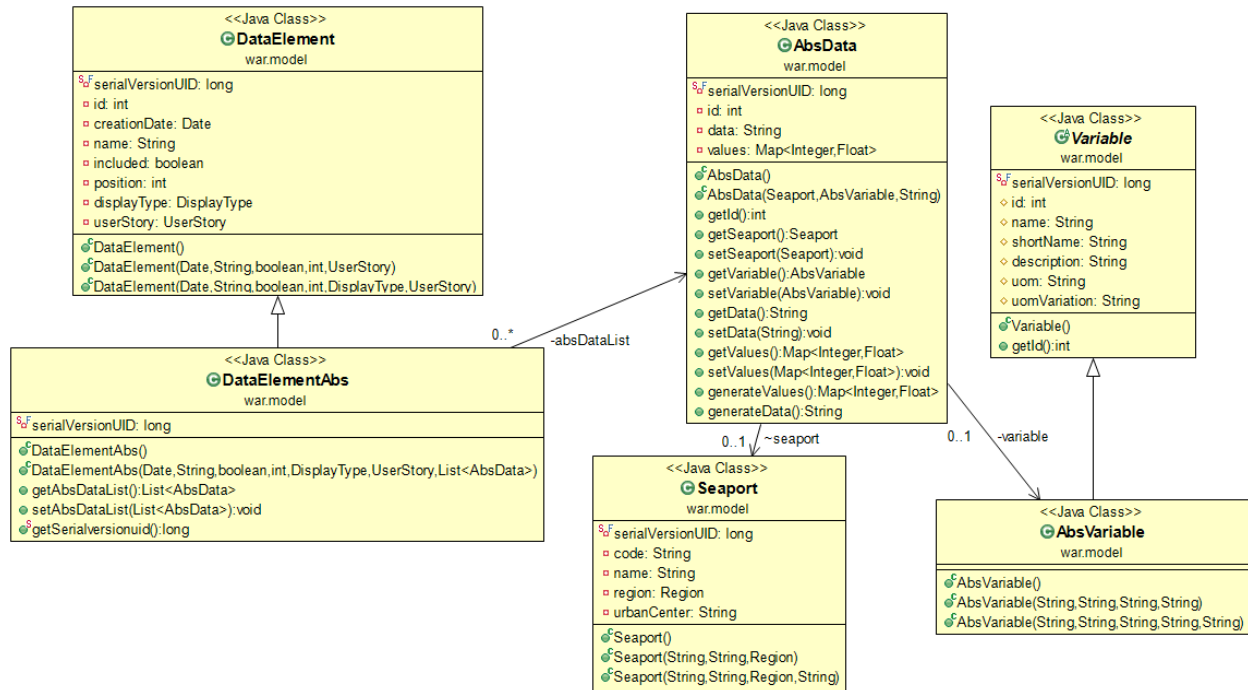


Figure 9: ABS data model's class diagram

This diagram shows a typical link between a data element and an underlying dataset. It corresponds to the explanation provided in *Figure 8*: simple schema of how data elements reference datasets.

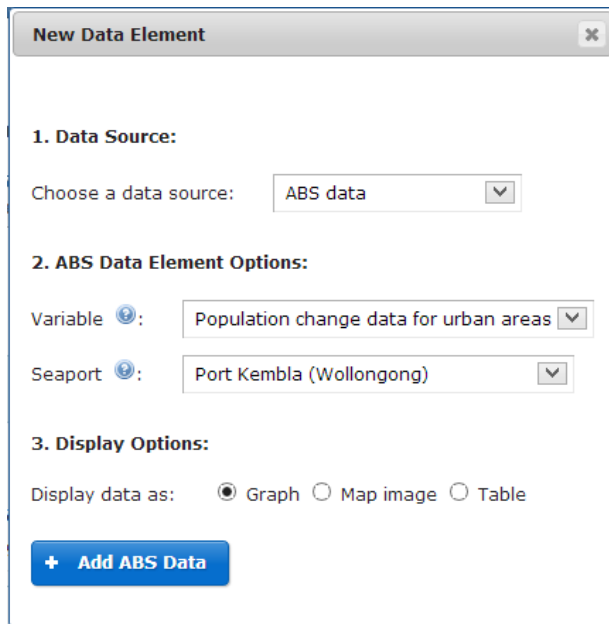


Figure 10: screenshot of ABS data element creation

The screenshot on the left shows the user interface to create a new data element of type ABS.

The first option allows selecting the ABS data data type.

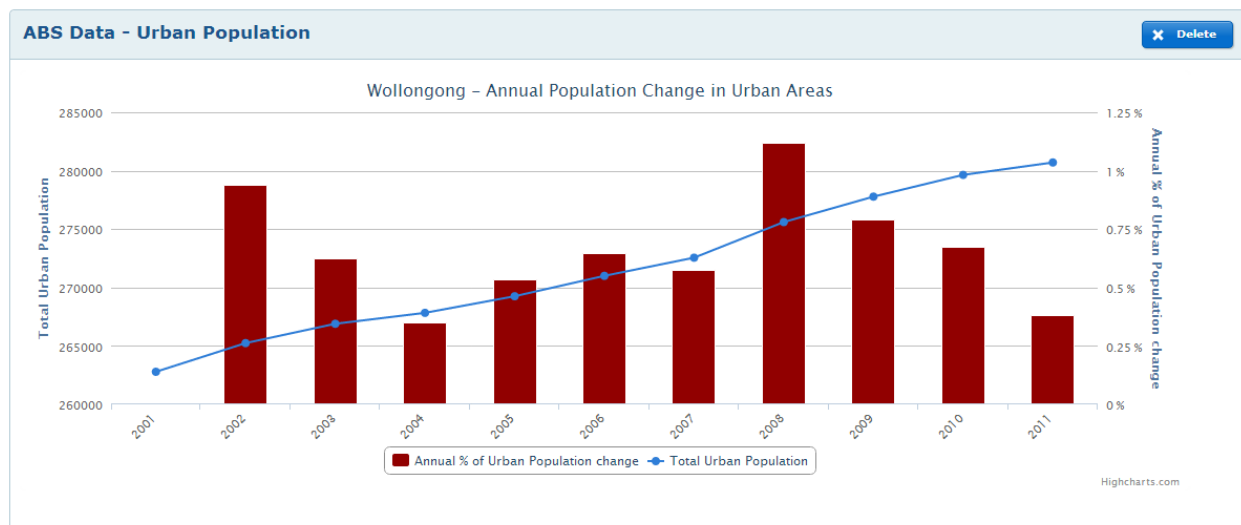
The second section of let the user choose between the different ABS variable available (so far, only “Population change”) and between the different Seaports. The seaports having no urban center are not selectable since they can’t have population change data.

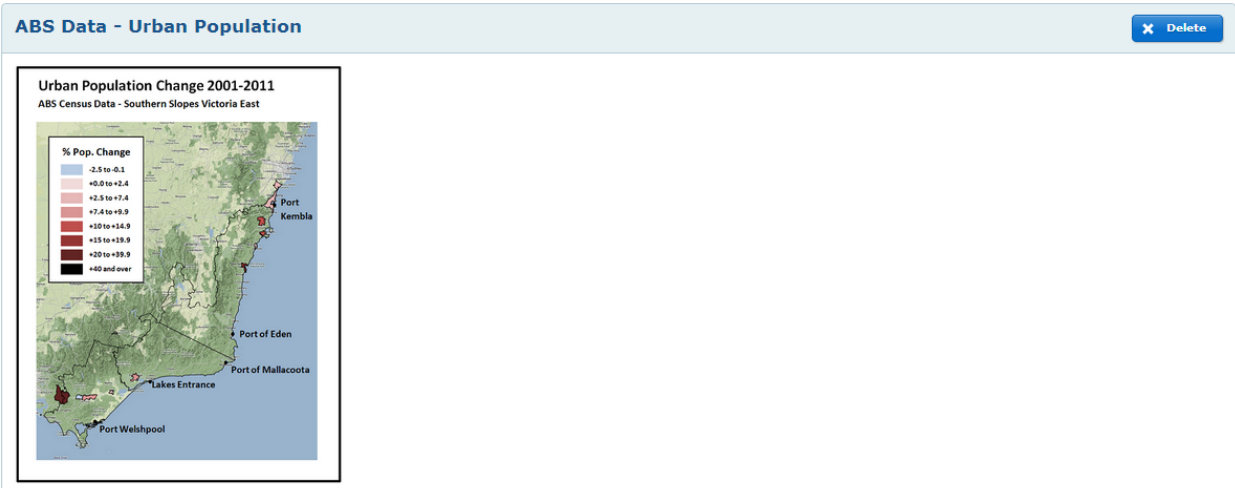
The third section allow choosing between different display options: this doesn’t affect the data referenced by the Data Element but only the way this data is displayed.

The resulting Data Element can be a graph showing the population growth (blue line on the example below) and the way the population increases between each year (red columns in the graph below).

The way the ABS Data Elements are displayed is in the view *dataElementAbs.jsp*

Figure 11: example data elements (one of each display type in ABS data):





**ABS Data - Urban Population** ✕ Delete

**Urban Population - Wollongong**

2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	Growth (%)	Total Growth (people)	Average Growth (%)
262,785	265,247	266,898	267,829	269,255	270,997	272,556	275,602	277,778	279,642	280,705	6.819	17,920	0.62

#### 4.6.2 BITRE DATA (PORTS AUSTRALIA)

The name BITRE data was given to this because the data of this dataset was supposed to come from the Bureau of Infrastructure, Transports and Regional Economics (BITRE). Finally, the data has been computed directly from the ports of Australia data, and therefore called as such in the user interface. However, the classes and database tables haven't been refactored and still hold the name BITRE.

The architecture of the BITRE data and data elements is similar to the ABS data, except the variables belong to categories.

To create a BITRE data element, the user chooses a variable category and a seaport: the data referenced by the BITRE data element is all the data related to the selected seaport and to any of the variable of the selected category.

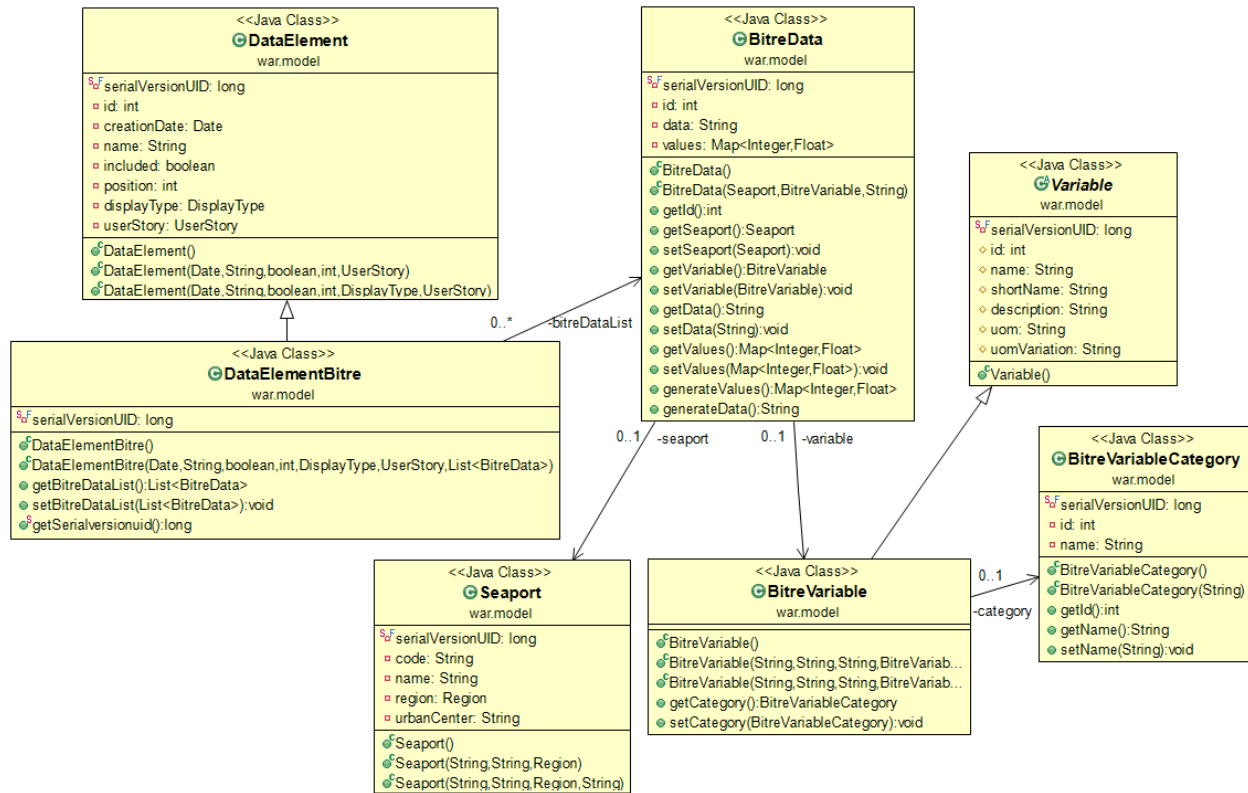
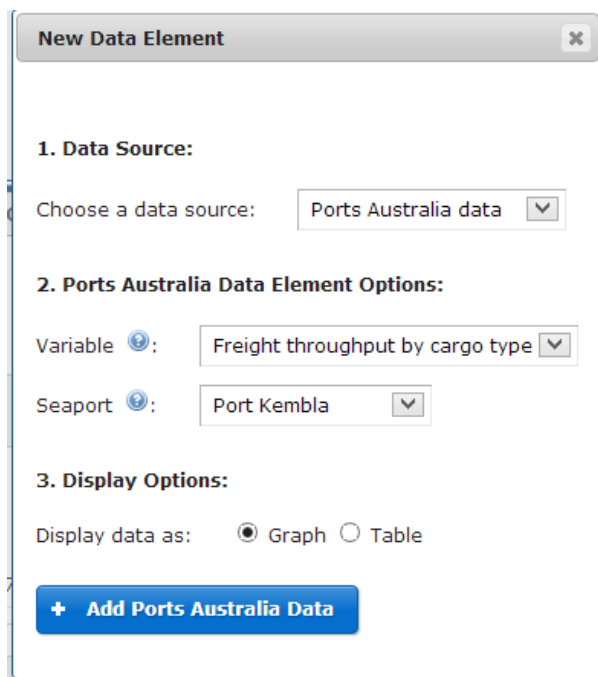


Figure 12: BITRE data model's class diagram



**New Data Element**

1. Data Source:

Choose a data source: Ports Australia data

2. Ports Australia Data Element Options:

Variable: Freight throughput by cargo type

Seaport: Port Kembla

3. Display Options:

Display data as: ☒ Graph ☐ Table

+ Add Ports Australia Data

Figure 13: screenshot of BITRE data element creation

The screenshot on the left shows the user interface to create a BITRE (Ports Australia) data element.

The first section is the selection of the “Ports Australia data” source.

The second option corresponds to the selection of the seaport and variable category.

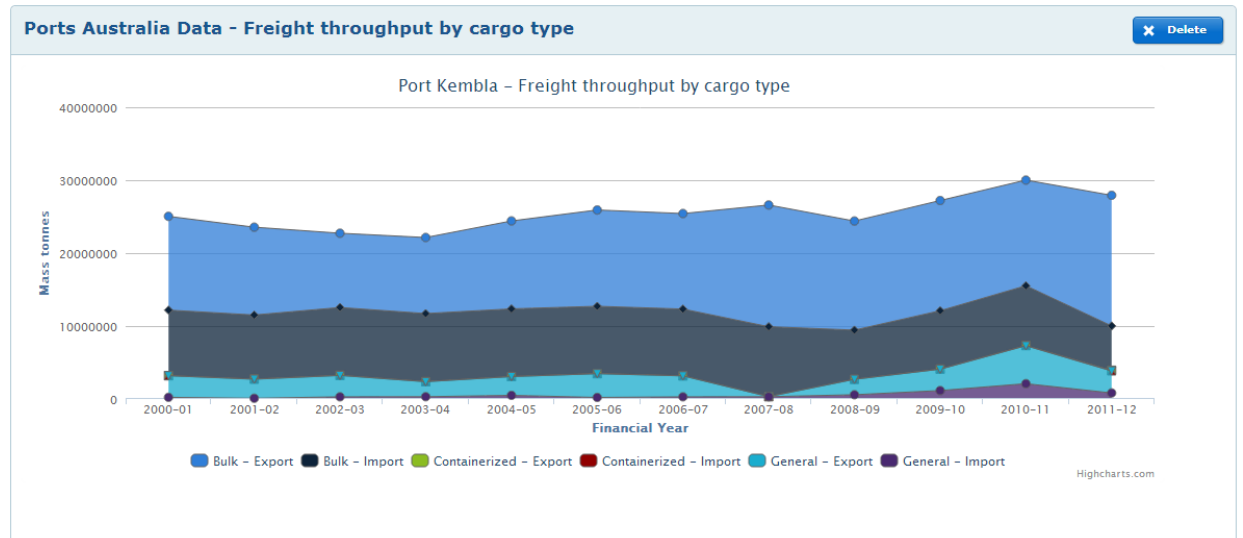
The third section allow choosing between different display options: this doesn't affect the data referenced by the data element but only the way this data is displayed.



The resulting data element can be a table or a graph plotted based on the data. There are 3 different types of graph, one for each variable category.

The way the BITRE Data Elements are displayed is in the view *dataElementBitre.jsp*

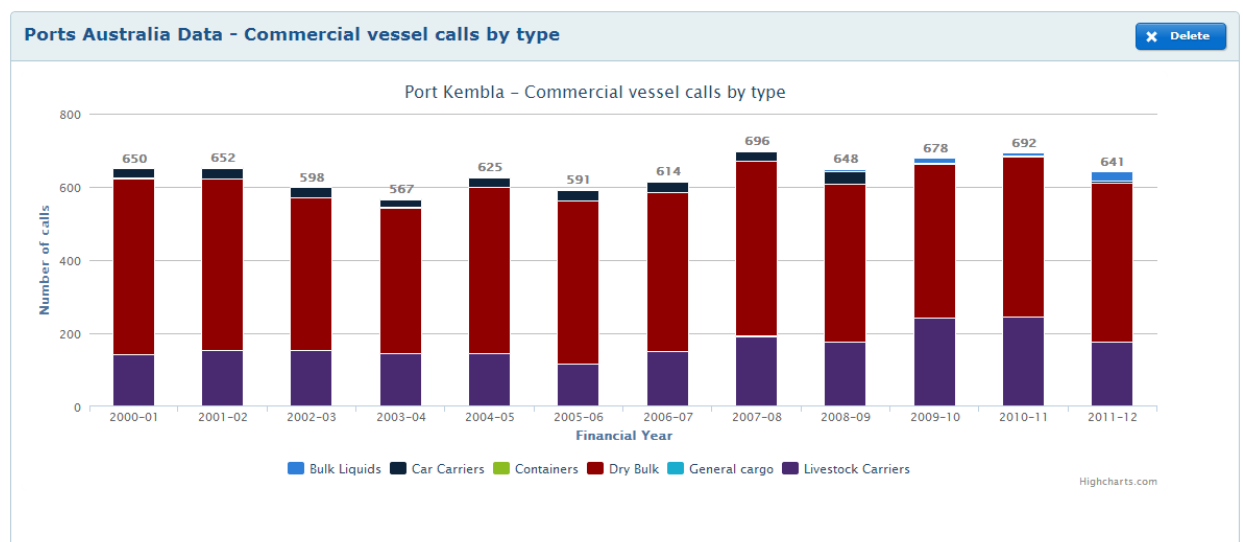
Figure 14: example data elements (graph for each BITRE variable category and a table display for the first category):

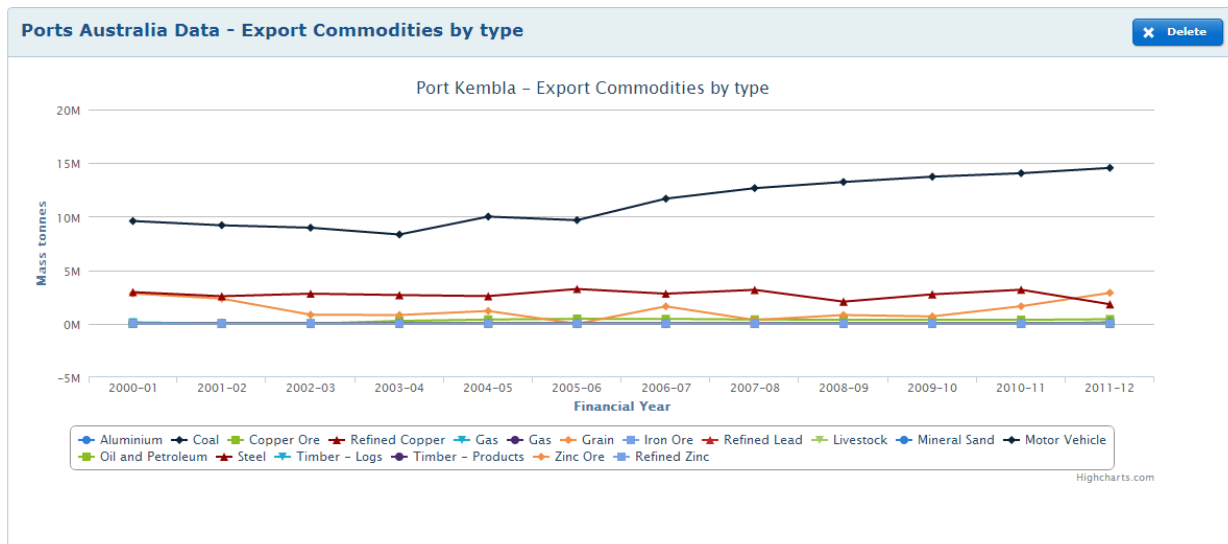


**Ports Australia Data - Freight throughput by cargo type** ✕ Delete

**Port Kembla - Freight throughput by cargo type**

Financial Year	2000-2001	2001-2002	2002-2003	2003-2004	2004-2005	2005-2006	2006-2007	2007-2008	2008-2009	2009-2010	2010-2011	2011-2012
<b>Bulk - Export</b>	12,853,239	12,051,792	10,188,944	10,438,099	12,045,676	13,203,775	13,100,871	16,684,623	14,959,831	15,107,871	14,484,573	17,915,908
<b>Bulk - Import</b>	8,985,314	8,765,896	9,292,500	9,308,784	9,273,930	9,232,962	9,137,586	9,583,765	6,737,328	7,984,614	8,186,946	6,068,593
<b>Containerized - Export</b>	8,584	17,370	17,130	21,565	22,720	10,023	10,490	818	5,760	21,353	14,167	80,342
<b>Containerized - Import</b>	4,531	3,170	3,476	595	2,316	668	312	2,625	4,469	11,732	15,603	23,042
<b>General - Export</b>	2,960,356	2,620,698	2,936,526	2,062,849	2,559,313	3,256,933	2,881,253	10,614	2,103,952	2,932,044	5,210,106	3,011,318
<b>General - Import</b>	229,084	100,694	293,020	314,860	505,888	215,579	293,687	312,393	592,616	1,151,199	2,096,321	824,858





### 4.6.3 CUSTOM FILES

### 4.6.4 PAST DATA (BOM AND CSIRO TRENDS)

The past data originate from the Bureau of Meteorology (BoM) and CSIRO. It is perhaps the simplest type of data element since it doesn't depend on any variable. The user is provided with a list of all the past data available and can choose one of them which will be referenced in the data element.

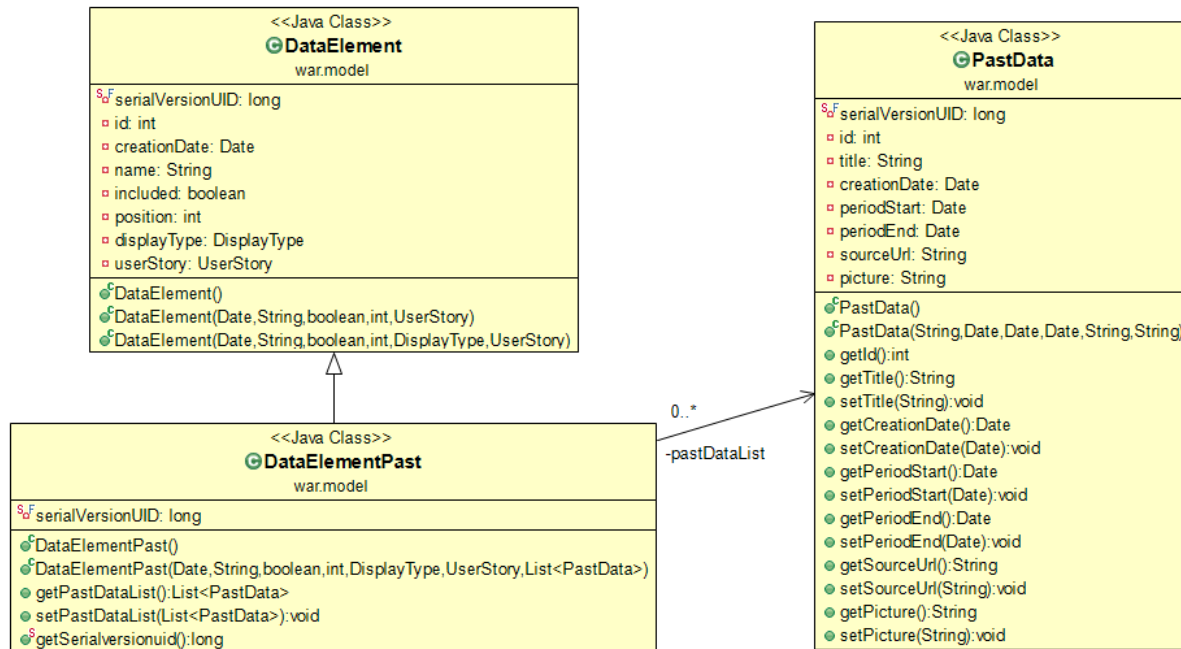
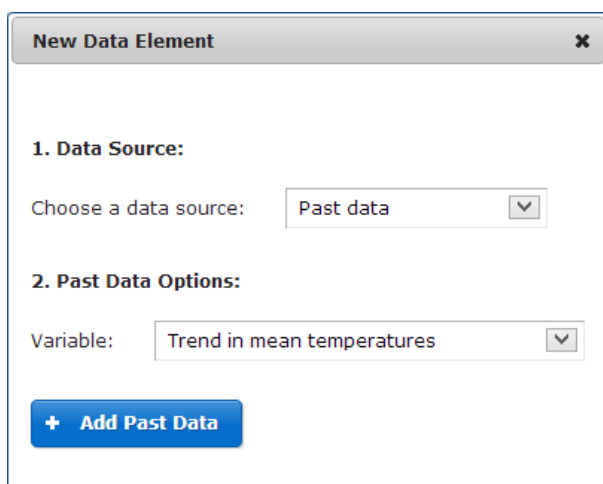


Figure 15: Past data model's class diagram

A past data consists in a *title*, a period covered represented by 2 dates (*periodStart* and *periodEnd*), a path to the *picture* representing the data itself and a URL to the source of data (*sourceURL*).



The screenshot shows a dialog box titled "New Data Element" with a close button (X). It contains two sections:

- 1. Data Source:** A label "Choose a data source:" followed by a dropdown menu showing "Past data".
- 2. Past Data Options:** A label "Variable:" followed by a dropdown menu showing "Trend in mean temperatures".

At the bottom, there is a blue button with a plus icon and the text "Add Past Data".

Figure 16: screenshot of Past data element creation

The screenshot on the left shows the user interface to create a past data element.

The first section is the selection of the "Past data" source.

The second option corresponds to the selection of the data title (called "variable" in the user interface to remain consistent across all data element creation).

The past data is always displayed as a picture and a hyperlink, so there is no display option.

The resulting data element is always a picture, which path depends on the user's variable selection and on which region the Workboard is related to. The picture path is constructed as follows.

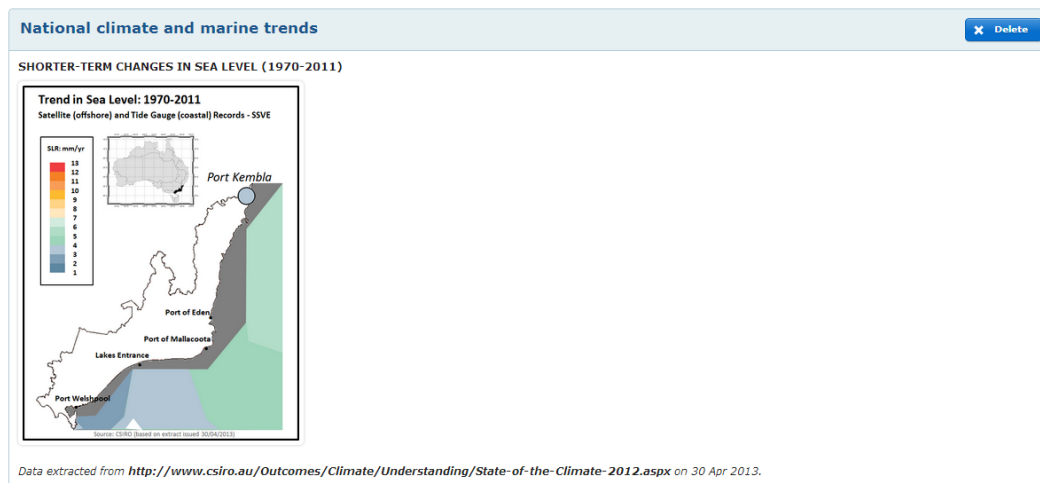
dataElementPast.jsp:

```
/resources/img/data/bom/  
${pastData.picture}-${fn:replace(userstory.seaport.region.name, ' ', '-')}-  
${formattedPeriodStart}-${formattedPeriodEnd}.png
```

Corresponding output:

/resources/img/data/bom/trend-mean-temp-east-coast-south-1970-2012.png

Figure 17: example data element:



#### 4.6.5 ACORN-SAT DATA

The ACORN-SAT data is corresponding to weather measures performed by local ACORN-SAT stations, selected by the climate change experts because measures are reliable. It is part of the observed data category.

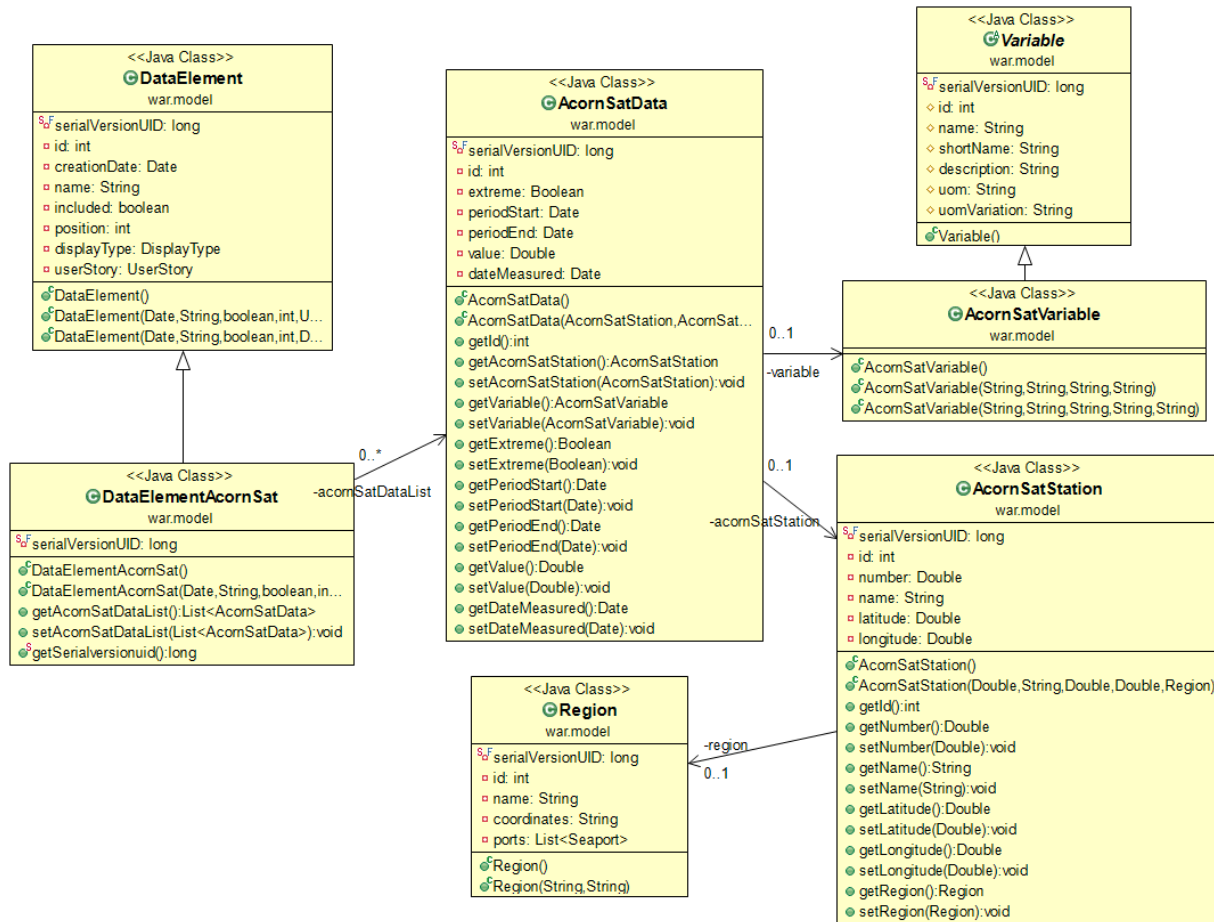


Figure 18: ACORN-SAT data model's class diagram

Unlike the other types of data, ACORN-SAT data are not related to a seaport. They are related to a variable and to an ACORN-SAT station (the station which measured it).

ACORN-SAT data exist across a **specific period of time**. An ACORN-SAT measure can be mean or extreme. **Mean measure** correspond to the mean value of the measured variable across the period. **Extreme measure** correspond to the most extreme value (highest or lowest) of the measured variable across the period.

The ACORN-SAT Data Elements always displays all available measures for all ACORN-SAT stations of the region. The user is only prompted to choose between mean or extreme measures.

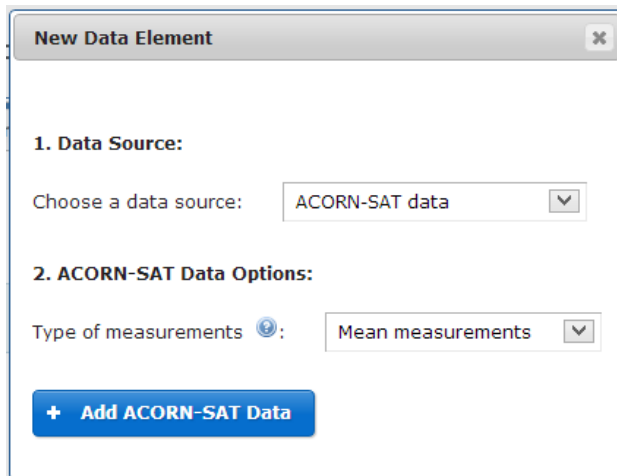


Figure 19: screenshot of ACORN-SAT data element creation

The screenshot on the left shows the user interface to create an ACORN-SAT data element.

The first section is the selection of the “ACORN-SAT data” source.

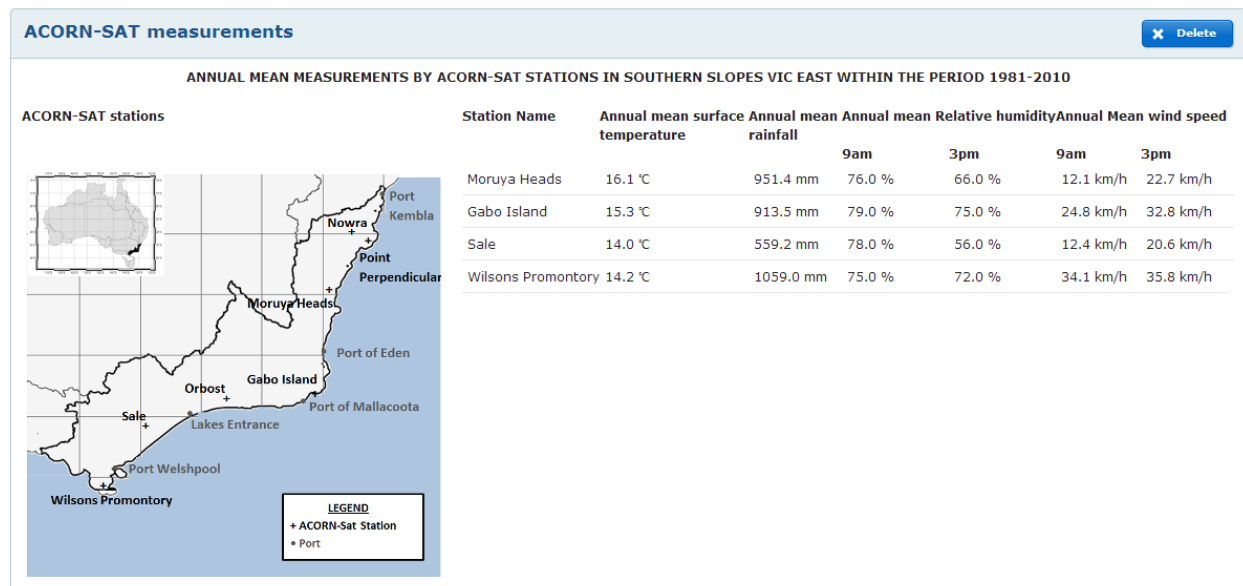
The second option corresponds to the selection of the type of measurement. Unlike the other Data Elements, the ACORN-SAT is not related to a single variable but contains several measures, each related to a variable and an ACORN-SAT station.

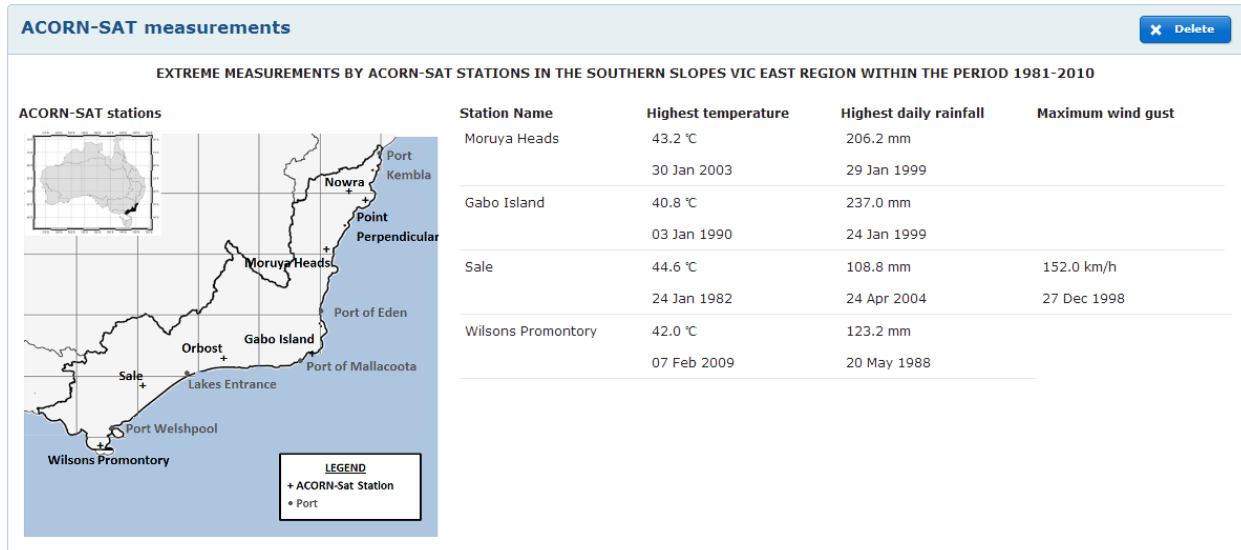
The past data is always displayed as a table showing the variables values and a picture of the ACORN-SAT stations location in the region, so there is display option.

The resulting data element is a table dynamically generated from the list of all the mean (or extreme) measures in the region. The rows are generated from the weather stations, and the columns from the variables.

The way the ACORN-SAT Data Elements are displayed is in the view *dataElementAcornSat.jsp*

Figure 20: example data elements for mean and for extreme measurements:





#### 4.6.6 CSIRO DATA

The dataset from CSIRO (Commonwealth Scientific and Industrial Research Organisation) is under the future climate category. It gathers forecast data about several variables: temperature, rainfall, relative humidity and wind speed. Each CSIRO data correspond to one of these climate variables, one set of climate parameters and one year of forecast.

The year can be 2030, 2055, or 2070. A set of climate parameters is composed of a climate model, a CO2 emission scenario, and a region.

The climate models are labeled “Hotter & Drier”, “Most Likely” or “Cooler & Wetter”. Note that for different regions, a same label can correspond to different underlying climate models.

*example: “Hotter & Drier” label corresponds to the “CSIRO mk3.5” model in the region “East Coast South”, but corresponds to the “Miroc 3.2 medres” model in the region “Southern and Southwestern Flatlands”.*

Each CSIRO data contains the value of a variation compared to a base value. For this variation to be meaningful, the CSIRO data has to be related to a base value onto which the variation should be applied. This is why it is linked to a baseline data which holds the base value for a given variable in a given region. This baseline value corresponds to the current value of the variable. It’s not a forecast.

*example: variable: temperature; year: 2055; climate parameters: region “Southern Slopes Vic East”, emission scenario “Medium (A1B)”, climate model “Most Likely” (which corresponds to the “IPSL CM-4” model in this region). The baseline value is: 12.8 °C  
The variation in 2055 is: +1.9 °C*

The following diagram shows how this has been modeled in term of Java classes:

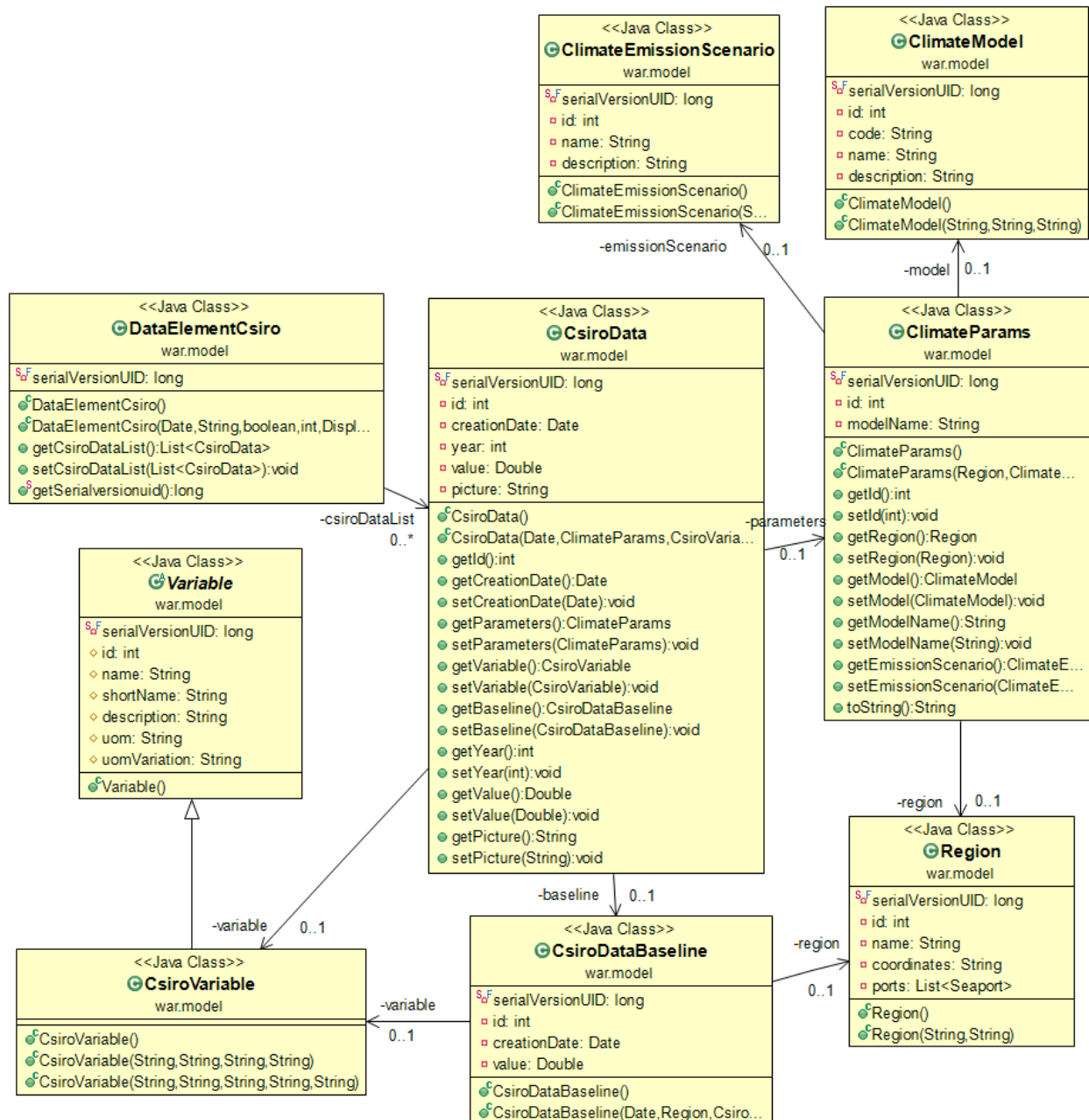
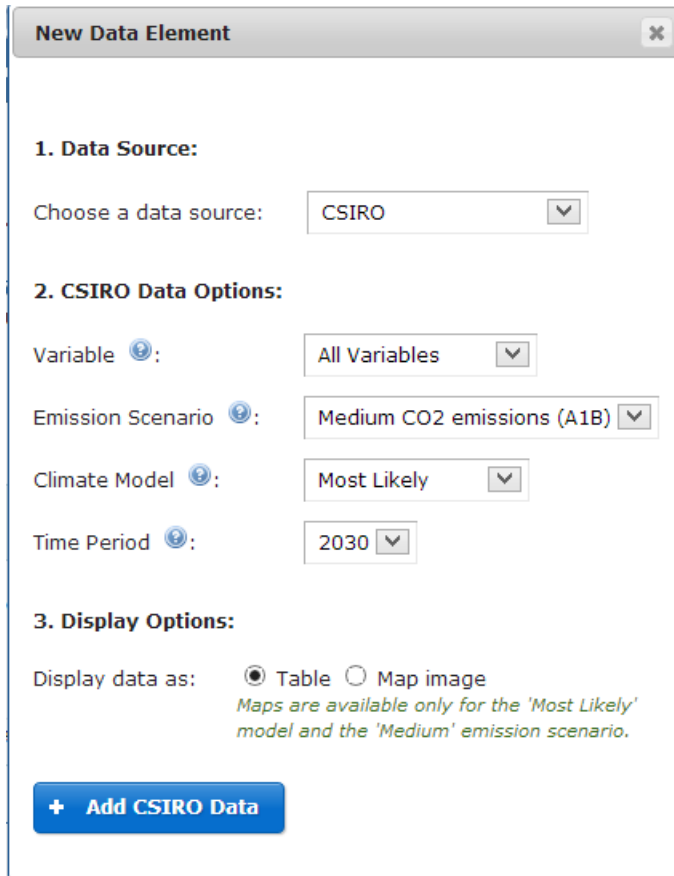


Figure 21: CSIRO data model's class diagram

As explained above, a CSIROData has a *CsiroVariable*, *ClimateParams* and a *CsiroDataBaseline*. The *ClimateParams* has a *Region*, *ClimateEmissionScenario* and *ClimateModel*. The *CsiroDataBaseline* has a *Region* and a *CsiroVariable*.

A CSIRO data element can be composed of one or several *CSIROData*. This allows having for example a table of all the variables for a given year in a single data element.





The screenshot on the left shows the user interface to create a CSIRO data element.

The first section is the selection of the “CSIRO data” source.

The second section corresponds to the selection of the CSIRO data options. It lets the user choose a single variable or all variables, one emission scenario, one climate model, and one year.

The last option is the display type: ideally, map image would be available for all combination of options, but so far they are only available for “Most Likely” models and “Medium” emission scenario.

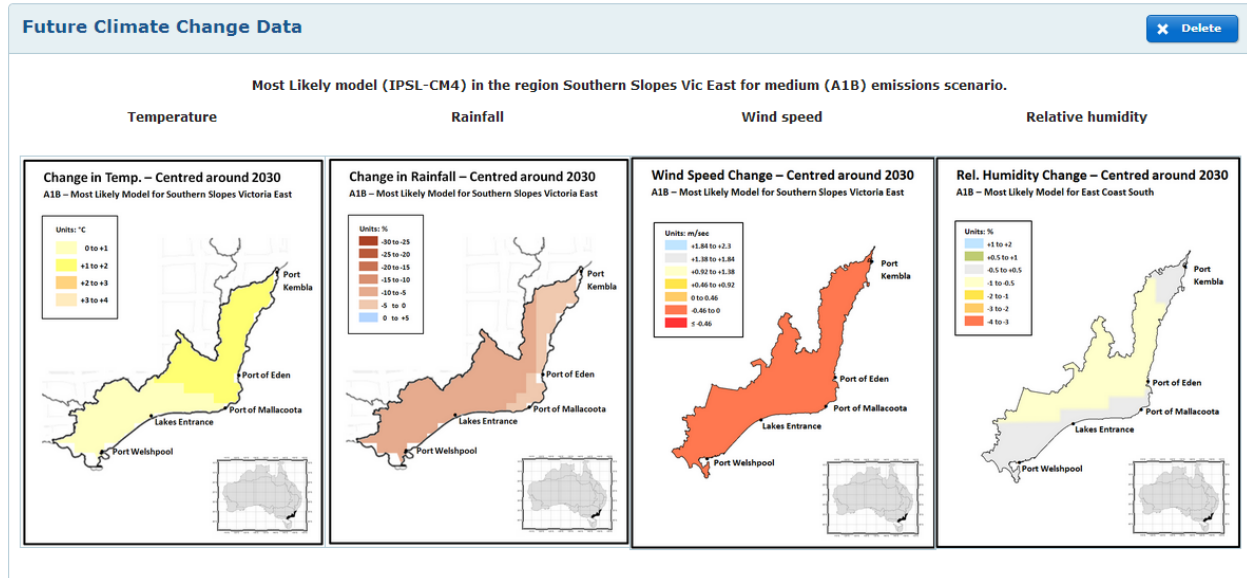
Data in tables are available for any combination of option.

Figure 22: screenshot of a CSIRO data element creation

The resulting data element is either a table based on the selected options giving a value for the entire region, or a picture of the region map giving more details about the variable value within the region.

Figure 23: example of CSIRO data elements (multiple variable, table and map picture display types)

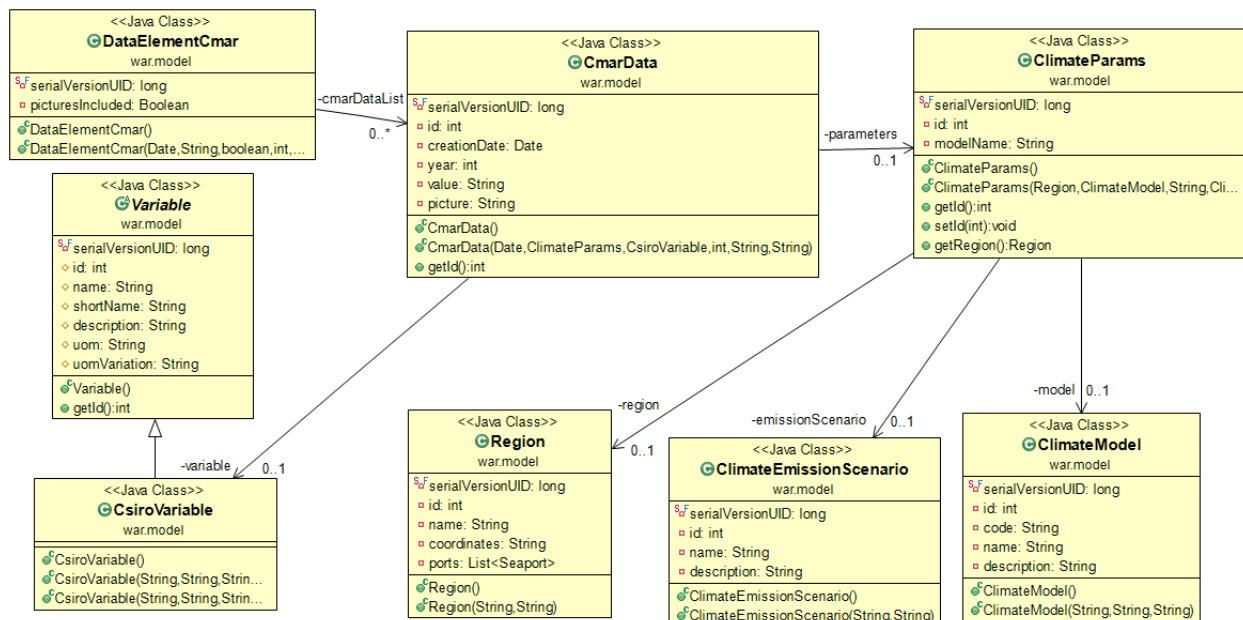
Future Climate Change Data <span>✕ Delete</span>		
Most Likely model (IPSL-CM4) in the region Southern Slopes Vic East for medium (A1B) emissions scenario.		
Variable	Baseline	Change centred around 2030
Temperature	12.8 °C	+1.0 °C
Rainfall	870.0 mm/y	-4.1 %
Wind speed	5.3 km/h	-2.7 %
Relative humidity	71.3 %	-0.8 %
Data provided by CSIRO on 28 Jul 2008 was the best available to date		



#### 4.6.7 CMAR DATA

CMAR data is almost the same as CSIRO data. It has climate parameters (still composed of region, climate model and emission scenario. For details, see 4.6.6 CSIRO Data), and even uses the same variable type as CSIRO data (*CsiroVariable*).

The only difference is that there is no baseline data for CMAR data.



So far, there is only one variable available for the CMAR data: the sea level rise. The values for the sea level rise variable are only available for the “*Medium (A1B)*” emission scenario, the “*Most Likely*” climate model and the years 2030 and 2070.

A CMAR data element is only related to a single sea level rise variable, but it holds several values for several locations in its “*value*” field, as a String formatted as follows: *latitude1,longitude1,value1; latitude2,longitude2,value2;...; latitudeN,longitudeN,valueN;*

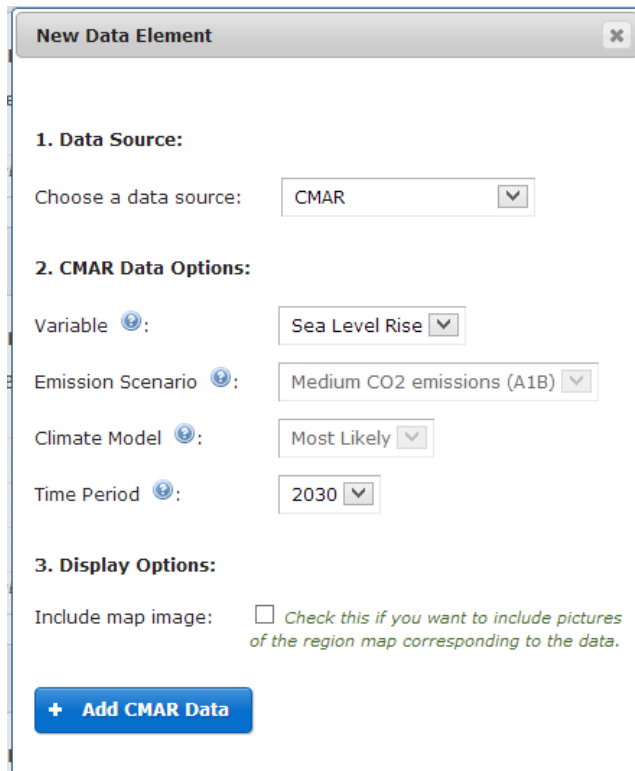


Figure 25: screenshot of a CSIRO data element creation

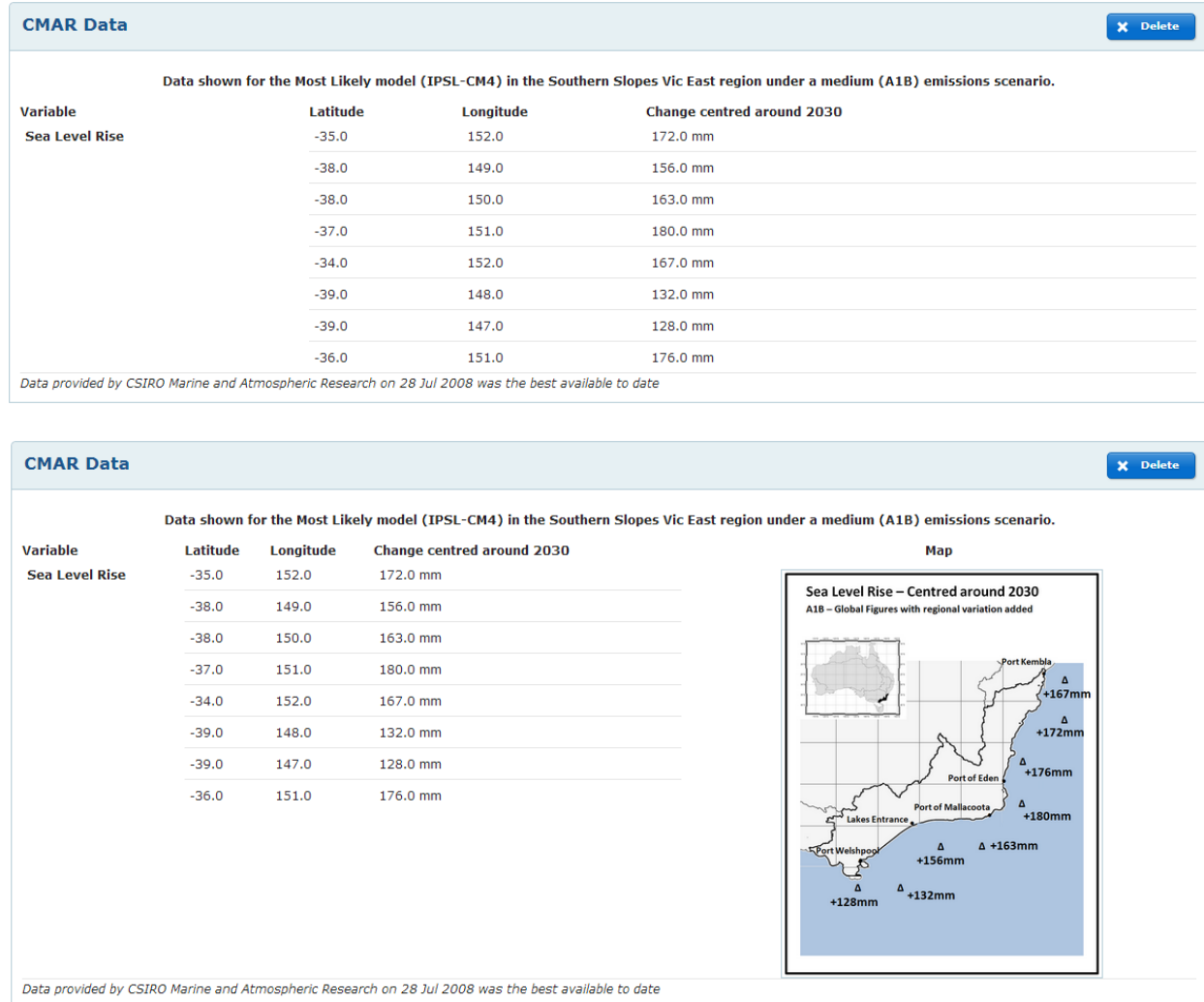
The screenshot on the left shows the user interface to create a CMAR data element. The first section is the selection of the “CMAR data” source.

The second section corresponds to the selection of the CMAR data options. It lets the user choose a single variable (only sea level rise available so far, and one year among 2030 or 2070). The Emission scenario and Climate Model dropdown lists are disabled since CMAR data is only available for these options.

The last option allows adding a map image to the table of data.

The resulting data element is always a table containing the sea level rise values in different locations (latitudes and longitudes). If the checkbox has been checked by the user, an additional column is added to the table containing a map picture which shows the values positioned on a map of the region.

Figure 26: example of CMAR data elements (table with and without map picture)



#### 4.6.8 ENGINEERING MODEL DATA (CONCRETE DETERIORATION)

The concrete deterioration model, or engineering model, gives the possibility to users to get a forecast of the future deterioration of concrete assets of a seaport.

Engineering model data aren't based on a static dataset like most of the others data sources. There exist an Excel template containing all the necessary macros and engineering data as necessary to compute the future concrete deterioration, based on some value defining an asset and on the same climate models and emissions scenarios that the CSIRO and CMAR data are using.

An external application has been specifically developed as part of a different project, funded by NCCARF. It provides a web user interface to this Excel tool, and a certain visualization of the output data. It also allows downloading the output data as what we call “Output Excel Files”.

The URL of the external engineering model tool is: <http://seaports.eres.rmit.edu.au:443/ccimt/>  
A link in the Climate Smart Seaports application is set directly in the *workboardToolbox* view: *src/main/webapp/WEB-INF/views/workboardToolbox.jsp*:

```
...

<a href="#" class="helpTooltip" title="Please use the <b><a
href=&quot;http://seaports.eres.rmit.edu.au:443/ccimt&quot; title=&quot;Go to
the concrete deterioration model tool&quot; target=&quot;blank&quot;>concrete
deterioration modelling tool</a></b> to import your port's assets data. That
tool will verify each entry and provide a forecast of the concrete
deterioration for each asset, under the form of Excel files.</p><p>You can
upload these Excel files here to use the forecast data in you
workboard.</p>">"
alt="Help" /></a>:

...

<p class="hint">
  <i>Upload an Excel file (.xls) generated by the <a
href="http://seaports.eres.rmit.edu.au:443/ccimt" title="Go to the concrete
deterioration model tool" target="blank">concrete deterioration tool</a>.</i>
</p>

...
```

Since this application was developed externally in C#.NET, it couldn't be easily integrated into the Climate Smart Seaports application which is strongly oriented toward Java: it would have been necessary to re-implement the entire logic of the other application, and that wasn't the scope of the project.

What has been done instead is an extraction algorithm, which takes as input the “Output Excel Files” from the engineering model tool, and saves this forecasted deterioration data into the Climate Smart Seaports database.

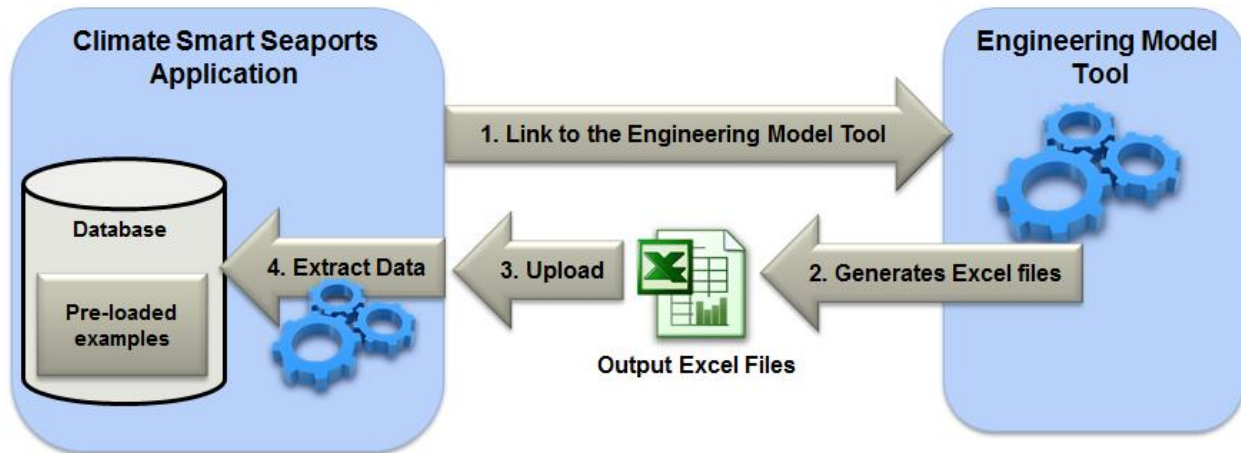


Figure 27: process of the Engineering Model tool integration

The above diagram (Figure 27) shows the process of how the Engineering Model tool is integrated with the Climate Smart Seaports application:

1. A hyperlink is provided from the Climate Smart Seaports application to the Engineering Model tool.
2. The user go through the Engineering Model tool process, setting all the required input data, and getting the tool to generate the output Excel file(s).
3. The user can upload each Excel file and get the Climate Smart Seaports tool to create a data Element based on the data for one of the 16 engineering variables present in the output Excel file.
4. The data to extract consists in: An **Engineering Model Asset**, storing the values regarding the concrete asset itself, given as input in the engineering model tool and the **Engineering Model Data** corresponding to the **Engineering Variable** the user selected. The data extracted are stored as *EngineeringModelData* in the Climate Smart Seaports database, and automatically added to a new data element. Each *EngineeringModelData* corresponds to an *EngineeringModelAsset*, an *EngineeringVariable* and a set of climate parameters. These climate parameters are the same as those used for CSIRO and CMAR data, composed of a region, a climate model and an emissions scenario. Each engineering data element is composed of 7 engineering model data, corresponding to the following combination: data for 1 variable for 1 asset in 1 region for 2 emissions scenarios and 3 climate models, plus 1 “Base” climate model:  $1 \times 1 \times 1 \times (2 \times 3 + 1) = 7$

Instead of going through steps 1 and 2, there is also a possibility for the user to choose to use a pre-defined example. This option can be chosen if the user doesn't have the necessary engineering knowledge to use the engineering model tool, but still want to integrate concrete deterioration data.

The pre-defined examples have been loaded in advance by the Climate Smart Seaport development team using the steps 1 and 2 mentioned above. There are pre-loaded examples only for the regions “East Coast South” and “Southern Slopes Vic East”.

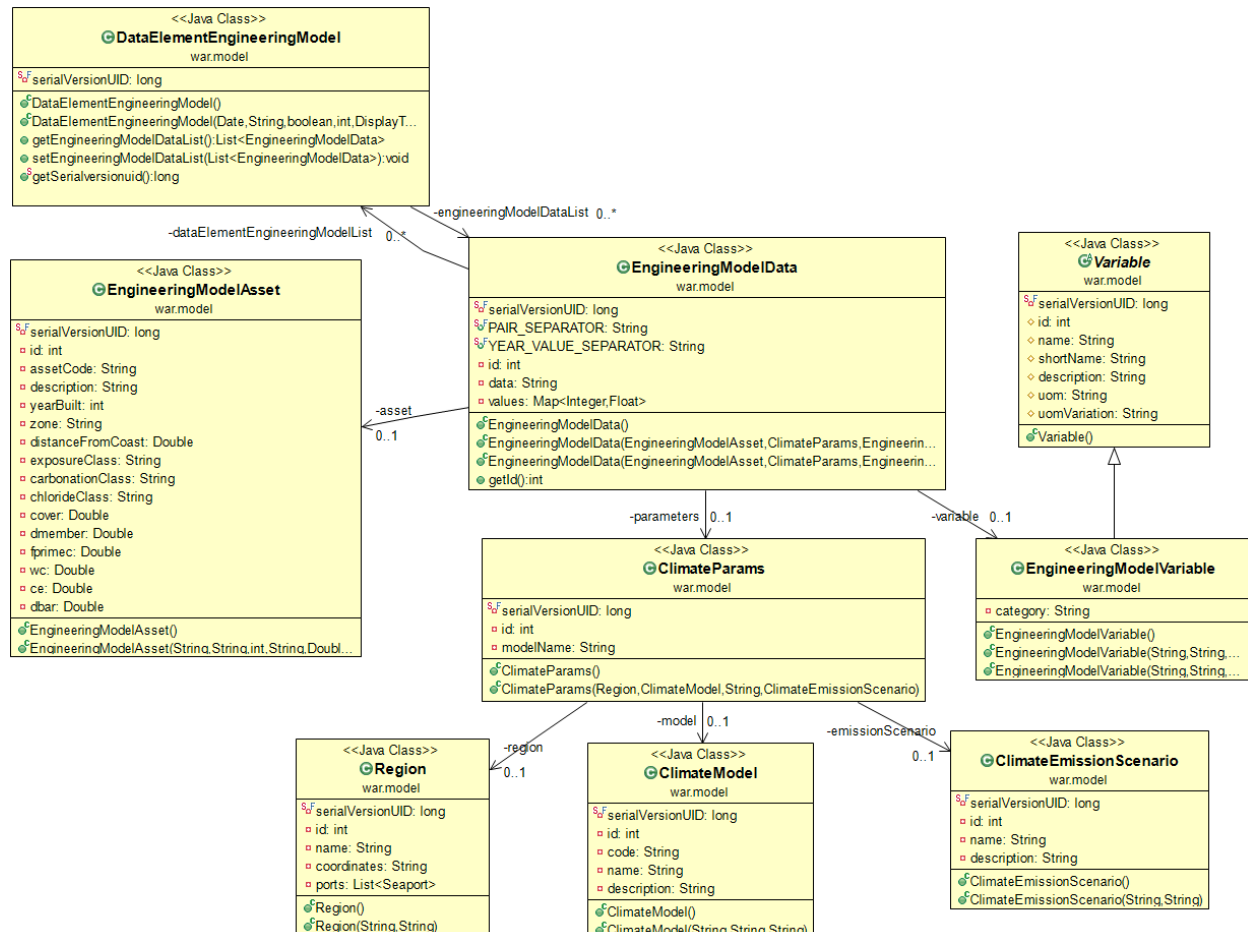


Figure 28: Engineering Model data model's class diagram

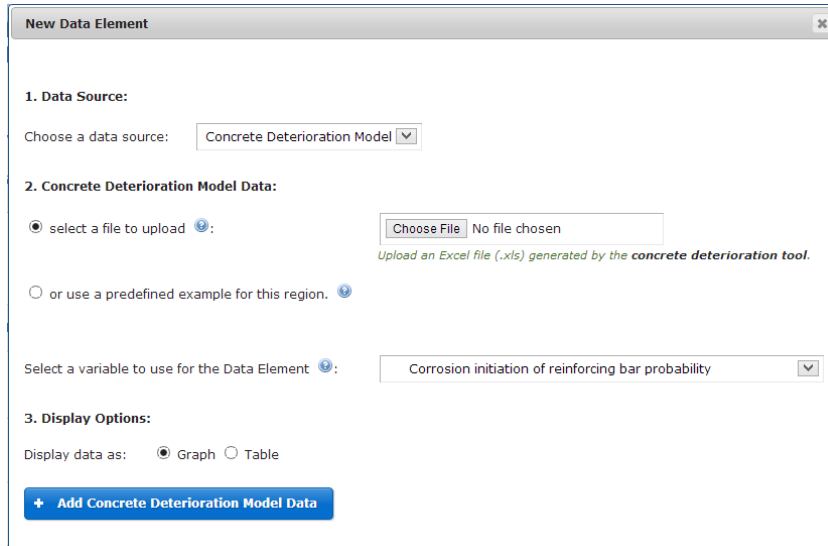
Note that there is a defined number of Engineering Model Assets considered as examples this number is defined in the class *EngineeringModelHelper.java* (src/main/java/helpers):

```

/**
 * Number of assets IDs used for engineering models examples
 */
public static final int ENGINEERING_MODEL_EXAMPLE_ASSETS_COUNT = 3;

```

It is defined to 3 at the moment because there are 3 regions available. That means the first 3 Assets added to the database are considered as Engineering Model examples.



The screenshot shows a web interface titled "New Data Element". It is divided into three main sections:

- 1. Data Source:** A dropdown menu labeled "Choose a data source:" is set to "Concrete Deterioration Model".
- 2. Concrete Deterioration Model Data:**
  - There are two radio buttons. The first is selected and labeled "select a file to upload". Next to it is a "Choose File" button and the text "No file chosen". Below this is a link: "Upload an Excel file (.xls) generated by the concrete deterioration tool."
  - The second radio button is labeled "or use a predefined example for this region."
  - Below the radio buttons is a label "Select a variable to use for the Data Element:" followed by a dropdown menu showing "Corrosion initiation of reinforcing bar probability".
- 3. Display Options:** A label "Display data as:" is followed by two radio buttons: "Graph" (which is selected) and "Table".

At the bottom of the form is a blue button with a plus icon and the text "Add Concrete Deterioration Model Data".

Figure 29: screenshot of a concrete deterioration data element creation

The screenshot on the left shows the user interface to create an engineering model data element.

The first section is the selection of the “Concrete Deterioration data” source.

The second section is the selection of the options specific to engineering model data. The user can choose between selecting an output excel file to upload and using

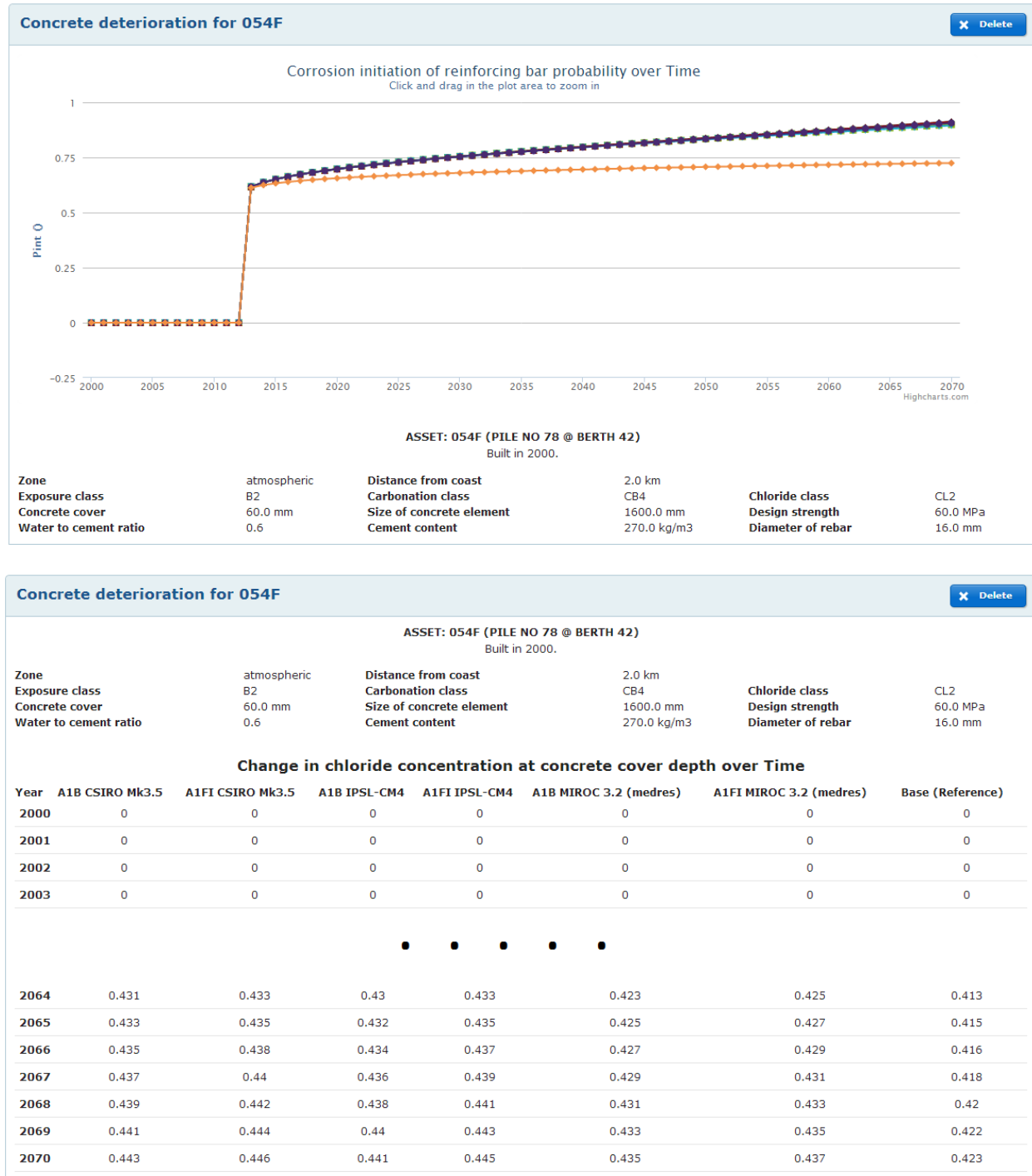
a pre-defined example for the region. The following option defines which engineering variable should be extracted from the file or example and put in the data element.

The last section gives a choice between displaying the data as a graph or as a table.

The resulting data element can be a table or a graph representing the evolution of the selected engineering variable over time (time period 2000 to 2070).



Figure 30: example of concrete deterioration data elements (graph and table), according to different climate models and emission scenarios



#### 4.6.9 VULNERABILITY ASSESSMENT DATA

Vulnerability assessment data does not come from a data set but directly from the user input. The user is prompted with several questions about past weather events that have impacted the seaport.

Since no underlying dataset is modeled in the database, the classes' organization remains quite simple: a vulnerability data element references a weather event.

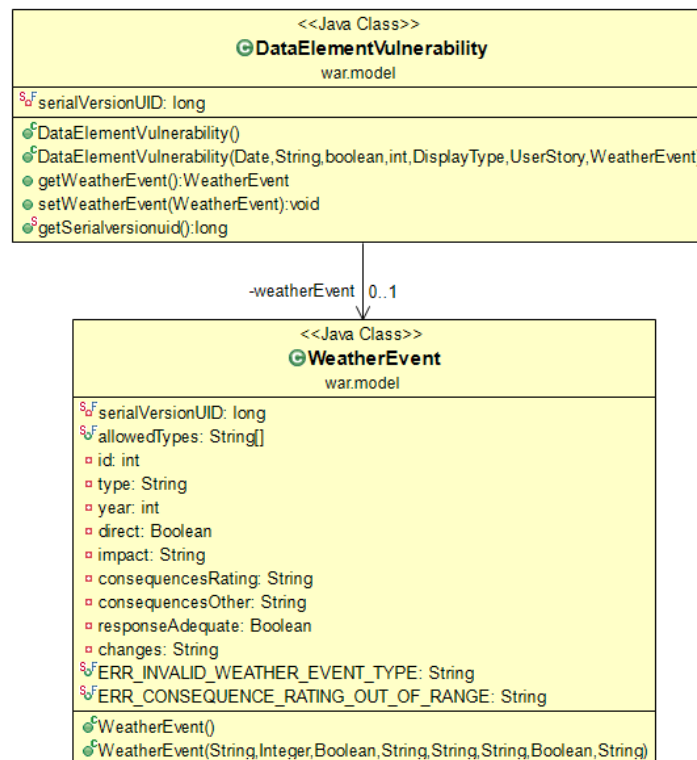
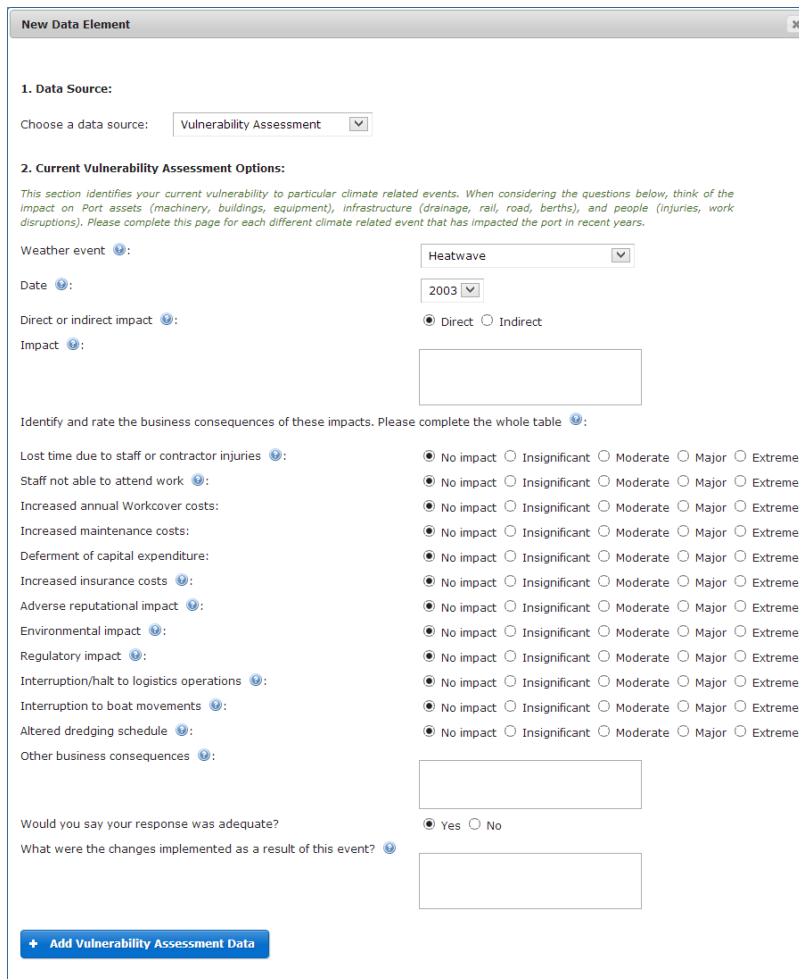


Figure 31: Vulnerability assessment data model's class diagram



**New Data Element**

**1. Data Source:**  
Choose a data source: Vulnerability Assessment

**2. Current Vulnerability Assessment Options:**  
*This section identifies your current vulnerability to particular climate related events. When considering the questions below, think of the impact on Port assets (machinery, buildings, equipment), infrastructure (drainage, rail, road, berths), and people (injuries, work disruptions). Please complete this page for each different climate related event that has impacted the port in recent years.*

Weather event: Heatwave

Date: 2003

Direct or indirect impact: ☒ Direct ☐ Indirect

Impact:

Identify and rate the business consequences of these impacts. Please complete the whole table:

Lost time due to staff or contractor injuries:	<input checked="" type="radio"/> No impact <input type="radio"/> Insignificant <input type="radio"/> Moderate <input type="radio"/> Major <input type="radio"/> Extreme
Staff not able to attend work:	<input checked="" type="radio"/> No impact <input type="radio"/> Insignificant <input type="radio"/> Moderate <input type="radio"/> Major <input type="radio"/> Extreme
Increased annual Workcover costs:	<input checked="" type="radio"/> No impact <input type="radio"/> Insignificant <input type="radio"/> Moderate <input type="radio"/> Major <input type="radio"/> Extreme
Increased maintenance costs:	<input checked="" type="radio"/> No impact <input type="radio"/> Insignificant <input type="radio"/> Moderate <input type="radio"/> Major <input type="radio"/> Extreme
Deferment of capital expenditure:	<input checked="" type="radio"/> No impact <input type="radio"/> Insignificant <input type="radio"/> Moderate <input type="radio"/> Major <input type="radio"/> Extreme
Increased insurance costs:	<input checked="" type="radio"/> No impact <input type="radio"/> Insignificant <input type="radio"/> Moderate <input type="radio"/> Major <input type="radio"/> Extreme
Adverse reputational impact:	<input checked="" type="radio"/> No impact <input type="radio"/> Insignificant <input type="radio"/> Moderate <input type="radio"/> Major <input type="radio"/> Extreme
Environmental impact:	<input checked="" type="radio"/> No impact <input type="radio"/> Insignificant <input type="radio"/> Moderate <input type="radio"/> Major <input type="radio"/> Extreme
Regulatory impact:	<input checked="" type="radio"/> No impact <input type="radio"/> Insignificant <input type="radio"/> Moderate <input type="radio"/> Major <input type="radio"/> Extreme
Interruption/halt to logistics operations:	<input checked="" type="radio"/> No impact <input type="radio"/> Insignificant <input type="radio"/> Moderate <input type="radio"/> Major <input type="radio"/> Extreme
Interruption to boat movements:	<input checked="" type="radio"/> No impact <input type="radio"/> Insignificant <input type="radio"/> Moderate <input type="radio"/> Major <input type="radio"/> Extreme
Altered dredging schedule:	<input checked="" type="radio"/> No impact <input type="radio"/> Insignificant <input type="radio"/> Moderate <input type="radio"/> Major <input type="radio"/> Extreme
Other business consequences:	

Would you say your response was adequate? ☒ Yes ☐ No

What were the changes implemented as a result of this event?

**+ Add Vulnerability Assessment Data**

Figure 32: screenshot of a vulnerability assessment data element creation

The screenshot on the left shows the user interface to create a Vulnerability data element.

The first section is the selection of “Vulnerability Assessment” source.

The second section is the vulnerability assessment that the user has to fill in. Each field corresponds to a property of the *WeatherEvent* class.

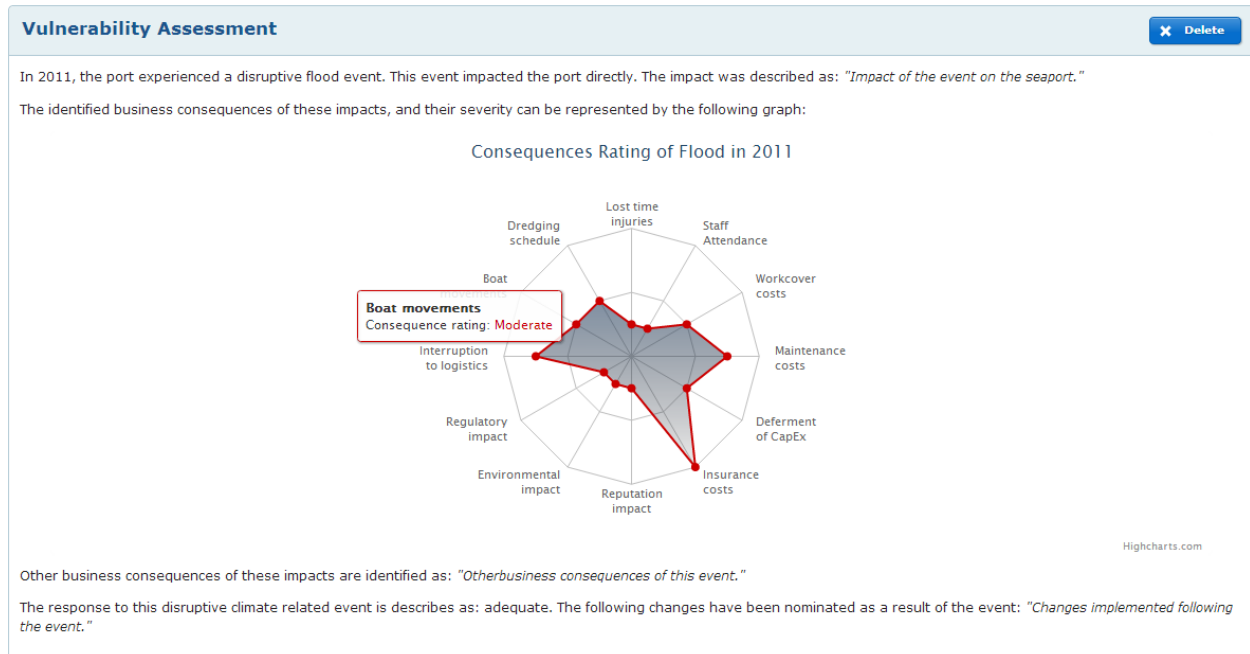
Note that the consequences rating are all stored in a single String field, formatted as the rating number (0 to 4) separated by comas (,).

example: 1,2,1,4,0,0,0,2,3,2,4,1

This formatting is used to store the rating more efficiently and generate an output graph more easily.

The resulting data element is a combination of statements generated automatically based on the user input, and a “spider” graph based on the rating of consequences.

Figure 33: example of vulnerability assessment data element



## 4.7 SECURITY

The Security module of the Spring framework is user to take care of the security of this application (<http://www.springsource.org/spring-security>). This section assumes that the basics of Spring Security are known, and details how it has been integrated into the Climate Smart Seaports application.

In the main web application configuration (*web.xml*), the location of the Spring Security context configuration file is referenced. This allows including it as part of the web application context.

*src/main/webapp/WEB-INF/web.xml:*

```

...
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    /WEB-INF/spring/root-context.xml
    /WEB-INF/spring/applicationContext-security.xml
  </param-value>
</context-param>
...

```

The Spring Security context configuration file defines several things, detailed below.

#### 4.7.1 AUTHENTICATION RULES

---

*extract of src/main/webapp/WEB-INF/spring/applicationContext-security.xml:*

```
...
<http auto-config="true" use-expressions="true">
  <intercept-url pattern="/public/**" access="permitAll" />
  <intercept-url pattern="/auth/**" access="hasAnyRole('ROLE_USER', 'ROLE_ADMIN')" />
  <intercept-url pattern="/admin/**" access="hasRole('ROLE_ADMIN')" />
  <form-login login-page="/login" authentication-failure-url="/loginfailed"/>
  <logout logout-success-url="/" />
  <access-denied-handler error-page="/accessDenied" />
</http>
...
```

Anyone can access URL starting with “public”

examples: “CSS/public/home” or “CSS/public/reports/list”.

Only authenticated users with a role ROLE\_USER or a role ROLE\_ADMIN can access URL starting with “auth”.

examples: “CSS/auth/workboard” or “CSS/auth/userstory”

Only authenticated users with a role ROLE\_ADMIN can access URL starting with “admin”.

example: “CSS/admin/users/list”

#### 4.7.2 USER LOGIN SERVICE

---

*extract of src/main/webapp/WEB-INF/spring/applicationContext-security.xml:*

```
...
<beans:bean id="userDao" class="war.dao.UserDao">
</beans:bean>

<beans:bean id="userService" class="security.UserLoginService">
  <beans:property name="userDao" ref="userDao"></beans:property>
</beans:bean>
...
```

The *UserLoginService* class enables to hook Spring Security with the *UserDao* class responsible of retrieving User objects from the Climate Smart Seaports database.

Two condition apply: *UserLoginService* needs to implements the *UserDetailsService* interface, and *User* needs to implement the *UserDetails* interface. These two interfaces are defined by Spring Security.

#### 4.7.3 PASSWORD ENCODER

---

*extract of src/main/webapp/WEB-INF/spring/applicationContext-security.xml:*

```
...
```

```
<authentication-manager>
  <authentication-provider user-service-ref="userService">
    <password-encoder hash="sha-256" />
  </authentication-provider>
</authentication-manager>
...
```

The *userService* bean is provided with a password encoder using the *sha256* hash. This allows Spring Security to transparently encode all the passwords before trying to authenticate with the database.

The same hash is used when a user registers, before saving the user in the database.

*extract of the registerNewUser method, in src/main/java/controllers/UserController.java:*

```
...
MessageDigest digest = MessageDigest.getInstance("SHA-256");
byte[] hash = digest.digest(user.getPassword().getBytes("UTF-8"));
StringBuffer hexString = new StringBuffer();
for (int i = 0; i < hash.length; i++) {
    String hex = Integer.toHexString(0xff & hash[i]);
    if(hex.length() == 1) hexString.append('0');
    hexString.append(hex);
}
...
```

## 5 PUBLISHING TO ANDS RIF-CS

### 5.1 WHAT IS RIF-CS PUBLISHING?

When a user has finished a report, he can decide to publish it (see 4.3 Workboards and User Stories). The published reports are available to the public on the Climate Smart Seaports application, but also available to be collected by ANDS.

To do so, ANDS provides an online service where a harvester can be configured to connect to a specific URL, which should respond with a specific XML format called RIF-CS (see Figure 34: RIF-CS publishing workflow diagram).

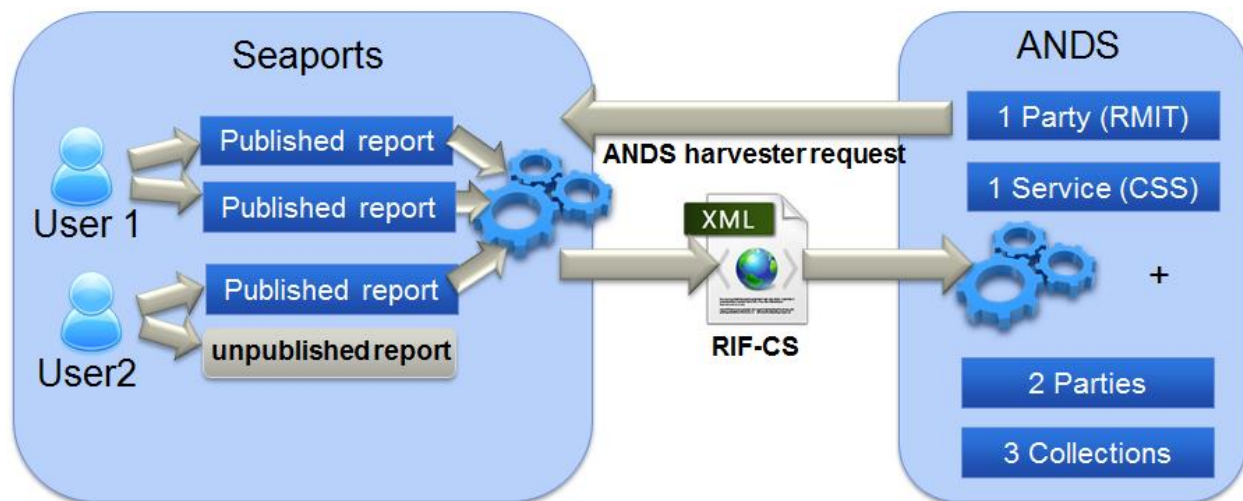


Figure 34: RIF-CS publishing workflow diagram

The ANDS online service can be accessed at: <https://researchdata.ands.org.au/registry/auth/login>

It is possible to log in using RMIT staff credentials. This online service allows managing the RIF-CS records, and configuring the automatic harvesting of records. The harvester needs to be configured to point to the URL: <http://climate.smart.seaport.server:8080/CSS/rif-cs>

The method “*getRifcsXML*” of the controller “*RIFCSController*” is responsible of handling that URL request, and unlike the other controllers which use views to generate HTML, it writes directly XML into an *HttpServletResponse* object. The result of the page /rif-cs is an XML file in the RIF-CS format defined by ANDS.

To generate the RIF-CS format, the controller is using a Java API provided by ANDS. For details about the setup and configuration of this API’s library, see section 3.3.1 Maven. This API gives

access to classes corresponding to the entities of the RIF-CS format, and enables constructing a hierarchy of these objects in Java.

It can then generate XML fully compliant with the version 1.3 of the RIF-CS format. Note that this is the latest API, but it doesn't generate the latest version of the RIF-CS format (latest is currently version 1.4). There are only minor changes and addition in the latest version, so the generated format is compatible even though some of the new fields can't be added.

## 5.2 FORMATTING FOR RIC-CS

This section describes which RIF-CS records are created based on the Climate Smart Seaports application's model. The following diagram comes from ANDS, and shows the 4 different types of records that can exist in RIF-CS, and all the relationships that can exist between them.

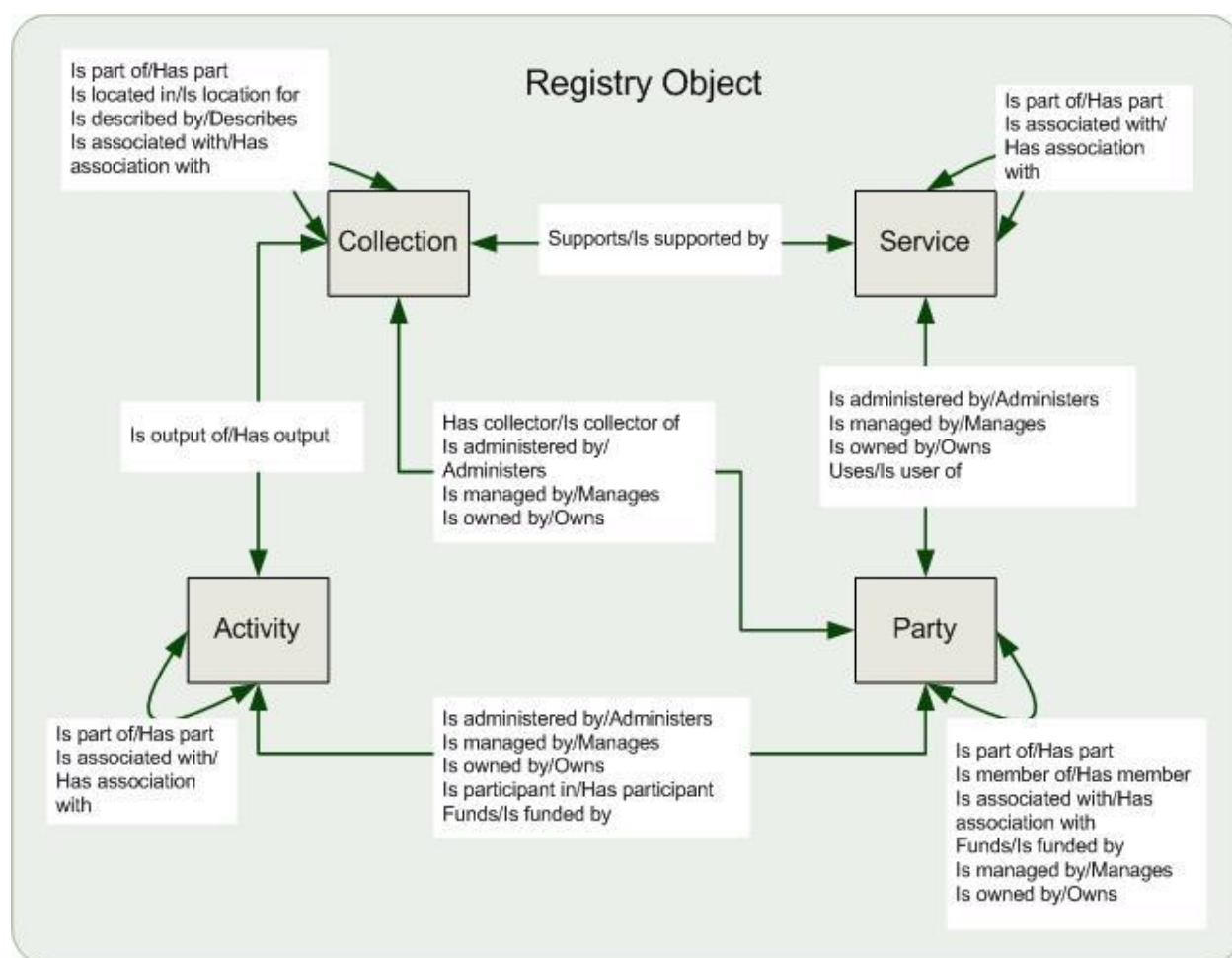


Figure 35: RIF-CS Registry Objects relationships - from ANDS

It is assumed that one is already familiar with the RIF-CS format. For more information, visit: <http://services.ands.org.au/documentation/rifcs/guidelines/rif-cs.html>



The diagram below shows the records generated by the Climate Smart Seaports application and the relationships created between them:

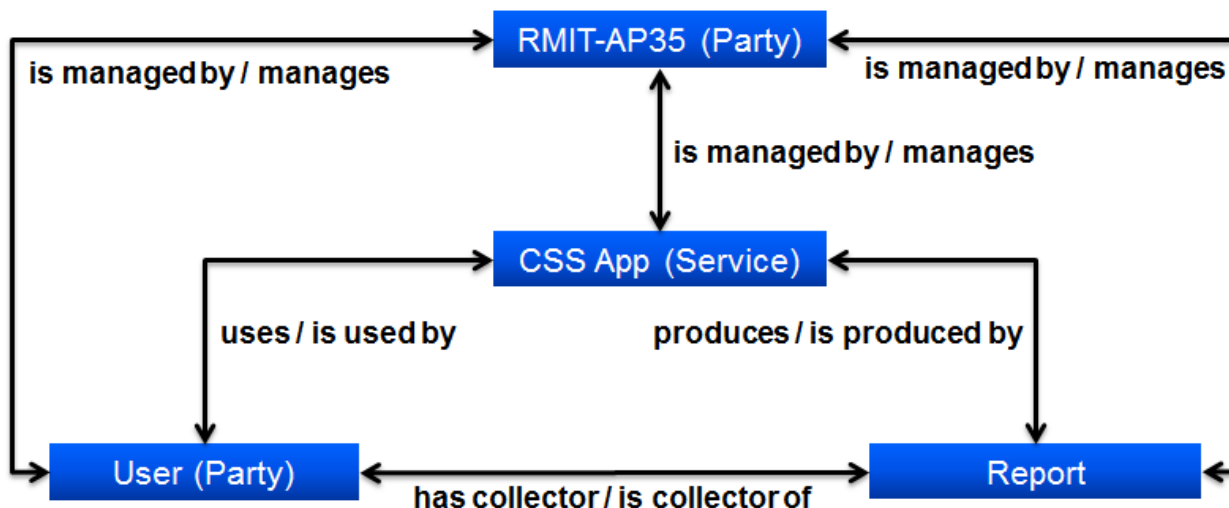


Figure 36: generated RIF-CS records and their relationships

Each published record generates a RIF-CS Collection record, and each of their authors generates a Party record. There are also a Service record corresponding to the application itself, and a Party record corresponding to the “RMIT AP-35” group, which manages every Service, Party and Collection record.

See the content of the *RIFCSController* for details about which property of which class corresponds to which RIF-CS record field.

### 5.3 RIF-CS USEFUL LINKS

ANDS Content Provider Guide:

<http://ands.org.au/guides/content-providers-guide.html>

RIF-CS v1.4 Schema Guidelines:

<http://services.ands.org.au/documentation/rifcs/guidelines/rif-cs.html>

Documentation of RIF-CS Java API 1.3 (Javadoc) :

<http://services.ands.org.au/documentation/rifcs/java-api-1.3/javadoc/>

ANDS Online Service:

<https://researchdata.ands.org.au/registry/auth/login>

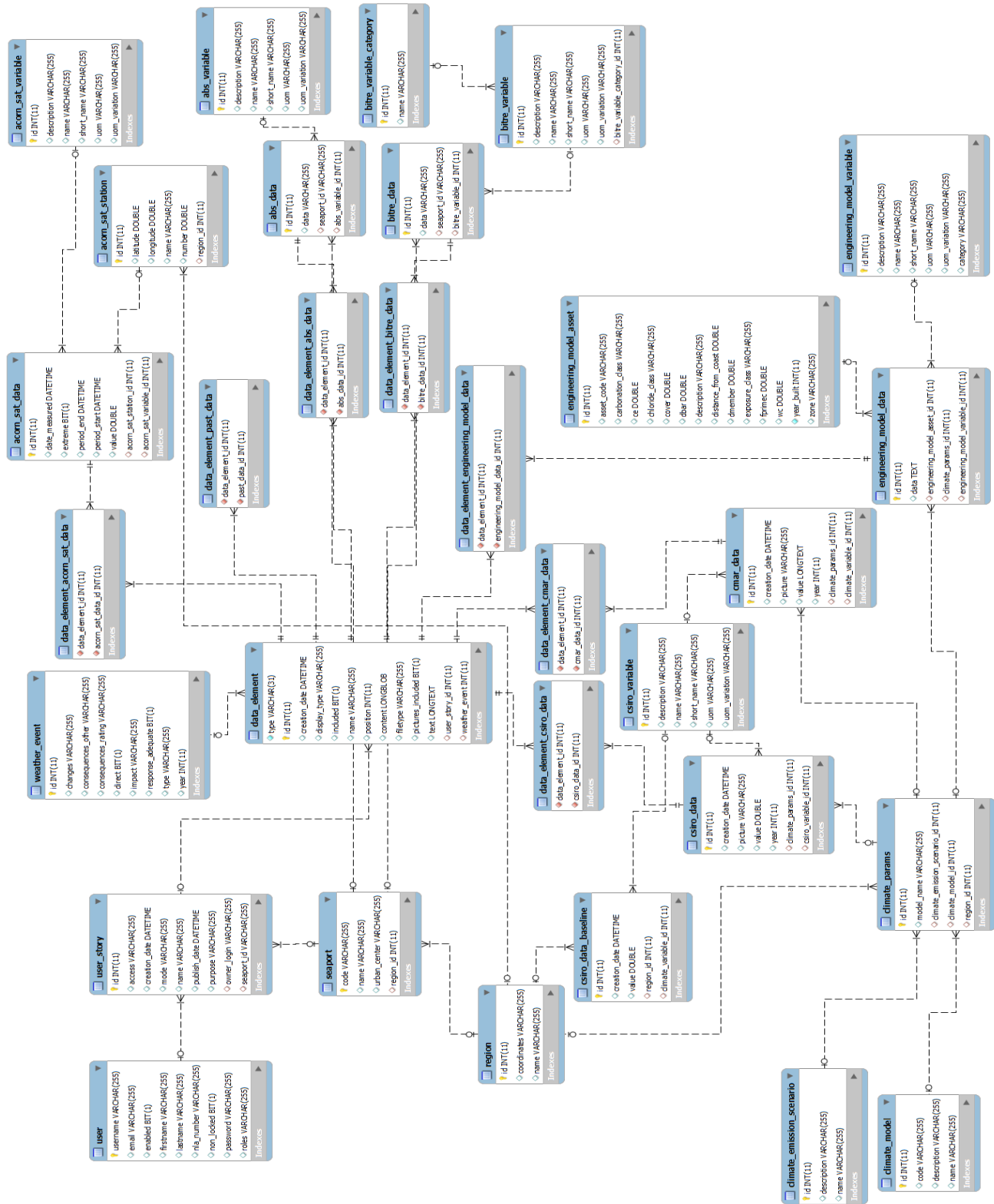
ANDS Demo Service (sandbox):

<https://demo.ands.org.au/registry/login.php>

Content Provider Quality Process and Gold Standards:

<http://www.ands.org.au/guides/cpguide/cpgquality.html>

## 6 ANNEX 1: FULL DATABASE DIAGRAM



**Figure 37: Full database schema diagram.**

## 7 ANNEX 3: GLOSSARY

**Administrator:** *User* with administration rights on the seaports application. An *Administrator* can access a list of all the users, remove users and block their account. He can also access, any user's *Workboard*.

**Analysis Stage:** stage of the workflow where the user organize *Data Elements* to format a Report and write his analysis by adding *Comments* to it. This stage comes after the *Data Gathering Stage*.

**Category:** a pre-defined category of *Data Element*. *Categories* appear as tabs in the *Workboard* page and allow filtering *Data Elements* based on their *Data Source*. The Categories are

**Comment:** a text (possibly formatted using HTML tags) added to a *Report* during the *Analysis Stage* containing analysis produced by the *User*.

**Climate Model:** a climate model interprets climate data to forecast the values of some climate variables such as temperature, wind speed, sea level, etc. Each *Region* has 3 references to *Climate Models*: one "Hotter & Drier", one "Most Likely" and one "Cooler & Wetter". Note that for different *Regions*, the same reference name can correspond to different *Climate Models* (example: the "Hotter & Drier" reference may lead to the *Climate Model X* for *Region A* and to the *Climate Model Y* for *Region B*).

**Data Element:** a piece of data from a *Data Source* that is added to a *Workboard* as an independent entity. A *Data Element* can be added or removed from a *Workboard* during the *Data Gathering Stage*, and later can be re-ordered, included or excluded from a Report during the *Analysis Stage*.

**Data Format:** the way a piece of data of a *Data Element* is displayed. The *Data Format* is usually directly related to the *Data Source* (for example, data from the *Engineering Model* is formatted as a Graph followed by a table).

**Data Gathering Stage:** stage of the *Workflow* where the user adds or removes *Data Elements* to the *Workboard*. In this stage, the application provides information about whether each *Category* is filled with at least one *Data Element* (it doesn't prevent to go to the next stage in the Workflow but is a good indication of how complete the *Workboard* is). This stage comes before the *Analysis Stage*.

**Data Source:** an available source of data to create *Data Elements* from. The available *Data Sources* are: CSIRO, BoM, BITRE, ABS, *Engineering Model*, and custom file.

**Draft:** an unpublished *Report* is called a *Draft* or a *Report Draft*. The *Drafts* are edited during the *Analysis Stage*. As soon as a *Draft* is *Published*, it becomes a *Report* and is not editable anymore.

**Emission Scenario:** a possible CO2 emissions scenario for the future. This is used as a parameter to compute the forecast the future evolution of climate variable. Two possible scenarios are taken into account into the Seaports project: a “High emissions” (corresponding to the CSIRO’s A1B scenario) and a “Medium emissions” (corresponding to the CSIRO’s A1FI scenario).

**Engineering Model:** refers to the NCCARF-funded concrete deterioration engineering model which allows computing a forecast for the deterioration of concrete port *Asset*. This *Engineering Model* is created a set of Excel template files which perform the computation and generate *Engineering Model Output*.

**Engineering Model Asset:** a port *Asset* that the *Engineering Model* is able to compute the future deterioration.

**Engineering Model Example:** a pre-defined example of Engineering Model Output already loaded into the Seaports application in order to allow a User to use example data instead of going through the all the process of the *Engineering Model Tool*.

**Engineering Model Tool:** Web application developed in .NET in order to provide a web interface to the *Engineering Model*. One can provide input for the *Engineering Model*, and get it to generate *Engineering Model Output* that can be downloaded.

**Engineering Model Output:** Excel file produced by the *Engineering Model* containing all the computed concrete deterioration data for a period of 70 years (2000 to 2070). This Engineering Model Output can be uploaded into the Seaports application as part of the “Engineering Model” *Data Source*.

**Private/Public:** adjective that relates to the privacy status of *Workboard*, *Drafts* and *Reports*. *Private* means that the *Report* can only be viewed by the *User* who created it (and by the *Administrator*). *Public* means that any *User* can view the *Report* and that it appears in the results of researches and listings of *Reports* in the Seaports application. A *Workboard* and *Drafts* are always *Private*, and a *Report* automatically becomes *Public* when it is *Published*.

**Publish:** action that finalizes a *Draft* to become a *Report*. It publishes the *Report* to ANDS (RIF-CS), prevents it to be edited or deleted, and makes it *Public* in the Seaports application.

**Region:** refers to a Natural Resources Management region of Australia. 3 NRM Regions are taken into account in the Seaports application: “East Coast South”, “Southern Slopes Vic East”, and “Southern and Southwestern Flatlands”.

**Report (formerly *User Story*):** A published Report which is submitted to ANDS and is Public within the Seaport application.

**Storyline:** combination of Climate Model, Emission Scenario and Region which is used as a parameter for the different forecast of future climate data (CSIRO) or future concrete deterioration (Engineering Model). Different storylines produce different data for a same year.

**User:** a user of the Seaports application, typically from port authorities.

**Workboard:** the entity containing *Data Elements* during the *Data Gathering Stage* of the *Workflow*.

**Toolbox:** column present on the side of the *Workboard* during the *Data Gathering Stage*, allowing the addition of new *Data Elements* to the *Workboard* and other available actions.

**Workflow:** the process followed by the *User* from starting a new *Workboard* to *Publishing a Report* to ANDS. The stages of the *Workflow* are: *Workboard creation*, *Data Gathering Stage*, *Analysis Stage*, *Publish*.