Estàndard d'aplicacions Java EE

Estàndards de desenvolupament

Palma, setembre de 2025

Vicepresidència Primera i Conselleria d'Economia, Hisenda i Innovació
Direcció General d'Estratègia Digital
i Desenvolupament Tecnològic

Índex

C/ de Sant Pere, 7
07012 Palma
Tel. 971 17 66 00
http://dgtic.caib.es

Historial de versions

Data	Versió	Descripció	Autor
05/04/16	7.2	Revisió d'estàndards	DGDT
17/01/20	9.0	Revisió d'estàndards	DGMAD
07/02/20	9.1	Correcció d'errors i aclariments	DGMAD
22/07/20	9.2	Eliminar versionat dels war i jar	DGMAD
03/12/20	9.5	Traducció al català i aclariments	DGMAD
23/03/21	9.6	Validació del framework Spring Separació de contextos API REST interna i externa	DGMAD
24/01/22	9.7	Correcció d'errors i aclariments	DGEDDT
08/09/22	9.8	Correcció error datasource	DGMAD
19/09/22	9.9	Altres correccions i aclariments	DGMAD
25/05/23	9.10	Nova secció de reutilització de dades	DGMAD
07/07/23	9.11	Correcció d'errors en l'exemple del datasource	DGMAD
19/09/25	9.12	Revisió de format i separació de la secció «7. Serveis Rest de reutilització de dades» en un nou document	DGEDDT

1. Introducció

Aquest document descriu l'estàndard de desenvolupament d'aplicacions Java Platform Enterprise Edition (Java EE) que s'ha de seguir per al desenvolupament d'aplicacions web que s'instal·laran en els servidors de la Direcció General d'Estratègia Digital i Desenvolupament Tecnològic (DGEDDT) del Govern de les Illes Balears (GOIB).

El document s'estructura en cinc grans apartats:

1. Normes de caràcter general (capítol 2).
2. Estructura del projecte (capítol 3).
3. Normativa i recomanacions sobre el codi font (capítol 4).
4. Aspectes de seguretat a les aplicacions (capítol 5).
5. Serveis web API REST (capítol 6).

Qualsevol tecnologia o decisió tècnica aliena a aquests estàndards de desenvolupament d'aplicacions haurà de ser consultada i aprovada per la DGEDDT. En cas contrari, la DGEDDT es reserva el dret a no acceptar el desplegament de l'aplicació en el servidors de la DGEDDT, sense perjudici d'altres accions derivades per l'incompliment del Decret 24/2022, d'11 de juliol, de la Comissió Superior de Sistemes d'Informació en Tecnologia i Comunicacions.

2. Normes de caràcter general

A continuació es detallen un conjunt de normes de caràcter general relacionades amb el desenvolupament d'aplicacions Java EE.

- A més de les especificacions descrites en aquest document, les aplicacions es desenvoluparan seguint les especificacions de la resta de documents dels estàndards de desenvolupament del GOIB, especialment les especificacions descrites en:
  - L'estàndard d'implantació d'aplicacions.
  - L'estàndard de base de dades.
  - L'estàndard de reutilització i obertura de dades.
- Les aplicacions hauran de desenvolupar els mòduls mitjançant aplicacions distribuïdes en tres nivells: interfície, lògica i dades; on la capa de presentació es pot subdividir en capa d'interfície gràfica i capa de control d'interfície; i la capa de dades en capa de persistència i capa de base de dades.
- Les aplicacions utilitzaran el patró de disseny Model-Vista-Controlador (MVC). La petició de l'usuari serà rebuda pel controlador de la capa de control d'interfície, el qual accedirà a la capa lògica de negoci implementat pels Enterprise Java Bean (EJBs) i els serveis REST. Els EBJs accediran a les entitats de persistència gestionades per JPA.
- L'ús de pàgines JSP està restringit a funcionalitats particulars i no es pot utilitzar com a tecnologia base de l'aplicació.
- Sota cap circumstància ni els controladors, ni els servlets, ni les pàgines JSF accediran de forma directa a la base de dades.
- Tot projecte haurà de publicar-se a un repositori git del GOIB segons la seva tipologia:
  i. Aplicacions de codi lliure: https://github.com/governib/
  ii. Aplicacions internes: https://git.caib.es/

- El producte final (i les posteriors actualitzacions) es lliurarà segons el model de quadern de càrrega de la DGEDDT (veure Document «Estàndards d'implantació d'aplicacions», Capítol 4. "Quadern de càrrega").

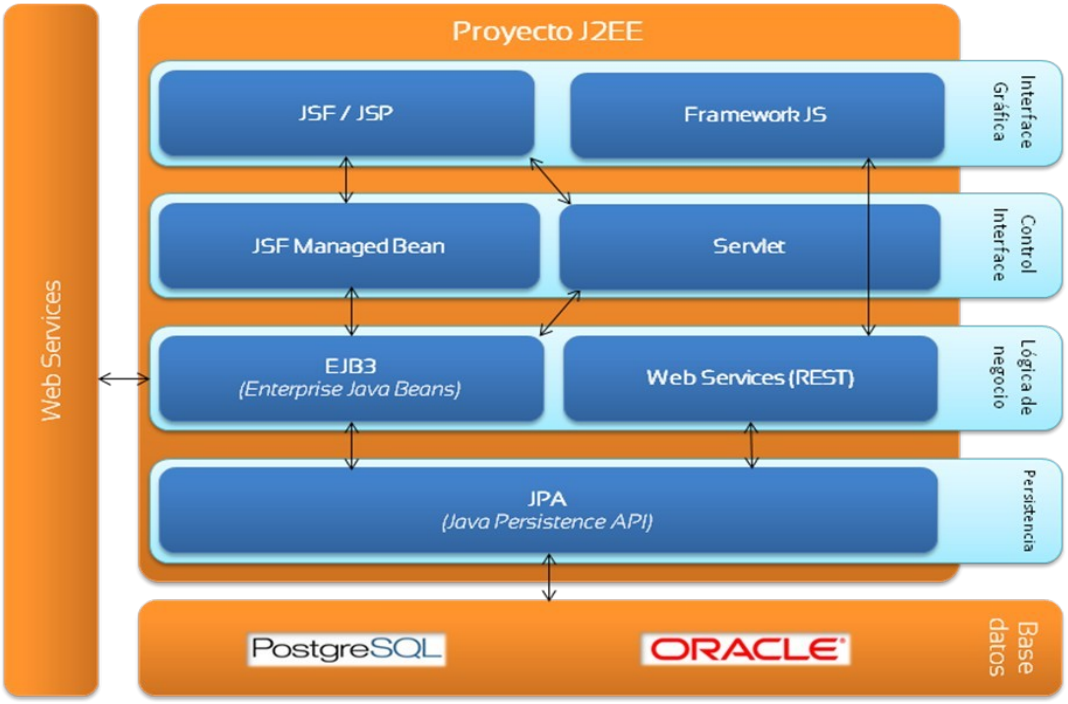


Figura 1: Patró de disseny Model-Vista-Controlador

- El sistema haurà de complir les mesures i criteris de seguretat designades en la legislació vigent, especialment en:

  i. La Llei orgànica 3/2018, de 5 de desembre, de Protecció de Dades Personals i garantia dels drets digitals, junt amb el Reglament (UE) 2016/679 del parlament europeu i del consell de 27 d'abril de 2016 relatiu a la protecció de les persones físiques pel que fa al tractament de dades personals i a la lliure circulació d'aquestes dades i pel qual es deroga la Directiva 95/46/CE (Reglament General de Protecció de Dades).

  ii. El Reial decret 3/2010, de 8 de gener, pel qual es regula l'Esquema Nacional de Seguretat en l'àmbit de l'Administració Electrònica (ENS).

  iii. La Llei 34/2002, d'11 de juliol, de serveis de la societat de la informació i de comerç electrònic (LSSI).

- D'acord amb el Reial decret 1112/2018, sobre accessibilitat dels llocs web i aplicacions per a dispositius mòbils del sector públic, s'exigeix que qualsevol aplicació web del Govern de les Illes Balears compleixi amb el nivell de conformitat AA de l'estàndard internacional WCAG (Web Content Accessibility Guidelines) en la seva versió 2.1.
- A més de la documentació generada en el desenvolupament, s'haurà de generar (o actualitzar si ja existeix) el manual d'usuari final, el manual d'administrador, el manual d'integració per tercers (si procedeix) i el manual d'instal·lació de l'aplicació. El manual d'usuari s'haurà d'integrar dins la interfície web de cada eina afegint-hi recursos multimèdia per millorar l'accessibilitat al seu contingut.

2.1. Especificacions tecnològiques.

A **nivell general**, les aplicacions hauran d'implementar-se utilitzant la versió lliure de la plataforma de desenvolupament de Java **OpenJDK 11** i executar-se sobre un servidor d'aplicacions **JBoss EAP 7.2** sense modificar (exceptuant els paràmetres descrits en aquest document).

Respecte a la **capa de presentació**:

- S'utilitzarà preferentment **JSF 2.3** (Java Server Faces) junt amb la biblioteca de components **Primefaces**.
- El punt anterior podrà substituir-se per un framework que utilitzi el patró de disseny Model-Vista-Controlador (MVC) i que hagi estat validat prèviament per la DGEDDT; actualment: **Spring**, **Angular** i **React**.
- No es permet l'ús de pàgines JSP com a tecnologia base de l'aplicació. El seu ús només està justificat per implementar funcionalitats molt concretes que ho requereixen.
- La interfície d'usuari haurà de ser compatible amb la darrera versió estable dels navegadors **Firefox ESR** i **Chrome**.

Respecte a la **capa de lògica de negoci**:

- S'utilitzarà preferentment **EJBs 3.2** i **serveis REST**.
- Els EJBs podran substituir-se pels mòduls corresponents dels frameworks utilitzats en la capa de presentació.

Respecte a la **capa de persistència de dades**:

- Les aplicacions hauran d'estar preparades per funcionar sobre una base de dades **Oracle 19.3**.
- La necessitat d'utilització d'altres bases de dades com **PostgreSQL 10** haurà de venir motivada i ser aprovada per part del responsable de l'Àrea d'Administració de Base de Dades a causa de les limitacions de suport i alta disponibilitat.
- L'accés a les dades es durà a terme mitjançant l'API **JPA 2.2**.
- El punt anterior podrà substituir-se pel mòdul corresponent del framework utilitzat en la capa de presentació (com per exemple Spring Data JPA inclòs en Spring).

Respecte a l'**autenticació d'usuaris** per accedir als recursos protegits:

- Si són usuaris interns del GOIB, es realitzarà mitjançant el mecanisme d'autenticació centralitzat RedHat Single Sign-On 7.3 basat en **Keycloak** (per a més informació, veure secció «5. Seguretat»).
- Qualsevol altre autenticació externa es realitzarà mitjançant el component horitzontal **LoginIB** (veure el document «Catàleg de Serveis»).

Finalment, tot projecte haurà d'estar desenvolupat mitjançant **Maven 3.6 o superior** per a la construcció del projecte (compilació i empaquetat) i la gestió de dependències.

En funció de criteris de manteniment i disponibilitat de versions el *Centre de Processament de dades* de la DGEDDT es reserva la facultat d'actualitzar les versions del programari aquí exposades per altres superiors en el moment de la posada en producció.

En cap cas es podrà utilitzar cap framework, llibreria o utilitat amb llicència propietària.

La imatge següent resumeix les especificacions descrites anteriorment.

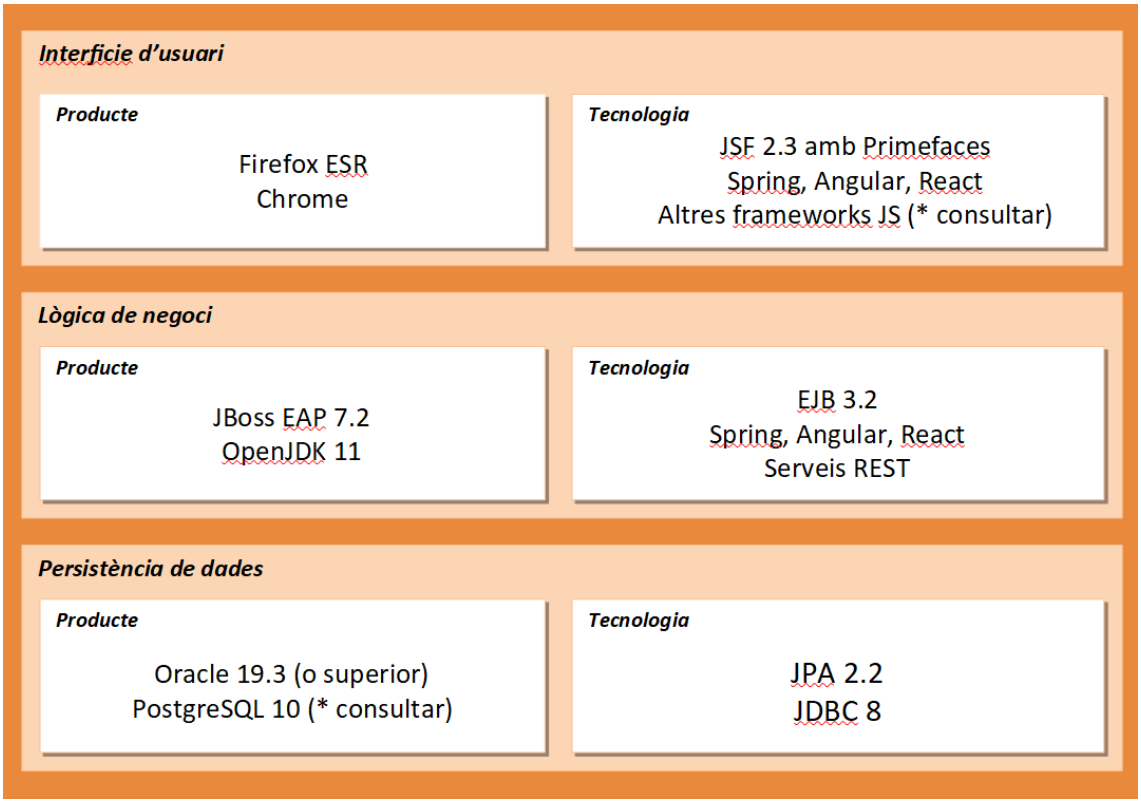


Figura 2: Especificacions tecnològiques.

3. Estructura del projecte

Partint de la base que tots els projectes han d'utilitzar Maven, l'estructura de directoris serà la següent:

```
[Proyecto]/
    ├── pom.xml
    ├── doc/
    ├── [Módulos]/
    │     ├── [Módulo_1]/
    │     │     ├── pom.xml
    │     │     ├── src/
    │     │     │     ├─ main/           (Código del módulo)
    │     │     │     │    ├── java/     (Clases java)
    │     │     │     │    ├── resources/ (Ficheros de configuración)
    │     │     │     │    └── webapp/   (Ficheros web)
    │     │     │     └── test/          (Tests unitarios)
    │     │     └── target/              (Ficheros entregables)
    │     ├── ...
    │     └── [Módulo_N]/
    ├── scripts/
    │     ├── bbdd/
    │     ├── config/
    │     ├── datasources/
    │     ├── templates/
    │     └── install/
    └── target/ (Deberá contener los entregables del proyecto)
          └── doc/
```

[Projecte]

És el mòdul pare de la resta de mòduls del projecte. El nom del directori haurà de ser el mateix que el nom del projecte (per exemple, *projectebase*).

[Projecte]/pom.xml

Fitxer *Maven* encarregat de la configuració, compilació i empaquetat de tot el projecte. S'encarregarà d'executar els fitxers *pom.xml* dels mòduls del projecte i deixar els fitxers a lliurar en la carpeta [Projecte]/target.

El fitxer *pom.xml* del mòdul pare haurà de tenir ***packaging*** tipus ***POM*** i deurà contenir a la resta de mòduls i les seves propietats (*properties*).

A continuació es mostra un exemple de fitxer *pom.xml* del mòdul pare:

```
<project xmlns="http://maven.apache.org/pom/4.0.0"
xmlns:xsi="http://www.w3.org/2001/xmlschema-instance"
xsi:schemaLocation="http://maven.apache.org/pom/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>es.caib.projectebase</groupId>
<artifactId>projectebase</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>pom</packaging>
<name>Projecte Base</name>
<description>Projecte Base amb OpenJDK11</description>
[...]
<properties>[...]</properties>
[...]
<module>
<module>projectebase-api-interna</module>
<module>projectebase-api-externa</module>
<module>projectebase-back</module>
<module>projectebase-commons</module>
<module>projectebase-ear</module>
<module>projectebase-ejb</module>
<module>projectebase-front</module>
<module>projectebase-persistence</module>
<module>projectebase-service</module>
</module>
[...]
</project>
```

[Projecte]/README.md

Document simple on s'han de detallar els requisits, configuracions i accions per poder compilar, provar i instal·lar el projecte des de zero.

[Projecte]/doc

En aquest directori se situaran els fitxers de documentació del projecte (diagrames, manuals d'usuari, manuals d'administració, manuals d'instal·lació, configuracions o relacions amb altres productes...). Preferentment estaran en format *LibreOffice*. També s'hauran de situar en aquesta carpeta els fonts utilitzats per a la generació dels documents, imatges, gràfics, formats natius,..

La documentació específica de cada mòdul, en cas de ser necessària, se situarà en subcarpetes amb el mateix nom que els mòduls.

[Projecte]/doc/pdf

Aquí es deixaran els documents anteriors convertits en format PDF. Són les versions finals dels documents.

[Projecte]/[Mòduls]/[Mòdul N]

Els projectes hauran d'estar formats (com a mínim) pels següents mòduls:

a) **Un mòdul pare** amb *packaging* tipus ***POM*** que haurà de contenir la resta de mòduls.

b) Un o diversos **mòduls Web (*Frontal / Backoffice*)** amb *packaging* ***WAR*** que continguin tot el codi de la capa de presentació: controladors d'interfície i fitxers d'interfície. Els noms dels fitxers generats seran ***codiAplicació-front.war*** i ***codiAplicació-back.war***.

c) Un mòdul amb la **lògica de negoci** i un mòdul amb la **capa de persistència** amb *packaging* ***JAR.*** El nom del fitxer generat serà ***codiAplicació.jar***.

d) Un mòdul d'**empaquetament** de la resta de mòduls amb *packaging* tipus ***EAR***. El nom del fitxer generat serà ***codiAplicació.ear*** i contindrà la resta de fitxers de l'aplicació, inclosos el fitxers war i jar anteriors.

La nomenclatura dels mòduls *MAVEN* serà la següent:

- **codiAplicació-api-interna**: serveis REST d'ús intern per integracions amb altres aplicacions.
- **codiAplicació-api-externa**: serveis REST susceptibles de publicar-se en Internet.
- codiAplicació-**back**: interfície web pel back-office.
- codiAplicació-**front**: interfície web pel front-office.
- codiAplicació-**commons** classes i mètodes amb funcionalitat comú a tots els mòduls.
- codiAplicació-**ear**: capa per a l'empaquetament de la resta de mòduls.
- codiAplicació-**service**: capa de lògica de negoci.
- codiAplicació-**ejb**: serveis per accedir a la capa de persistència.

- codiAplicació-**persistence**: emmagatzemament i consulta de dades.
- codiAplicació-**lib**: repositori local de llibreries externes. Utilitzant Maven no hauríem d'utilitzar aquest mòdul; si bé és cert que algunes llibreries no estan disponibles en repositoris externs i en ocasions no queda més remei que afegir-les manualment.

[Projecte]/[Mòduls]/[Mòdul N]/pom.xml
Fitxer *Maven* encarregat de configurar, compilar i empaquetar el mòdul.

[Projecte]/[Mòduls]/[Mòdul N]/src
Contindrà el codi font del mòdul corresponent.

[Projecte]/[Mòduls]/[Mòdul N]/src/main/java
Contindrà les classes Java i les interfícies del mòdul, L'estructura de la paqueteria ve definida en la secció 4.2.1. *Packages*.

[Projecte]/[Mòduls]/[Mòdul N]/src/main/resources
Contindrà recursos necessaris per a l'aplicació (*properties*, *xml*, imatges,...).

[Projecte]/[Mòduls]/[Mòdul N]/src/main/webapp
Contindrà tots els fitxers necessaris per especificar la interfície web del projecte. Estarà format principalment per fitxers *JSF*, plantilles *HTML*, *CSS*, *Javascript*, imatges, fitxers de configuració, etc. L'estructura bàsica del mòdul web haurà de ser:

```
webapp/
    ├── css/
    ├── img/
    ├── js/
    ├── WEB-INF/
    │       ├── web.xml
    │       ├── jboss-web.xml
    │       ├── lib/
    │       └── ...
    └── ...
```

[Projecte]/[Mòduls]/[Mòdul N]/src/test/
Contindrà els tests unitaris per provar el codi Java situat en «[Projecte]/[Mòduls]/[Mòdul N]/src/main/java».

[Projecte]/[Mòduls]/[Mòdul N]/target
En aquest directori es guardaran els fitxers resultats de la compilació i empaquetament del mòdul.

[Projecte]/scripts/

Contindrà els scripts de l'aplicació.

[Projecte]/scripts/bbdd/

Contindrà tots els *scripts sql* de base de dades, tant per a la creació i actualització dels objectes (sentències DDL), com els *scripts* d'inserció, eliminació i actualització de dades (sentències DML). El format i estructuració d'aquests *scripts* seguiran el format del Capítol 3.5 del document «Estàndards d'implantació d'aplicacions». El contingut dels scripts haurà de complir les normes establertes en el document «Estàndards de base de dades».

[Projecte]/scripts/bbdd/complet

Scripts sql de creació de la base de dades des de zero a la darrera versió.

[Projecte]/scripts/bbdd/[versió]

Scripts sql incrementals per a l'actualització de la base de dades des de la versió immediatament anterior a la versió corresponent.

[Projecte]/scripts/config/

Scripts de configuració del sistema i del producte.

[Projecte]/scripts/datasources/

Fitxers utilitzats per *JBoss* per accedir a la base de dades. Aquests fitxers contindran la informació necessària per connectar-se a una base de dades: controlador *(driver),* gestor de base de dades, servidor (host), port, nom de la base de dades, usuari i contrasenya.

[Projecte]/scripts/templates/

En aquest directori es guardaran les plantilles per generar fitxers. Per exemple, per generar els fitxers de versió que s'expliquen en el punt 4.3.

[Projecte]/target/

Directori on se situarà el producte final encapsulat a desplegar.

[Projecte]/target/doc

En aquest directori s'hauran de situar els documents associats al projecte empaquetat.

4. Normativa

4.1. Estil i format del codi

Remarques:

- S'ha de documentar cada fitxer Java segons l'especificació JavaDoc. Abans de la declaració de les classes, s'ha d'informar de la utilitat de les classes, autors i versions. Al mètodes s'indicarà la seva utilitat, les variables d'entrada i sortida i les possibles excepcions.
- Es recomana no fer servir tabulacions. La tabulació es realitzaria mitjançant quatre espais. Es poden configurar els IDEs de programació perquè emulin la tabulació.
- Es recomana no superar els 120 caràcters per línia.
- Es recomana no excedir les 1000 línies de codi per fitxer. L'objectiu d'aquesta mesura és distribuir el codi de forma equitativa en els fitxers i dividir conceptualment el codi de forma més efectiva.
- Es recomana obrir el claudàtor en la mateixa línia que la sentència i tancar-lo en una línia posterior.

  Exemple d'utilització:

```
public class Exemple {
   public void getMetode(){
        for(int x=0; x < 10; x++){
           while(...) {
             if (x < 5) {
                ...
             } else {
                ...
             } // Final if-else
           } // Final while
        } // Final for
   } // Final mètode
} // Final classe
```

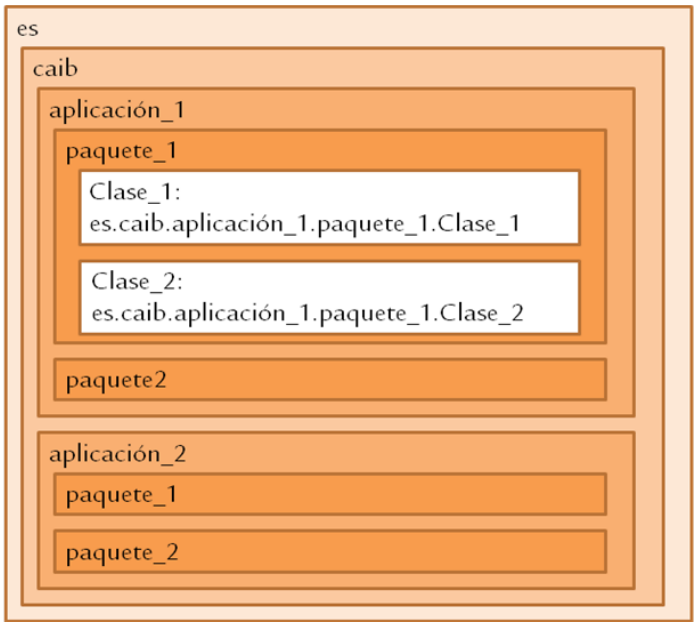
4.2. Nomenclatura dels objectes

4.2.1. Nomenclatura i jerarquia de paquets

Les classes d'objectes s'estructuraran en paquets (*packages*).

Els noms dels paquets estaran sempre en minúscules i podran ser nomenats, dins del paquet d'aplicació, a criteri d'analistes i dissenyadors.

Totes les aplicacions hauran de tenir com a mínim quatre nivells: els dos primers nivells seran sempre: ***es.caib***; el tercer nivell indicaran el nom de l'aplicació; i el quart nivell el mòdul. Així, les classes es denominaran *es.caib.codiAplicació.paquet.Classe*, tal com pot observar-se en la següent il·lustració:



El codi d'aplicació serà subministrat pel Centre de Processament de Dades de la DGEDDT (veure document «Estàndards d'implantació d'aplicacions», Capítol 2. «Sol·licitud de codi d'aplicació»).

4.2.2. Nomenclatura de classes

Respecte al nom de les classes:

- Estarà format per una o diverses paraules concatenades que descriguin la funcionalitat de la classe.

- La primera lletra de cada paraula ha d'estar en majúscula i la resta en minúscules.
- No han d'aparèixer guions.
- Ha de tenir més d'un caràcter.

Exemples: Empresa, Client, CentresDeTreball,PersonaPerMunicipi,...

4.2.3. Nomenclatura de mètodes

Respecte al nom dels mètodes:

- Estarà format per una o diverses paraules concatenades que descriguin la funcionalitat del mètode.
- La primera paraula ha d'estar tota en minúscules.
- La primera lletra de cada paraula interna (a partir de la segona) ha d'estar en majúscula.
- No han d'aparèixer guions.
- S'han de seguir la convenció de Java d'un prefix més un nom descriptiu. El prefix (opcional si el nom és molt simple) serà: «is», «get», «set», «add», «del», «init», «close»,..
- Ha de tenir més d'un caràcter.

Exemples: connect(), isAdministrator(), getState(), addUser(), initDades(),...

4.2.4. Nomenclatura d'atributs i variables

Respecte al nom dels atributs i variables:

- Estarà format per una o diverses paraules concatenades que descriguin la funcionalitat de la variable.
- La primera paraula ha d'estar tota en minúscules.
- La primera lletra de cada paraula interna (a partir de la segona) ha d'estar en majúscula.
- No han d'aparèixer guions.
- Ha de tenir més d'un caràcter.

Exemples: total, municipi, nomSocial, dataNotificacio,...

4.2.5. Nomenclatura de constants

Respecte al nom de les constants:

- Estarà format per una o diverses paraules concatenades que descriguin la funcionalitat de la constant.
- Totes les paraules en majúscules.
- Guions baixos per separar paraules.
- Ha de tenir més d'un caràcter.

Exemples: AMPLADA, URL_SERVIDOR, USUARI_BD,...

4.2.6. Nomenclatura de serveis i interfícies

Respecte al nom dels serveis i interfícies (interface):

- Es seguirà la mateixa nomenclatura que la resta de classes, afegint al final el literal «EJB» per a la implementació del servei i el literal «Service» per la definició de la interfície.

Exemples: EmpresaEJB i EmpresaService, CentresDeTreballEJB i CentresDeTreballService.

4.2.7. Nomenclatura de tests unitaris

Respecte al nom dels tests unitaris:

- Es seguirà la mateixa nomenclatura que la resta de classes, afegint al final el literal «Test».

Exemples: EmpresaService i EmpresaServiceTest, CentresDeTreballService i CentresDeTreballServiceTest.

4.2.8. Serveis de directori de servidor d'aplicacions

L'accés al servei de directori de noms (NamingFactory) es realitzarà sempre amb els paràmetres per defecte, assumint que les propietats JNDI estan correctament definides.

Els serveis de directori del servidor transaccional identificaran cada Enterprise Java Bean mitjançant el seu nom jeràrquic complet, havent d'accedir les classes Java a ell mitjançant aquest nom.

4.3. Accés a bases de dades

L'accés a bases de dades es realitzarà a través de la capa de persistència del mòdul **codiAplicació-persistence**. En concret, a través de la unitat de persistència (persistence-unit) definida en el fitxer **persistence.xml**.

A continuació es mostra un exemple complet de fitxer «persistence.xml» per l'aplicació «projectebaseexemple».

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd" version="2.2">
   <persistence-unit name="projectebaseexemplePU" transaction-type="JTA">
      <jta-data-source>java:jboss/datasources/projectebaseexempleDS</jta-data-source>

      <!-- Llista de classes persistents -->
      <class>es.caib.projectebaseexemple.persistence.UnitatOrganica</class>
      <class>es.caib.projectebaseexemple.persistence.Procediment</class>

      <!-- Llista de propietats -->
      <properties>
         <property name="hibernate.dialect" value="org.hibernate.dialect.OracleDialect"/>
         <property name="hibernate.showSql" value="false[1]"/>
      </properties>
   </persistence-unit>
</persistence>
```

Les dades de connexió (servidor, port, usuari, contrasenya,..) es definiran a un fitxer *datasource* independent amb nomenclatura ***codiAplicació-ds.xml***.

Dins del fitxer *datasource*, s'han de definir els següents paràmetres:

- El codi del *datasource* per registrar-lo a la interfície de noms de directori de Java (**jndi-name**). Aquest codi ha de seguir la

1 En entorns de producció aquesta propietat ha d'estar definida sempre com a «false».

nomenclatura «java:jboss/datasources/codiAplicacióDS» (exemple: «java:jboss/datasources/projectebaseexempleDS»).

- L'adreça de connexió (**connection-url)**, indicant: el tipus de client[2], el servidor, el port i el nom de la instància de la base de dades -SID- (exemple: «jdbc:oracle:thin:@localhost:1521:projectebaseexemple»).
- El tipus de controlador (**driver**): «oracle» o «postgresql».
- Les credencials (**security**) de l'usuari (**user-name** i **password**) de base de dades. El nom haurà de seguir la nomenclatura «WWW_CODIAPLICACIO».
- L'esquema per defecte (**new-connection-sql**).

A continuació es mostra un exemple complet de fitxer projectebaseexemple-ds.xml per l'aplicació «projectebaseexemple».

```
<datasource jndi-name="java:jboss/datasources/projectebaseexempleDS" pool-
name="projectebaseexempleDS">
<connection-url>jdbc:oracle:thin:@localhost:1521:projectebaseexemple</connection-url>
        <driver>oracle</driver>
        <security>
          <user-name>WWW_PROJECTEBASEEXEMPLE</user-name>
          <password>contrasenya</password>
        </security>

        <new-connection-sql>
        BEGIN
        EXECUTE IMMEDIATE 'ALTER SESSION SET CURRENT_SCHEMA =
        PROJECTEBASEEXEMPLE';
        END;
        </new-connection-sql>
</datasource>
```

4.4. Versionat de codi

4.4.1. Mostrar la versió del producte durant l'execució del producte

Per mostrar la informació de versió de l'aplicació en logs i pàgines serà obligatori generar les classes necessàries. Per a això cal:

2 En *Oracle*, l'accés ha de fer-se utilitzant el client «thin» (no «OCI»)

Crear la plantilla de la classe del versionat

Crear el fitxer ***Version.java.template*** en [Projecte]/scripts/templates amb el següent contingut:

```
package es.caib.projectebase;

// Codi autogenerat per la Version.java.template.
public final class Version {
	public static final String VERSION="@project.version@";
	public static final String BUILDTIME="@project.buildtime@";
	public static final String VCS_REVISION="@vcs.revision@";
	public static final String JDK_VERSION="@jdk.version@";
}
```

Fer referència a la versió en les pàgines o classes Java

Per fer referència a la versió i data de generació de la versió actual en les pàgines es pot fer invocant qualque servlet o controlador que utilitzi la classe Version.java generada anteriorment i que introdueixi en el peu de la pàgina la versió i la data de generació del producte.

Quant a les referències en classes Java es podrà fer utilitzant Version.VERSION, sempre que s'hagi definit la classe Version.java a partir de la plantilla Version.java.template.

4.4.2. Mostrar la versió en el log *d'aplicació*

Durant la posada en marxa de l'aplicació s'haurà d'imprimir un registre (utilitzant el logger a nivell INFO) amb el nom del producte i la versió que s'està executant així com la data de generació. Per exemple:

```
log.info("Carregant l'aplicació PROJECTEBASE versió "+Version.VERSION+ " generada en data: "+Version.BUILDTIME);
```

4.5. Propietats de configuració

Les aplicacions hauran de fer ús de propietats per parametritzar diferents aspectes de l'aplicació mitjançant l'ús de dos fitxers *.properties:*

- ***codiAplicació.properties***: de propietats internes de cada aplicació. La DGEDDT en cap cas es farà càrrec d'afegir propietats o modificar els valors

d'aquest fitxer. Si s'ha de modificar, s'haurà d'enviar el fitxer sencer que ja incorpori els canvis respecte al fitxer anterior.

- ***codiAplicació.system.properties***: de propietats configurables per part de la DGEDDT. La DGEDDT s'encarregarà de configurar aquest fitxer que s'usarà exclusivament per a les següents propietats (d'ús opcional):
  - **es.caib.codiAplicació.fitxers**: prèvia sol·licitud, apuntarà a un directori en el qual l'aplicació podrà guardar fitxers. En aquest directori, l'aplicació podrà crear l'estructura que millor s'adapti a les seves necessitats. Si fos necessari crear una estructura amb diferents subdirectoris, aquests hauran de crear-se sota aquesta ruta i s'hauran de definir les propietats necessàries relatives al directori basi apuntat per es.caib.codiAplicació.fitxers en el fitxer codiAplicació.properties.
  - **es.caib.codiAplicació.int.nomIntegració.usuari** i
  - **es.caib.codiAplicació.int.nomIntegració.secret**: per als casos en què l'aplicació hagi de realitzar trucades a webservices, la DGEDDT configurarà l'usuari i la contrasenya, respectivament en aquestes propietats. En el cas d'usuaris d'aplicació, nomIntegració té el format aplicacióOrigen_aplicacióDesti.
  - **es.caib.codiAplicació.int.nomIntegració.endpoint**: Url que apunta al servei d'aplicació que es desitja cridar.
  - **es.caib.codiAplicació.int.nomIntegració.path**: Ruta completa cap a la integració desitjada. Ej: ruta cap als fitxers estàtics: es.caib.codi.int.staticfiles.path=C:/Desarrollo.

Les aplicacions podran carregar els fitxers de *properties* a través de les propietats de sistema:

- es.caib.codiAplicació.properties
- es.caib.codiAplicació.system.properties

A continuació es mostra un exemple de codi per carregar les propietats dels diferents fitxers *properties:*

```
try (InputStream input = new
FileInputStream(System.getProperty("es.caib.codiAplicacio.properties")) {
        Properties prop = new Properties();
```

```
        // carrega propietats
        prop.load(input);
        // obté el valor de cada propietat
        String x = prop.getProperty("es.caib.codiAplicacio.xxxx");
        String i = prop.getProperty("es.caib.codiAplicacio.yyyy");
        String z = prop.getProperty("es.caib.codiAplicacio.zzzz");
} catch (IOException ex) {
        ex.printStackTrace();
}
```

Les propietats tindran com a prefix es*.caib.codiAplicació.* Per exemple, el contingut del fitxer *codiAplicació.properties* podria ser:

```
es.caib.codiAplicacio.propietat1=valor1
es.caib.codiAplicacio.propietat2=valor2
es.caib.codiAplicacio.propietat3=valor3
```

4.6. Codificació

Remarques:

- El fitxers .java hauran d'utilitzar el joc de caràcters UTF-8.
- S'haurà d'afegir la següent etiqueta en tots els fitxers XML:

  <?xml version="1.0" encoding="UTF-8"?>.
- S'haurà d'afegir la propietat següent en el fitxer pom.xml del projecte Maven pare:

  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
- En els cas de les APIs REST, s'haurà d'fegir la codificació a la classe que genera la sortida del servei REST seguint el següent exemple:

  String value = new String(myString.getBytes("UTF-8"), "UTF-8");

4.7. Implantació de bones pràctiques en el codi.

L'equip de desenvolupament de l'aplicació haurà d'aplicar regles de bones pràctiques generalment reconegudes amb l'objectiu de reduir l'aparició d'errors en el codi quan el sistema es porti a producció.

Algunes d'aquestes bones pràctiques passen per:

- Prioritzar la llegibilitat del codi.
- Minimitzar l'acoblament (dependència) entre mòduls.
- Generar i executar tests unitaris de totes aquelles classes que incloguin lògica de negoci.
- Evitar l'ús de codi redundant.
- Tancar correctament els fitxers i connexions quan ja no es necessitin.
- Eliminar variables, constants i importacions de classes que no s'utilitzin.
- Establir un correcte sistema de control d'errors.
- Documentar i comentar adequadament el codi.

Actualment, la majoria d'entorns de desenvolupament integrat (Eclipse, Intellij, Netbeans,..), inclouen eines natives o extensions que faciliten la labor del desenvolupador en aquest sentit (per exemple SonarLint).

4.8. Restriccions addicionals

- Els missatges de depuració generats per l'aplicació hauran d'aparèixer en el fitxer de log de l'aplicació amb categoria «DEBUG» (mai «INFO» o «WARN»).
- Els EJBs seran preferentment de sessió sense estat (stateless session beans).
- Els EJBs de sessió amb estat (stateful session beans) hauran d'implementar adequadament els mètodes activate i passivate a efecte de minimitzar el consum de memòria i recursos.
- Per a totes aquelles funcionalitats no especificades en aquest document, es recomana la utilització dels estàndards Java EE en la seva versió 8, evitant en la mesura del possible solucions particulars que només funcionin sobre JBoss.

5. Seguretat

Tots els aspectes relatius a identificació i autorització dels usuaris a servlets, serveis REST, controladors, pàgines JSF, o EJBs estaran gestionats de forma externa a les aplicacions. Per tant, no s'ha de codificar dins l'aplicació cap regla o criteri d'autenticació.

En cas que l'aplicació requereixi restringir l'accés als recursos concrets hauran de configurar-se els elements security-constraint, login-config i security-role del fitxer web.xml.

5.1. Autenticació

A partir de la versió EAP 7.2 de JBoss, l'autenticació per accedir als recursos protegits dels mòduls web s'ha de realitzar mitjançant el mecanisme d'autenticació centralitzat RedHat Single Sign-On (Keycloak). El propi servidor de l'aplicació (JBoss), en rebre una petició d'accés a un recurs que requereixi autorització, s'encarregarà de redirigir a l'usuari a la pàgina d'autenticació.

Aquest mòdul d'autenticació permet el control d'accessos a les aplicacions a nivell mòdul web (fitxer .war). Per tant, tota aplicació haurà d'estar separada en diferents mòduls que requereixin diferents mecanismes d'autenticació. Aquests es configuren en Keycloak com clients i s'indiquen en el fitxer standalone.xml del JBoss com a resources.

Per utilitzar el mètode d'autenticació és necessari instal·lar l'adaptador de RedHat Single Sign-On sobre Jboss. Per a això cal descarregar l'adaptador de RH-SSO 7.3 i aplicar-ho a la instància de JBoss:

```
$ cd $JBOSS_HOME\bin
$ unzip rh-sso-7.3.0.ga-eap7-adapter.zip
$ ./jboss-cli.sh –file=adapter-install-offline.cli
```

Aquest adaptador instal·larà el subsistema de *Keycloak* dins del fitxer ***standalone.xml*** de JBoss.

Els paràmetres principals de **configuració** del subsistema d'autenticació són els següents:

- **real-name** i **realm**: Nom del domini de confiança definit a Keycloak (normalment posarem «GOIB»).
- **auth-server-url**: Adreça del servidor de Keycloak. Generalment serà una adreça d'un servidor local (el servidor de desenvolupament de la DGEDDT està destinat per les aplicacions corporatives). Per més informació veure document «Guia de configuració de l'entorn de desenvolupament».
- **ssl_required**: Per configurar si acceptam connexions SSL. Els possible valors són: «ALL» (per totes les peticions), «EXTERNAL» (per peticions externes), o «NONE».
- **resource**: En els servidors de la DGEDDT hi ha disponibles tres *resources*:
  - goib-default: permet a l'usuari autenticar-se amb certificat digital (qualsevol dels admesos per *@firma*) o, si no disposa de certificat, mitjançant el seu usuari i contrasenya del GOIB.
  - goib-ws: es tracta del mecanisme d'autenticació que ha d'aplicar-se per protegir mòduls que continguin *serveis REST*. Permet autenticació *BASIC* sense redirigir la petició a la web centralitzada d'autenticació.
  - goib-cert: només habilita l'autenticació amb certificat, amb la qual cosa, només es podrà accedir als continguts protegits del mòdul, mitjançant l'ús d'un certificat vàlid.

A continuació es mostra un exemple de configuració per a un **mòdul web** de *backoffice*:

```
<subsystem xmlns="urn:jboss:domain:keycloak:1.1">
        <realm name="GOIB">
           <auth-server-url>URL_DEL_KEYCLOAK</auth-server-url>
            <ssl-required>ALL</ssl-required>
        </realm>
        <secure-deployment name="projectebase-back.war">
            <realm>GOIB</realm>
            <resource>goib-default</resource>
            <use-resource-role-mappings>false</use-resource-role-mappings>
            <public-client>true</public-client>
            <verify-token-audience>true</verify-token-audience>
        </secure-deployment>
```

```
</subsystem>
```

A continuació es mostra un exemple de configuració per a un **API REST**[34]:

```
<secure-deployment name="projectebase-api.war">
    <realm>GOIB</realm>
    <auth-server-url>URL_DEL_KEYCLOAK</auth-server-url>
    <resource>goib-ws</resource>
    <use-resource-role-mappings>true</use-resource-role-mappings>
    <bearer-only>true</bearer-only>
    <enable-basic-auth>true</enable-basic-auth>
    <ssl-required>ALL</ssl-required>
    <credential name="secret">CREDENTIAL_SECRET</credential>
</secure-deployment>
```

5.2. Control d'accés al mòdul web

En el fitxer ***web.xml*** de l'aplicació s'han de definir els rols d'accés permesos (***security-role***). Els noms d'aquest rols han de seguir les normes estandarditzades de la DGEDDT. Per al cas d'una aplicació amb prefix «*APL*» el nom especificat a l'etiqueta ***role-name*** ha de seguir el format «***APL_XXXXXX***», on «*XXXXXX*» ha de ser un nom simple i representatiu.

Exemples de noms de rols:

- APL_CONSULTA
- APL_USUARI
- APL_ADMIN

Exemple d'element *security-role*:

```
<security-role>
<description> ... descripció ...</description>
<role-name>PBE_ ADMIN</role-name>
```

3 Els servidors de la DGEDDT tenen activada l'opció *Standard Flow Enabled del client goib-ws*.
4 Els serveis REST requereixen activar el paràmetre «use-resource-role-mappings» i afegir un «credential name».

```
</security-role>
```

Al mateix fitxer, l'element ***security-constraint*** s'haurà d'utilitzar en cas d'haver de definir privilegis d'accés per a una col·lecció de recursos. Dins d'aquest element, haurà d'especificar-se els següents paràmetres:

- **auth-constraint/role-name**: Llistat de rols que tindran accés als recursos protegits. Aquest rols s'hauran de definir prèviament dins de l'element *security-role*. Per exemple: PBE_ADMIN.
- **login-config/auth-method**: Mètode d'autenticació. Tindrà sempre el valor «**KEYCLOAK**»; independent si es vol fer servir autenticació amb usuari / contrasenya (BASIC/FORM) o amb certificat (CLIENT-CERT). D'aquesta manera, no serà necessari modificar l'aplicació si es vol modificar el tipus d'accés ja que aquest comportament està definit dins del Keycloak i dins de la configuració del fitxer *standalone.xml* del JBoss.

A continuació es mostra un exemple de configuració del *web.xml* per restringir l'accés a tot l'entorn de backoffice del Project Base:

```
<security-constraint>
   <web-resource-collection>
      <web-resource-name>Tots els recursos</web-resource-name>
      <description>Tots els recursos</description>
      <url-pattern>/*</url-pattern>
   </web-resource-collection>
   <auth-constraint>
      <role-name>PBE_ADMIN</role-name>
   </auth-constraint>
</security-constraint>
<login-config>
   <auth-method>KEYCLOAK</auth-method>
   <realm-name>Govern dels Illes Balears</realm-name>
</login-config>
<security-role>
   <role-name>PBE_ADMIN</role-name>
</security-role>
```

5.3. Protecció d'EJBs

És necessari protegir els *EJBs* de manera que cap usuari o servei anònim pugui executar-los, tret que els *EJBs* hagin de ser públics.

Si es requereix que l'usuari tingui assignat un o diversos rols específics s'afegirà l'anotació **@RolesAllowed({"ROL1", "ROL2",...})** a tota la classe o al mètode concret en funció del nivell de seguretat requerit.

Si es requereix que l'usuari estigui autenticat, però no fa falta que tingui cap rol concret, s'afegirà l'anotació **@RolesAllowed("**")**. Això substitueix a l'antic rol "tothom".

En el cas que els EJBs hagin de ser públics, s'haurà d'especificar l'anotació **@PermitAll**.

Exemple:

```
@RolesAllowed({"PBE_ADMIN"})
public class FooService implements FooServiceInterface {
...
}
```

5.4. Dominis de seguretat

No s'haurà d'especificar cap ***security-domain***, ni explícitament ni en els descriptors *jboss-jar.xml, jboss-web.xml* o *jboss.xml.* Si aquests fitxers només s'usen per indicar el *security-domain,* aquests es poden eliminar directament.

Així mateix, tampoc s'haurà d'especificar cap referència al *security-domain* mitjançant anotacions en el codi font (*@SecurityDomain*).

6. Serveis WEB i API REST

Les aplicacions desenvolupades han de publicar serveis Web que permetin la integració d'altres aplicacions i/o la reutilització de dades.

Es restringeix l'ús de serveis *SOAP* en favor de serveis *REST*. Si es desitgen publicar serveis *SOAP* s'haurà de consultar amb la DGEDDT i justificar la seva utilització.

La publicació dels serveis REST es farà mitjançant una *API REST.* Es podran definir dos tipus d'*API REST*: interna i/o externa.

- **API interna**. S'hi publicaran els serveis REST d'ús intern a l'Administració de la Comunitat Autònoma de les Illes Balears. Aquests serveis REST permetran la integració amb altres sistemes propis de la CAIB i/o la consulta interna de les dades. Aquests serveis REST **no seran accessibles des d'Internet**. Les API REST internes podran estar securitzades.
- **API externa**. S'hi publicaran els serveis REST d'ús extern a l'Administració de la Comunitat Autònoma de les Illes Balears. Aquests serveis REST **seran accessibles des d'Internet** i permetran la integració amb sistemes externs a la CAIB i/o la consulta externa de les dades. Aquestes API REST podran estar securitzades.

Remarques per a la definició de l'API REST (interna i/o externa):

1) S'haurà de crear un submòdul ***Maven*** anomenat ***codiAplicació-api-interna*** i **codiAplicació-api-externa** sota el projecte pare on se situaran els recursos de l'API .
2) L'API s'haurà de publicar sempre en el **context */codiAplicacióapi/interna/*** o **/codiAplicacióapi/externa/***, segons el tipus d'API.
3) S'haurà de generar la documentació de l'API de forma automàtica mitjançant una eina tipus ***Swagger-UI***. El context de publicació haurà de ser /codiAplicacióapi/interna/swagger.json o /codiAplicacióapi/interna/index.html, en el cas d'una api interna, i /codiAplicacióapi/externa/swagger.json o /codiAplicacióapi/externa/index.html en el cas d'una API externa.

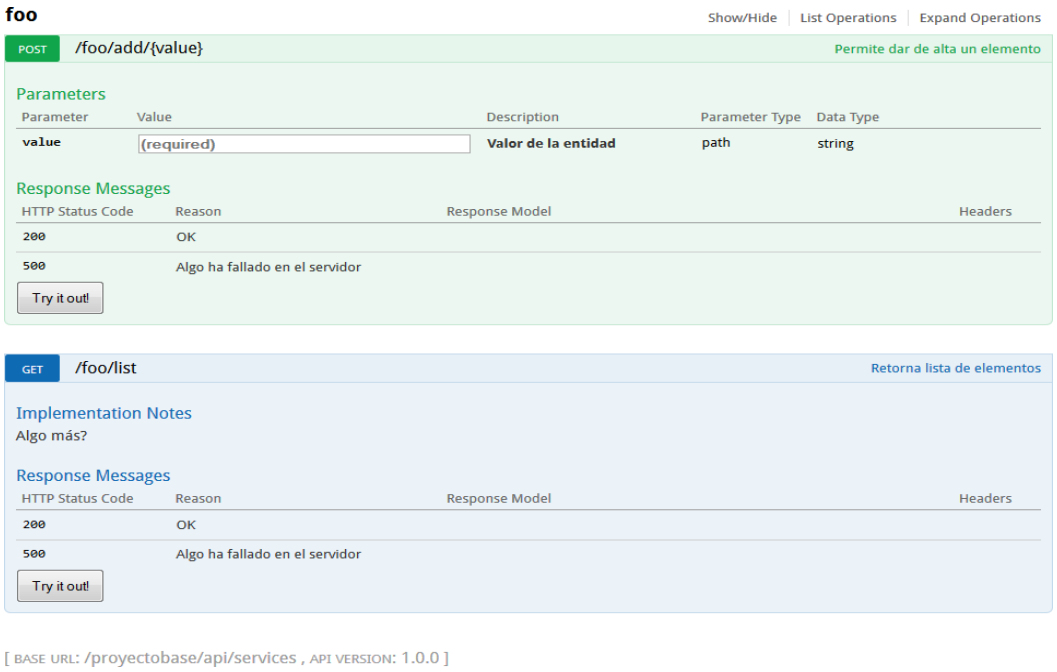


Figura 3: Exemple de documentació API publicada amb Swagger-UI