Development of an automatic Model-based generator for test automation projects

Treball de Fi de Grau - Memòria

Autor: Toni Miquel Llull Amengual
Director: Albert Tort Pugibet
Ponent: Jaime M. Delgado Mercé

Grau en Enginyeria Informàtica - Tecnologies de la Informació
Centre de desenvolupament del projecte: ***Sogeti España***

30 de juny de 2017

Facultat d'Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Resum

El desenvolupament de software requereix determinades activitats per assegurar la seva qualitat, denominades *Software Quality Assurance (QA)*. Una d'aquestes activitats és el testing (realització de proves), amb l'objectiu de comprovar el funcionament del software en el màxim possible de situacions i contextos. Aquestes proves es poden executar de forma manual, o bé es poden automatitzar per reduir el temps d'execució en els casos en què les proves s'han de repetir en diverses iteracions.

Empreses com **Sogeti España** tenen departaments especialitzats en QA y tenen com a focus l'automatització de proves, fet que permet als desenvolupadors concentrar-se en la millora del software, i als testers dissenyar i executar més proves de manera més freqüent.

Aquest projecte té com a objectiu millorar el procés d'automatització de proves a través d'una solució basada en models UML a partir dels quals s'autogenera l'estructura d'un projecte d'automatització.

Resumen

El desarrollo de software requiere determinadas actividades para asegurar su calidad, denominadas *Software Quality Assurance (QA)*. Una de estas actividades es el testing (realización de pruebas), con el objetivo de comprobar el funcionamiento del software en el máximo posible de situaciones i contextos. Estas pruebas se pueden ejecutar de forma manual, o bien se pueden automatizar para reducir el tiempo de ejecución en los casos en que las pruebas se deben repetir en diversas iteraciones.

Empresas como **Sogeti España** tienen departamentos especializados en QA y tienen como foco la automatización de pruebas, hecho que permite a los desarrolladores concentrarse en la mejora del software, i a los testers diseñar i ejecutar más pruebas de manera más frecuente.

Este proyecto tiene como objetivo mejorar el proceso de automatización de pruebas a través de una solución basada en modelos UML a partir de los cuales de autogenerará la estructura de un proyecto de automatización.

Abstract

Software development requires certain activities to ensure its quality, called *Software Quality Assurance (QA)*. One of these activities is testing, in order to verify the operation of the software in the maximum possible situations and contexts. These tests can be run manually or can be automated to reduce execution time in cases where tests must be repeated in several iterations.

Companies like **Sogeti España** have departments specialized in QA and have as focus the automation of tests, fact that allows the developers to concentrate in the improvement of the software, and to the testers design and to execute more tests of more frequent way.

This project aims to improve the process of automation of tests through a solution based on UML models from which will autogenerate the structure of an automation project.

Índex

Índex de figures

Índex de taules

1 Introducció

El projecte "*Development of an automatic Model-based generator for test automation projects*" correspon al Treball de Fi de Grau d'Enginyeria Informàtica de la **FIB** (Facultat d'Informàtica de Barcelona) dins de l'especialitat de Tecnologies de la Informació, i ha estat realitzat a **Sogeti España**, empresa especialitzada en oferir serveis d'automatització, especialment de testing automàtic.

La proposta d'aquest TFG és, per tant, estudiar una forma d'aportar valor a aquestes proves de testing automàtic aprofitant el treball previ que requereixen, i està enmarcat dins del departament d'investigació i desenvolupament de l'empresa.

1.1 Formulació del problema

Dins del desenvolupament d'aplicacions (o software en general) una part important, a banda del propi desenvolupament, és assegurar-se que aquest software funciona de la forma que esperem. A priori pot semblar que a mesura que l'anem fent ja tenim en compte tots els casos i situacions possibles, però això és, gairebé sempre, des del nostre punt de vista; no tots els usuaris el faran servir de la mateixa manera, i no sempre de la forma en que nosaltres creiem que s'ha de fer servir.

Aquests "imprevistos" poden provocar que el software falli en determinats moments, o que no funcioni de la forma esperada. Per això és important realitzar un testeig el més acurat possible.

Per dur a terme aquestes proves tenim varies opcions:

- Per una banda podem realitzar-les de forma manual, que en certs moments ens pot ser útil si les proves a realitzar no són molt extenses o per provar situacions puntuals. L'inconvenient de fer-ho manualment és que les proves poden ser molt lentes: per exemple si volem provar tot el procés de compra d'un bitllet d'avió a través d'una pàgina web, i una nova funcionalitat o millora que s'ha implementat o revisat es troba al final de tot el procés, haurem de passar per totes i cada una de les diferents pantalles i emplenar-ho tot a ma.

- Una altra opció més elaborada seria fer servir alguna eina que ens permeti automatitzar aquests processos com pugui ser, per exemple, Selenium [1]. Amb aquesta segona opció podem programar certes proves perquè es realitzin de forma automàtica, el que ens facilita molt la feina ja que no tenim la necessitat d'estar nosaltres realitzant les proves una a una. Seguint amb l'exemple anterior, en aquest cas el temps que invertim seria el de preparar i programar el test, però una vegada preparat, cada prova ens requerirà de pocs segons. Això ens permet iterar sobre el mateix test moltes més vegades que de forma manual en el mateix lapse de temps, el que ens facilita la realització de proves, canvis i modificacions d'una forma més ràpida.

Tot i així, això requereix d'un estudi previ del funcionament de l'aplicació (en cas que no siguem els propis desenvolupadors de la mateixa) i la posterior generació del codi que ens permeti realitzar aquestes proves, tant per a les proves manuals com automàtiques, i és la part que més temps ens requerirà.

1.2 Objectius del treball

Quan ens disposem a testejar un programa o aplicació el primer que hem de fer és aprendre com funciona l'aplicació que volem provar, pel que haurem de "jugar" una mica amb ella a fi de tenir una primera idea. Una vegada ja sabem què podem fer és hora de preparar les proves que volem realitzar, la qual cosa també ens requereix cert temps. La part bona del testing automàtic és que sabem que el temps invertit en generar el codi que ens permet executar les proves ens compensa el temps que tardem en realitzar-les.

Un exemple podria ser el procés de fer un registre a una web. Per fer-ho haurem de saber quins camps o elements de la web hi intervenen, quins són necessaris i quins no, i després generar el codi que ens permeti provar aquest procés. Si només fem aquesta prova, el temps no serà molt, però si hem de provar totes i cada una de les funcionalitats d'aquesta web, ens pot portar molt temps.

Aquest procediment, emperò, podria veure reduït el seu temps de dedicació si podem aprofitar part de la feina que ja està feta. Ens referim a l'anàlisi previ, i del qual en podem treure un bon profit.

La idea principal es basa en representar en un diagrama UML [2] (Unified Modeling Language, llenguatge que ens permet veure de forma més visual quins elements tenim i com interaccionen entre si) totes les pantalles, accions i atributs que poden requerir les diferents proves. El fet de fer servir un UML és que ens permet fer una representació prou fidel de les pantalles de la nostra aplicació (ja sigui mòbil o web), i dins de cada una tots els atributs o funcionalitat amb les que podem interactuar.

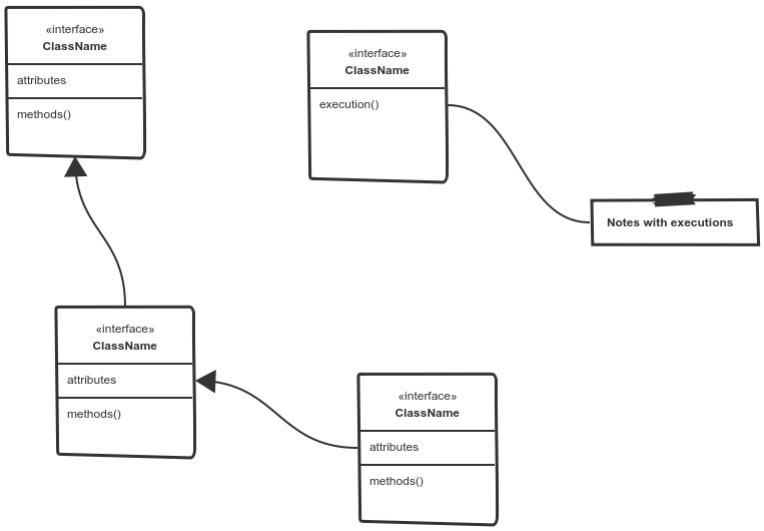


Figura 1: Exemple d'un diagrama UML

El següent pas és exportar aquest UML a format XML [3] (eXtensible Markup Language, metallenguatge basat en etiquetes que es pot fer servir per a molts usos diversos com web, bases de dades, fulls de càlcul, editors de text, etc.), que ens servirà per generar el nostre codi en llenguatge JAVA, i que podrem importar al nostre projecte d'automatització amb Selenium o, inclús, crear un nou projecte a partir d'aquest codi.

La forma en que generarem aquest codi serà mitjançant un script programat en Javascript [4].

1.3 Introducció al testing automàtic

La idea o objectiu de portar a terme una automatització de testing és la d'estalviar temps, a la llarga, a l'hora de realitzar aquestes proves. A la següent taula podem veure una petita comparativa entre el testing automàtic i testing manual:

Manual	Automàtic
Les proves manuals no són fiables al 100%, ja que cada execució pot no ser exacta a l'anterior	Les execucions repetitives són sempre exactament iguales
L'execució de proves manuals és molt ràpida la primera vegada, però la regressió pot no ser del tot útil si hi ha canvis constants en el codi	Les proves automàtiques aporten regressions molt útils en proves on el codi canvia freqüentment
Són útils quan les proves només s'hagin d'executar unes poques vegades	Són útils quan les proves s'han de repetir un gran nombre de vegades
Les proves manuals requereixen el mateix temps d'execució que la primera vegada, i cada prova requereix el seu temps	Les proves automàtiques, una vegada preparada la "suite de proves", es poden executar de forma més ràpida i es requereix menys d'interacció per part dels testers
No es pot provar el mateix test en diferents dispositius o SO alhora, el que requereix de més testers	Les proves automàtiques es poden executar en diferents plataformes de forma simultània
No es poden fer proves per comprovar informació interna de l'aplicació	Permet programar tests que interactuen amb la informació interna, el que permet crear proves més completes
Les proves manuals poden ser molt lentes	Els tests automàtics permeten realitzar les proves de forma molt més ràpida
Són més útils per les proves de UI o UX	No aporten molta informació a les proves de UI o UX
El cost inicial de les proves manuals és menor, però pot augmentar de forma molt ràpida	El cost inicial és més elevat degut a la preparació, però a la llarga pot estalviar gran quantitat de recursos

2 Contextualització

2.1 Estat de l'art

A l'hora de plantejar el projecte, i partint de la base en que aquest es desenvolupa en una empresa, s'ha estudiat la forma en que es treballa i es porten a terme aquestes feines d'automatització per intentar veure quines opcions hi ha per tal de fer-les més eficients.

Per poder abordar de la millor manera el projecte i veure si és possible la seva realització, s'han realitzat les següents tasques:

- Estudi previ dels diagrames UML

- Estudi previ del format XML

- Estudi previ d'eines similars

2.1.1 Estudi previ dels diagrames UML

Quan parlem de diagrames UML generalment ens referim a diagrames que fan referència al modelatge de software, però degut a la seva similitud amb el que s'està cercant en aquest projecte, s'ha pensat que seria un bon punt de partida.

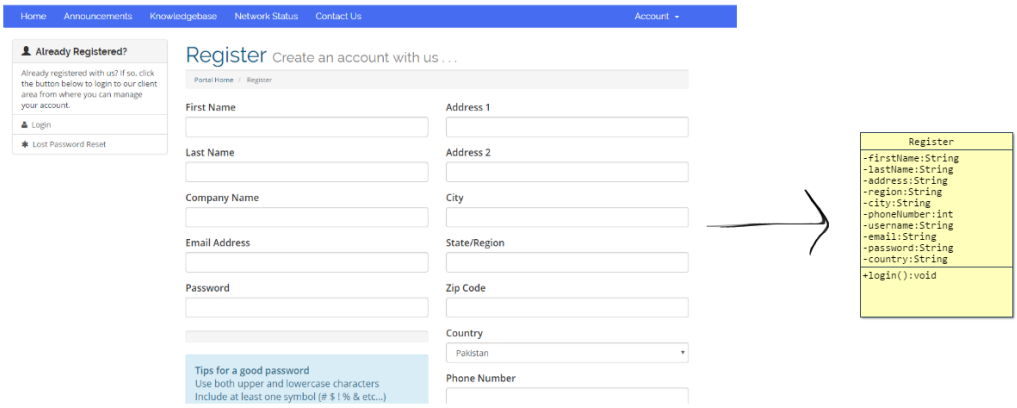


Figura 2: Representació d'una pantalla en UML

Amb aquest exemple podem veure que és relativament fàcil representar l'aplicació, inclús aprofitar les herències, generalitzacions o altres opcions que ens permet crear UML.

2.1.2 Estudi previ del format XML

Com que només de la imatge d'un UML no en podem fer res per generar codi, necessitem poder "llegir" aquest UML d'alguna forma, i la que s'ha pensat de fer servir és el llenguatge XML.

Aquest permet assignar un tag o etiqueta a cada tipus d'element, com el nom d'una classe, un atribut, el valor de cada un, etc. Així ja tindríem un bon punt de partida per poder analitzar l'UML d'alguna forma i poder començar a crear el nostre codi.

El resultat de representar una classe UML en XML és similar al de la següent figura:

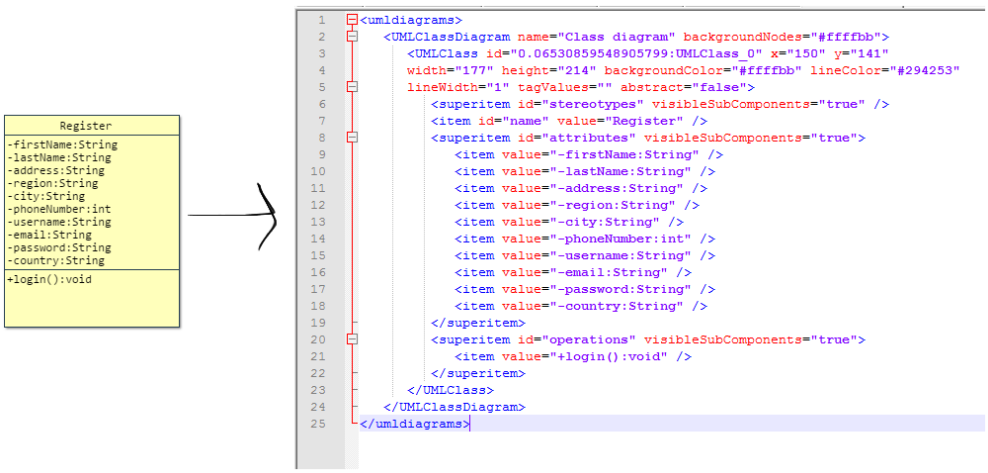


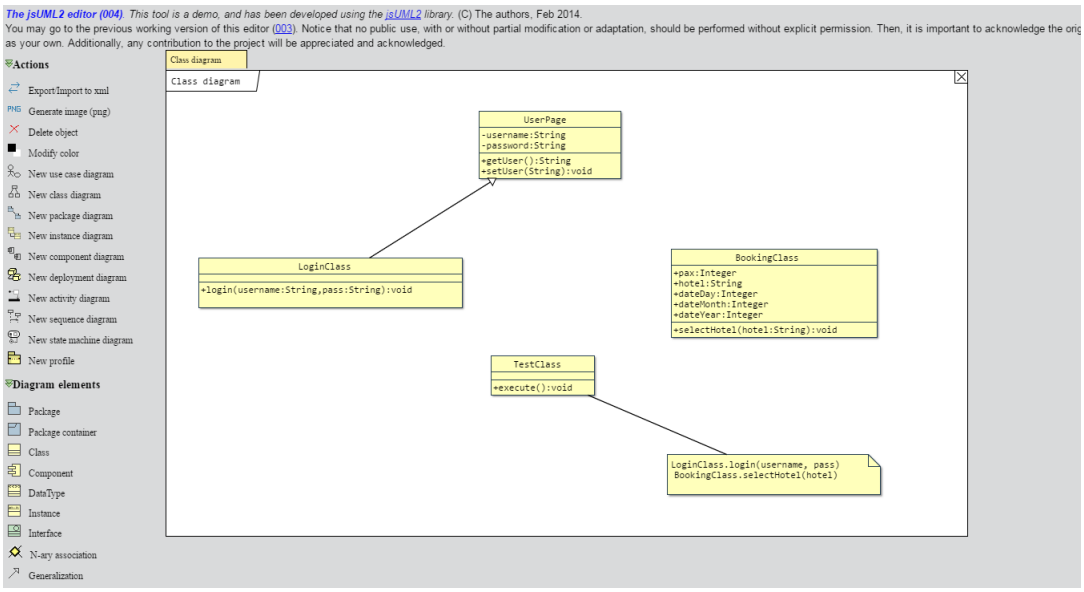
Figura 3: Representació d'una classe UML en XML

2.1.3 Estudi previ d'eines útils/similars

Una altra part fonamental abans de començar amb el projecte és realitzar una cerca per veure si hi ha alguna eina que ja hagi estat desenvolupada i que ens solucioni el problema.

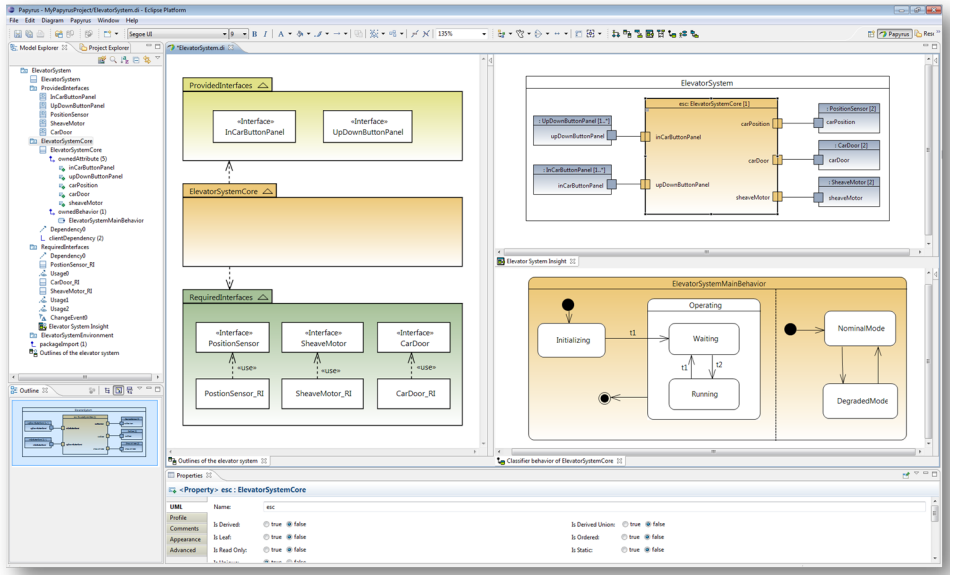
En aquest estudi s'han intentat trobar tant eines independents com plugins per altres softwares o inclús alguna solució web, i es detallen les més rellevants a continuació:

jsUML2 [5]: aquesta és una eina que ens permet generar diagrames UML d'una forma molt senzilla, la qual es pot fer servir via web, i que a més permet exportar-los a XML, pel que creiem que és una bona eina per utilitzar en aquest projecte.

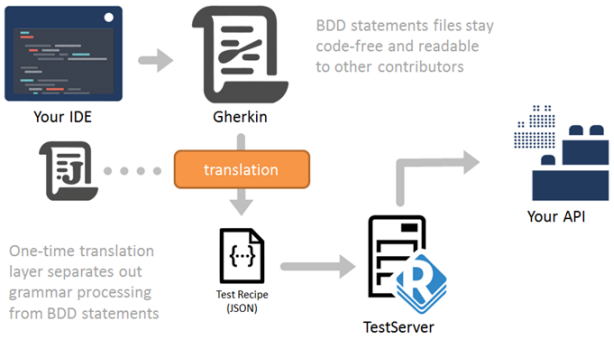


Papyrus [6]: aquest és un plugin per Eclipse [7] que permet integrar un editor d'UMLs dins del propi IDE, però el seu funcionament està pensat per projectes molt grans i requereix d'un aprenentatge extra per fer-lo servir, punt que ens ha fet descartar-lo quasi directament.

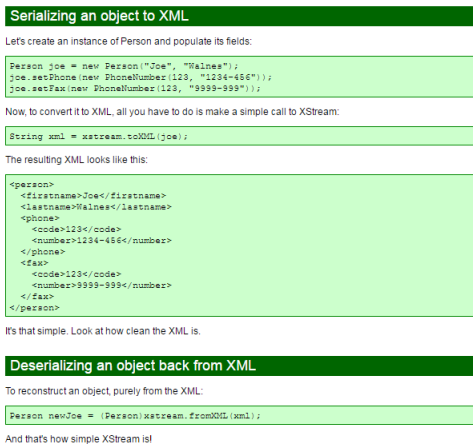
Per altra banda, és necessària la seva instal·lació a cada un dels ordinadors on es farà servir i en el nostre cas no li donarem tant d'ús.



Gherkin [8] + **Cucumber**[9]: aquesta opció es basa en crear histories d'usuari [10], el que tampoc ens soluciona el problema ja que en el nostre cas ens centrarem més en com estan distribuïdes les diferents pantalles de l'aplicació a testejar, i què hi podem realitzar a cada una d'elles. A més, el procés per fer-lo servir és també molt complexe pel que realment necessitem.



XStream [11]: aquesta és una llibreria en Java que permet passar de codi Java a XML i a l'inrevés. El problema és que el seu funcionament és molt "marcat", analitzant únicament les etiquetes una a una, però no tenim flexibilitat per filtrar aquestes etiquetes. A banda d'això, igual que passa amb Papyrus, és un codi que cada ordinador hauria de tenir carregat per poder-lo executar i estudiar amb deteniment el seu funcionament.



Serializing an object to XML

Let's create an instance of Person and populate its fields:

```
Person joe = new Person("Joe", "Walnes");
joe.setPhone(new PhoneNumber(123, "1234-456"));
joe.setFax(new PhoneNumber(123, "9999-999"));
```

Now, to convert it to XML, all you have to do is make a simple call to XStream:

```
String xml = xstream.toXML(joe);
```

The resulting XML looks like this:

```
<person>
  <firstname>Joe</firstname>
  <lastname>Walnes</lastname>
  <phone>
    <code>123</code>
    <number>1234-456</number>
  </phone>
  <fax>
    <code>123</code>
    <number>9999-999</number>
  </fax>
</person>
```

It's that simple. Look at how clean the XML is.

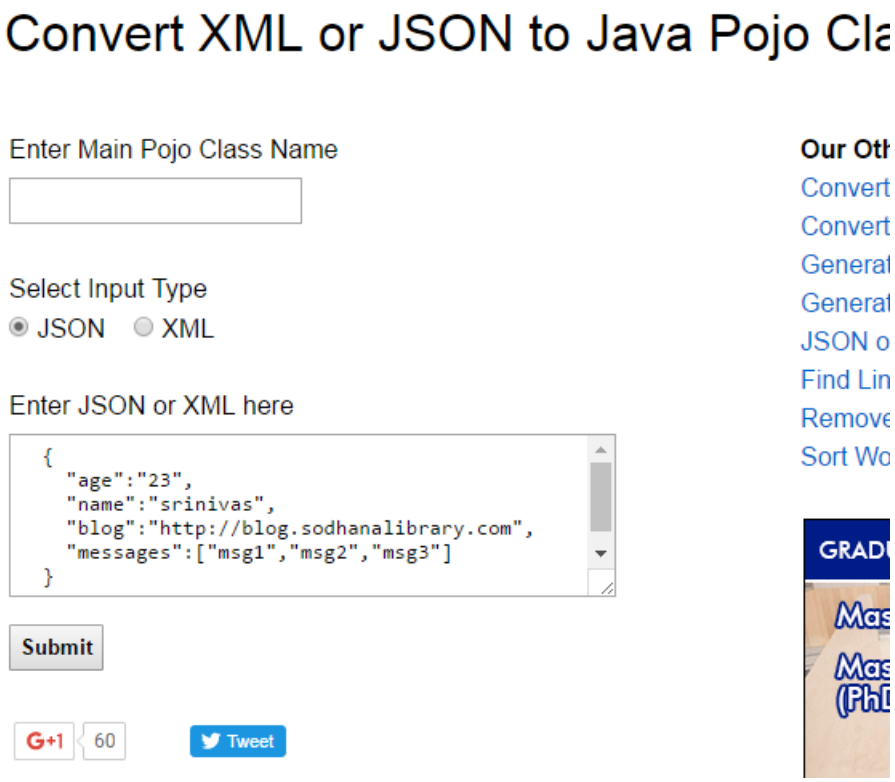
Deserializing an object back from XML

To reconstruct an object, purely from the XML:

```
Person newJoe = (Person)xstream.fromXML(xml);
```

And that's how simple XStream is!

xmlPOJO [12]: l'eina més semblant que hem trobat és una web que tracta fitxers "plans" per convertir-los a codi Java —POJO [13] (Plain Old Java Object)—, entre ells XML com volem fer noltros, però igual que passa amb XStream, només parseja cada tag amb el seu contingut, sense que puguem tractar-los de forma específica depenent del tipus de tag que sigui, pel que pel nostre cas tampoc ens és útil.



2.1.4 Conclusions

Després d'aquest estudi previ, on no hem trobat cap eina que s'adapti al que estem buscant, s'ha decidit desenvolupar el nostre propi sistema de generació de codi, adaptat específicament per projectes d'automatització.

2.2 Actors implicats

Els actors implicats en aquest projecte, a banda del propi desenvolupador, els detallarem a continuació:

- Per una banda, els principals beneficiaris en serien els **testers**. Tot i que no són molts els desenvolupadors que dediquen el temps sufcient a provar les seves aplicacions (ja sigui per falta de temps o perque aquest desenvolupament de software depèn de moltes persones), sí que hi ha empreses que es dediquen precisament a això, i estan especialitzades en el testing automatitzat. És per això que reduir el temps que s'-ha d'invertir a realitzar aquestes proves és fonamental per augmentar l'eficiència.

- Per altra banda, les **empreses** que es dediquen a fer testing automatitzat també es podrien aprofitar d'aquesta eina ja que la seva productivitat, lligada directament a la dels testers, es podria veure incrementada; major rapidesa per fer les proves implicaria un major volum de proves, el que permetria poder donar servei a més usuaris i/o empreses i les ajudaria a posicionar-se millor dins d'aquest sector.

- Finalment, els **usuaris** que sol·licitin aquestes proves també es veurien beneficiats ja que, lligat als dos actors anteriors, li permetria obtenir els resultats de les proves més ràpidament, pel que ajudaria a agilitzar el desenvolupament del seu software.

2.3 Abast

Per portar a terme l'estudi de la viabilitat d'aquesta idea farem servir una web fictícia on s'hi hauran de realitzar diferents operacions, i de la qual generarem el seu diagrama UML que representarà cada una de les diferents pantalles que podem trobar a la web, així com els elements clau que intervenen a cada una d'elles i els diferents processos.

L'objectiu final del projecte és poder generar tot el codi "estàtic" que ens faci falta per poder realitzar les proves de testeig necessàries. Això comporta realitzar un anàlisi exhaustiu que ens permeti tenir en compte totes i cada una de les etiquetes (tags) que tenim en el nostre XML, i sempre tenint en compte que aquest fitxer XML es generarà amb un mateix editor d'UMLs, ja que és impossible donar compatibilitat per a tots els editors de UML que generin fitxers XML degut a que cada un pot fer servir etiquetes diferents.

Nota: Encara així, si el projecte arriba a complir els requisits mínims, i després d'un temps d'ús es creu que val la pena, es podria pensar en ampliar aquesta compatibilitat a diferents editors d'UML.

Nota2: És impossible generar tot el codi necessari, ja que cada prova requereix que sigui preparada individualment, però sí que podem generar una estructura bàsica o esquelet del projecte.

2.3.1 Obstacles

En aquest projecte ens podem trobar varis obstacles que en poden limitar el seu abast, i que s'han de tenir en compte per tal de poder aconseguir completar-lo al 100%.

El primer, i potser més important, és la necessitat d'haver de disposar d'un diagrama UML; això implica que s'ha de tenir un coneixement previ d'aquests diagrames tant per generar-los com per entendre'ls.

A banda d'això, s'ha d'invertir un temps per generar-lo, el que fa que com més domini es tingui, més ràpid el podrem generar.

Un altre punt a tenir present és que la informació que podem considerar rellevant per la generació de codi de testeig és molt concreta; no tot el que apareix a una web o aplicació s'ha de representar al nostre diagrama, i això requereix d'un estudi previ de com funciona aquesta aplicació i determinar què és rellevant i què no.

Per altra banda, no totes les eines per generar UMLs permeten exportar-los a XML i, de les que sí, el format que li donen a les etiquetes, atributs, etc. segurament no és el mateix que una altra. Aquest problema és més difícil de solventar, i el que es farà en aquest projecte és fer servir una eina específica per generar UMLs, que serà jsUML2 [5], desenvolupada pel Dr. José Raúl Romero Salguero, professor de la Universidad de Córdoba.

Pel que fa a possibles problemes d'implementació (els anteriors són més de conceptes i de compatibilitats), és que el llenguatge que es farà servir és Javascript, juntament amb PHP [14], HTML (Hyper Text Markup Language) [15] i CSS (Cascading Style Sheets) [16]. Això requereix que s'han de tenir unes nocions bàsiques com a mínim per poder començar el projecte, i seguint amb l'apartat web, es requereix la configuració d'un servidor que ens permeti interactuar amb els diferents fitxers, lo que també pot suportar un obstacle a l'hora d'aconseguir la fita final de projecte.

Part del temps inicial s'ha invertit en agafar soltesa en aquests llenguatges i en la preparació del servidor web.

2.4 Metodologia i rigor

2.4.1 Mètodes de treball

En quant a la forma de desenvolupar el projecte, s'ha pensat en utilitzar una metodologia Agile [17], ja que és la metodologia que es fa servir dins de l'entorn de treball en el qual s'està portant a terme aquest projecte. Aquesta es basa en realitzar petites tasques a curt termini, les quals es posen en comú amb la resta de l'equip cada 4 setmanes aproximadament.

D'aquestes reunions en diem "iteracions" o "sprints", ja que es realitzen de forma iterativa i la durada de les mateixes és relativament curta, igual que el temps que passa entre una i altra. D'aquesta manera es poden posar fites més concretes i a més curt termini, les quals són relativament fàcils de solucionar o intentar encaminar que no pas si només es fes una reunió cada més temps i es posessin en comú parts molt més grans del projecte.

Pel que fa a les tasques o objectius, es determinen la gran majoria al principi del projecte, i s'intenta abastar el màxim de tasques importants o generals, perquè a cada sprint vagin sortint subtasques associades a cada una d'elles. D'aquestes tasques inicials en direm "Backlog", que seria la llista total de tasques.

A cada Sprint hi associem un "Sprint Backlog", que no és més que una sèrie de tasques del Backlog general que s'intentaran completar en aquesta iteració. En cas que es completin satisfactòriament, passaran a la llista de tasques completades, i en cas de que no sigui així, pot tornar al Backlog general degut a que hagin sortit imprevistos, i es posi directament al nou Sprint Backlog en cas que només facin falta alguns retocs finals.

En el nostre projecte s'han determinat 3 sprints principals que ens asseguren un producte final fiable:

La **primera iteració** ens permetrà obtenir una primera funcionalitat on es podrà generar un codi molt senzill per veure el funcionament del nostre codi, mostrant el resultat per pantalla, deixant de banda la generació de cap tipus de projecte. Això ens ajudarà a veure si el procés i plantejament decidit per abordar el problema és l'adequat.

La **segona iteració** ja incorpora la funcionalitat de generar el ZIP que ens permet descarregar els fitxers generats, però encara no s'ha implementada la funció de crear un projecte complet.

Finalment, serà en la **tercera iteració** on afegirem les funcionalitats de crear un projecte amb tots els fitxers i dependències necessàries, així com incloure les diverses opcions per personalitzar aquest projecte.

2.4.2 Eines de seguiment

Ja que aquest projecte el desenvolupa una única persona, les eines de seguiment no han estat eines específiques pel món empresarial ni per gestionar equips gran de treball.

Per mantenir la llista de tasques a realitzar s'ha fet servir l'aplicació Todoist, que ens permet crear projectes, subprojectes i tasques podent assignar deadlines, prioritats, notes, etc. S'havia pensat en fer servir Trello, una eina molt útil pel desenvolupament de projectes, però al final s'ha descartat ja que per aquest projecte no es requerien tantes opcions.

La comunicació entre en director del projecte i el ponent s'ha fet via correu electrònic tant per l'intercanvi de missatges i l'enviament de documentació o fitxers.

En quant al desenvolupament en si, s'ha fet servir Git [18], juntament amb GitHub [19], ja que ens permet tenir un registre de totes les versions que s'han fet del codi del projecte.

2.4.3 Mètode de validació

Per poder portar un control dels avenços del projecte s'han fet servir els següents mètodes:

- S'han establert una o dues reunions a cada iteració amb el tutor del projecte per poder rebre el *feedback* necessari.
- A cada nova implementació s'ha destinat una part del temps a realitzar les proves necessàries per assegurar-nos del seu correcte funcionament abans de continuar amb el desenvolupament de noves funcionalitats.
- S'han establert fites "parcials" o intermèdies per assegurar-nos un projecte funcional en cas que alguna de les funcionalitats no es pogués completar, les quals s'han anat complint a poc a poc a mesura que s'ha anat desenvolupant el projecte.

3 Anàlisi de Requisits

En aquest apartat s'especifiquen els requisits que ha de complir el projecte, que els separarem en requisits funcionals no funcionals.

A diferència d'un projecte de d'Enginyeria del Software, aquests requisits no són tan extensos i detallats ja que en el nostre cas el projecte està centrat en Tecnologies de la informació, i ve a cobrir una necesitat molt concreta de l'empresa on s'està desenvolupant.

3.1 Requisits funcionals

Els requisits funcionals fan referència als requisits que afecten a les funcionalitats que ofereix el projecte de cara a l'usuari.

- **Requisit #1**: Descarregar els fitxers .java

  L'usuari podrà descarregar els fitxers .java generats a partir de l'XML. Això implica que cada classe representada al diagrama UML ha de ser plasmada en un fitxer .java, amb els seus atributs i operacions.

- **Requisit #2**: Descarregar un projecte complet

  Hi ha d'haver l'opció de poder descarregar el projecte complet per poder ser importat en qualsevol IDE de desenvolupament JAVA, com per exemple Eclipse. L'usuari es descarregarà un ZIP on hi trobarà totes les llibreries necessàries per poder importar aquest projecte.

- **Requisit #3**: Seleccionar extres per ajustar més el projecte

  L'usuari podrà escollir què vol incloure al ZIP, depenent de si ja té o no les llibreries i fitxers necessaris per importar el projecte. També podrà escollir certs atributs específics de Selenium per tal de poder agilitzar el procés.

- **Requisit #4**: Oferir un exemple

  Si l'usuari no ha fet servir mai aquesta eina, se li facilitarà un exemple i una petita guia d'ús perquè es pugui familiaritzar amb el seu funcionament.

3.2 Requisits no funcionals

Els requisits no funcionals fan referència al comportament intern del projecte i què ha de commplir pel seu correcte funcionament, sense dependre de què faci l'usuari.

- **Requisit #1**: Indentació del codi generat

  Quan es genera el codi s'ha de tenir dura de que la indentació sigui l'adequada perquè l'usuari no hagi d'haver de "reindentar"el codi quan el faci servir. Es farà servir una indentació de 4 espais.

- **Requisit #2**: Interfície fàcil d'entendre

  La web que es farà servir ha de tenir la suficient informació però ha ser senzilla i fàcil d'entendre perquè l'usuari la pugui fer servir de la millor forma possible.

- **Requisit #3**: S'ha de poder fer servir en els diferents navegadors actuals

  La web ha de ser accessible i funcional des de qualsevol dels diferents navegadors actuals per facilitar el seu ús.

- **Requisit #4**: Un error en l'UML no ha de provocar un error generalitzat

  En cas que l'usuari hagi dissenyat malament l'UML, això no ha de provocar que tota la generació de codi sigui errònia, sinó que només afecti a la part corresponent.

4 Especificació

A partir de l'anàlisi de requisits es farà una especificació on es descriurà el comportament del sistema (servei) des del punt de vista dels actors que el faran servir.

Novament, com en el punt anterior, tant els actors com els casos d'ús són molt limitats, però es detallaran a continuació.

4.1 Actors

Realment només hi ha un actor que interactua amb l'aplicació, i és l'usuari com a tal. L'única interacció que té és la de pujar un fitxer XML perquè el sistema el processi amb les diferents opcions escollides, i posteriorment descarregar-lo.

4.2 Casos d'ús

Aquí també només tenim un únic cas d'ús:

Pujar un fitxer al servidor pel seu processament

- **Actor principal**: Usuari
- **Precondició**: El fitxer ha d'estar generat amb l'eina jsUML2
- **Escenari**:
  - **1**: L'usuari clica el botó destinat a seleccionar un fitxer per pujar al servidor
  - **2**: Se li mostren les diferents opcions que pot escollir/personalitzar
  - **3**: Configura el projecte al seu gust
  - **4**: Clica el botó de "Descarregar", i se li generarà un ZIP amb el projecte i se li demanarà on el vol descarregar.

5 Disseny

En aquesta secció es mostrarà el disseny del projecte tant de l'arquitectura lògica com física.

5.1 Arquitectura lògica

Per aquest projecte s'ha fet servir una arquitectura típica per les aplicacions del tipus client-servidor, com és el nostre cas. Aquesta és l'arquitectura en 3 capes, que ens permet separar els diferents components i tenir una independència que ens faciliti les coses.

Tot i que en certa manera podem dir que el nostre projecte no fa un ús intens de la capa de persistència ja que no guarda cap tipus d'informació més enllà del fitxer que es puja per analitzar, sí que és necessària per emmagatzemar la resta de fitxers per a la generació del projecte en JAVA.

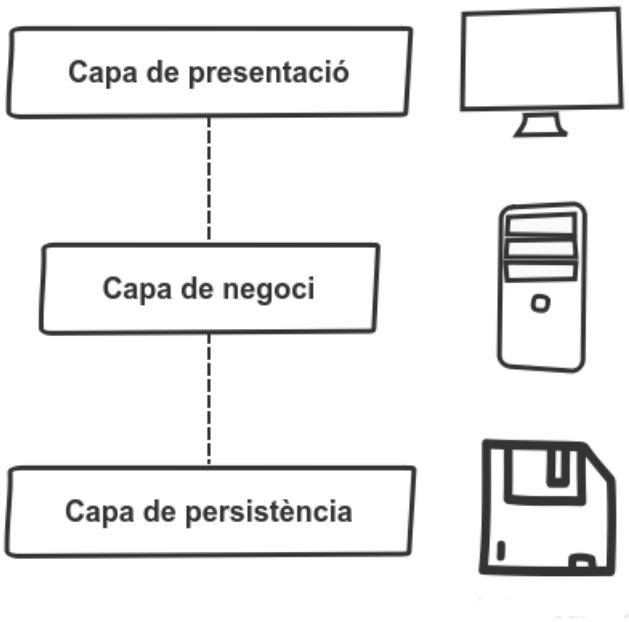


Figura 4: Representació de l'arquitectura de 3 capes

A continuació tenim una petita definició de cada una de les diferents capes:

Capa de presentació: És l'encarregada de la interacció usuari-aplicació, i es basa en mostrar la informació de la que disposa l'usuari. Es comunica amb la capa de negoci, enviant a aquesta les peticions que l'usuari ha seleccionat a la capa de presentació per a que siguin processades, i la capa de negoci li retorna un resultat. En el nostre cas, el resultat és la possibilitat de descarregar un fitxer ZIP.

Capa de negoci: És la capa situada entre la capa de presentació i dades (o persistència), i és l'encarregada d'implementar totes les funcions i operacions que ofereix l'aplicació. Tracta les dades que venen de la capa de presentació i fa la comunicació entre aquesta i la de persistència en cas de ser necessari, per retornar-li els resultats de la petició.

Capa de persistència: Aquesta capa s'encarrega que proveir a la capa de negoci la informació necessària per processar certa informació que li demana la capa de presentació. És on s'emmagatzemen totes les dades necessàries pel correcte funcionament de l'aplicació.

5.2 Arquitectura física

Pel que fa referència a l'arquitectura física, es basa novament en la típica arquitectura de client-servidor, on l'usuari es connecta des del seu ordinador al servidor remot a través d'Internet.

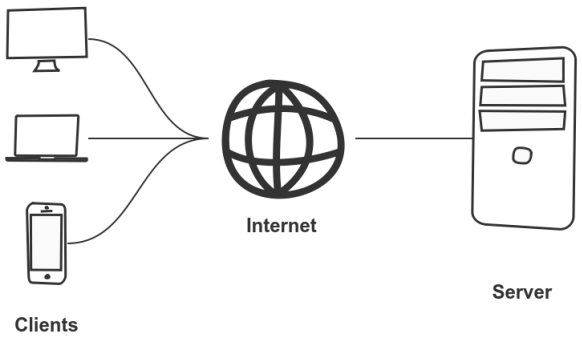


Figura 5: Representació de l'arquitectura física d'una connexió client-servidor

Seguint aquesta arquitectura, podem veure dos esquemes similars més específics del funcionament de l'aplicació.

Per una banda tenim la representació de la connexió de l'usuari que fa servir el nostre servei. La banda de client les tecnologies utilitzades són CSS, HTML o JavaScript, que són les que fan les peticions al servidor. A l'altra banda, el servidor fa servir PHP i unes llibreries ZIP per generar els fitxers que enviarà al client una vegada processades les dades.

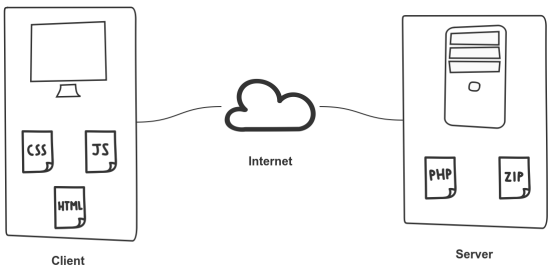


Figura 6: Representació de la connexió que fa el client al nostre servidor

Per altra banda, com que el desenvolupament es fa en remot, per poder pujar les noves actualitzacions i millores de la web, es fa una connexió similar. Es té un servidor local on es fan les diferents proves (tant de noves funcionalitats com de correcció d'errors) i, una vegada les proves han estat satisfactòries, es pugen a un servidor Git per posteriorment "desplegarles"en el servidor final.

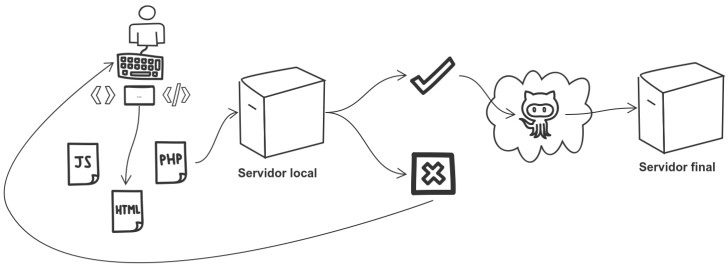


Figura 7: Representació del fluxe de desenvolupament entre els servidors local i remot

6 Desenvolupament i implementació

En el desenvolupament i la implementació han estat la part on més temps hem dedicat, ja que s'han hagut de plasmar totes les especificacions anteriors. Aquesta part es divideix de la següent manera:

- Preparació del servidor
- Disseny de la web (front-end)
- Implementació del processament de dades (back-end)
- Extres

6.1 Preparació del servidor

Obviant la instal·lació tant de Windows com de les eines que farem servir, el que sí que ens cal especificar una mica més és la part de preparar i muntar el servidor on estarà allotjat el nostre projecte. Es farà en una Raspberry Pi degut al seu reduït cost i que es pot treballar de la mateixa forma que un servidor Linux qualsevol, però si el projecte té l'acceptació per part dels testers i compleix els mínims requerits per l'empresa es passaria a un servidor intern.

Per instal·lar el SO a la Raspberry Pi ens baixarem la imatge des de la seva pròpia web i la "bolcarem" a una SD, des de Linux, amb la següent comanda:

```
dd bs=4M if=2017-04-10-raspbian-jessie.img of=/dev/sdX
```

Sent X el número que ens ha assignat el SO quan hem muntat el dispositiu. Això ens "instal·larà" Raspbian dins de la SD i ja la podrem fer servir.

El primer que farem una vegada instal·lat SO serà actualitzar-lo amb

```
sudo apt-get update && sudo apt-get upgrade
```

Això és necessari ja que, tot i haver acabat d'instal·lar el SO, potser no sigui la verssió més actualitzada.

A continuació procedirem a la instal·lació del servidor Apache amb

```
sudo apt-get install apache2
```

Per assegurar-nos que aquest servidor està sempre accessible, és necessari assignar una IP estàtica perquè el servidor DHCP no ens la canvii en algun moment. Per això, ho podem fer modificant l'arxiu *dhcpcd.conf*

```
sudo nano /etc/dhcpcd.conf
```

Afegint al final el següent codi:

```
interface eth0

static ip_address=192.168.1.13/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

on *ip_address* serà l'adreça IP que volem tenir, *routers* és l'adreça IP del nostre router, i *domain_name_server* serà, generalment, la mateixa.

A partir d'ara, bastarà accedir a la direcció indicada (sempre estant dins de la mateixa xarxa) per connectar-nos al nostre servidor.

Nota: Per la demostració del projecte s'ha requerit que aquest servidor sigui accessible des de fora (Internet), i degut a que és una part molt lligada a l'especialitat, explicarem com s'ha fet.

Igual que els servidors DHCP dels routers interns ens poden assignar una IP diferent de tan en tan, passa el mateix amb les IPs públiques que ens donen accés a Internet. Degut a això pot passar que en algun moment aquesta IP canvii i ja no sapiguem quina és la nova; per solventar aquest problema ens ajudarem de No-IP [20], un servei que ens associa un domini i manté actualitzada la nostra IP pública en tot moment. És, bàsicament, un servidor DNS.

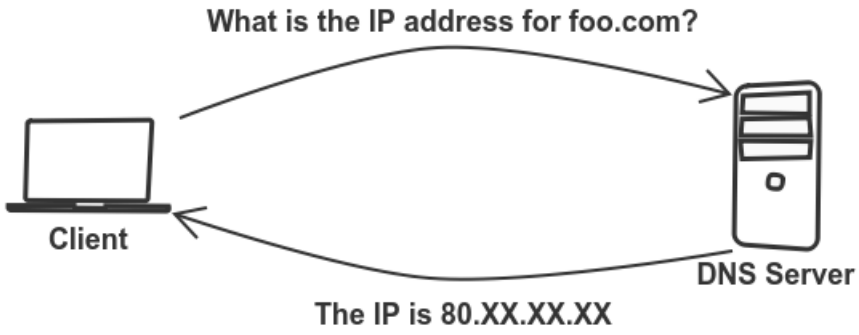


Figura 8: Representació del funcionament bàsic d'un servidor DNS

Crearem un compte a la seva web, un domini (gratuït) i instal·larem el client de No-IP a la nostra Raspberry. Primer, baixarem el client

```
wget http://www.no-ip.com/client/linux/noip-duc-linux.tar.gz
```

El descomprimim

```
tar -zxvf noip-duc-linux.tar.gz
```

Accedim a la carpeta que s'ha creat

```
cd noip-2.1.9-1
```

I l'instal·lem

```
sudo make
sudo make install
```

En aquest punt ens demanarà el nom d'usuari i la contrasenya del nostre compte de No-IP, i degut a que només tindrem un domini registrat, agafarà aquest per defecte. El temps de refresc el podem deixar per defecte, i a la següent pregunta, respondrem que NO (n).

Ara creem un nou fitxer que li direm noip2

```
sudo nano /etc/init.d/noip2
```

I hi copiarem la següent comanda

```
sudo /usr/local/bin/noip2
```

Guardem el fitxer amb **Ctrl+x**, acceptem els canvis amb **y**, premem **enter**, i li donem permissos d'execució

```
sudo chmod +x /etc/init.d/noip2
```

Actualitzem el fitxer d'inici perquè arrenqui cada vegada que engeguem la Raspi

```
sudo update-rc.d noip2 defaults
```

I posem el servei en marxa

```
sudo /usr/local/bin/noip2
```

Per acabar, ens falta obrir els ports del router perquè permeti l'accés des de Internet, i redirigeixi el trafic cap a la nostra Raspberry. Això es fa fent port-forwarding al router, i que tampoc detallarem ja que cada router és diferent, però és un procés relativament senzill de fer amb els routers actuals.

I ara sí, serà suficient amb accedir al domini que hem creat des de qualsevol lloc sense haver de saber quina IP tenim.

6.2 Disseny de la web (front-end)

Per començar el que s'ha fet ha estat crear una web que permeti pujar un fitxer al servidor. Aquesta web inicialment ha estat creada únicament amb HTML, però finalment hem inclòs CSS per donar-li un acabat més complert.

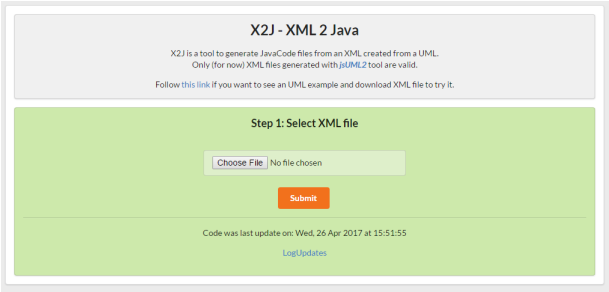


Figura 9: Pujar un fitxer al servidor X2J

NOTA: Cal tenir en compte que només hi pot haver un fitxer carregat en el servidor, pel que si pugem un de nou, sobreescriurà el que ja hi hagi.

Un cop hem seleccionat el nostre fitxer, tenim vàries opcions que podem seleccionar si ens interessa:

- Seleccionar l'adreça base de la nostra pàgina perquè s'inclogui de forma automàtica al nostre projecte
- Afegir varis elements del tipus "By", escollint el tipus d'atribut html que són d'entre id, name, xpath o css de moment
- Escollir si volem afegir al nostre projecte els fitxers .jar del framework Selenium o l'arxiu "chromedriver"
- *La selecció del format XML està deshabilitada, ja que la compatibilitat de moment es limita a jsUML2
- Esborrar el fitxer que hem pujat

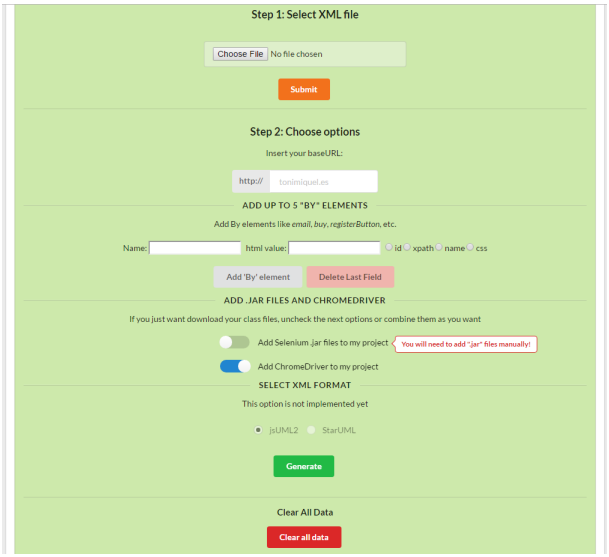


Figura 10: Llistat d'opcions per preparar el projecte abans de la seva descàrrega

Una vegada hem decidit les opcions, farem click al botó "Generate", el que començarà el procés de generació de codi i prepararà el fitxer .zip per poder-lo descarregar.

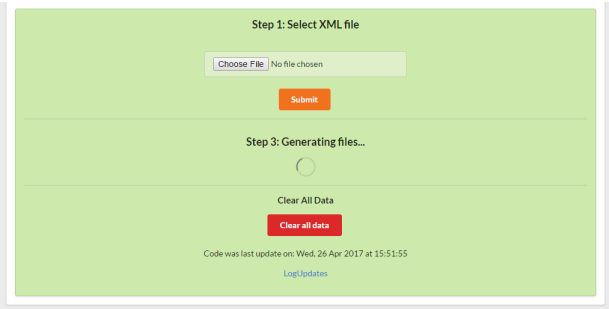


Figura 11: Generació del projecte

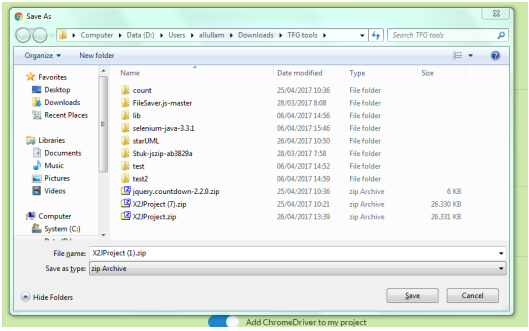


Figura 12: Finestra de selecció d'on volem guardar el nostre projecte

El codi de la web es pot consultar a l'apartat A dels anexes, i el CSS a l'apartat B.

6.3 Implementació del processament de dades (back-end)

Pel que fa al processament del fitxer, s'ha creat un codi en JavaScript que és l'encarregat de "traduir" l'XML a codi Java. Per fer-ho, tenim vàries funcions principals:

- Per una banda, es fa un primer anàlisi del fitxer per determinar quantes classes s'han de crear en total, per a la qual cosa ens ajudarem de la funció "responseXML" que ens ofereixen els navegadors, especificant que volem consultar un tag en concret (UMLClass en en nostre cas):

```
1  // Analize all classes into elems array. Main function
2  function fillElems(xml) {
3    var xmlDoc;
4    xmlDoc = xml.responseXML;
5    if (xmlDoc == null) {
6      alert("Please, upload a valid XML file");
7      return;
8    }
9    classes = xmlDoc.getElementsByTagName("UMLClass");
10   class_association =
     ↪  xmlDoc.getElementsByTagName("UMLAssociation");
11   if (class_association.length == 0) {
12     class_association =
       ↪  xmlDoc.getElementsByTagName("UMLLine");
13   }
14   class_notes = xmlDoc.getElementsByTagName("UMLNote");
15   class_generalization =
     ↪  xmlDoc.getElementsByTagName("UMLGeneralization");
16   //Analize classes
17   for (i = 0; i < classes.length; i = i + 1) {
18     getItems(i);
19   }
20   //Prepare generalization "pointer"
21   for (var n = 0; n < elems.length; ++n) {
22     elems[n][7] = "";
23   }
24   //Analize Generalizations
25   for (i = 0; i < class_generalization.length; ++i) {
26     getGeneralizations(i);
27   }
```

```
28    //Initializate notes to []. To prevent undefined pointers
29    for (var n = 0; n < elems.length; ++n) {
30      elems[n][6] = [];
31    }
32    //Analize associations
33    for (i = 0; i < class_association.length; ++i) {
34      getNotes(i);
35    }
36    //Write elements
37    for (i = 0; i < elems.length; i = i + 1) {
38      result = { val : "" };
39      writeItems(i);
40      elems[i][42] = result.val;
41    }
42    preZip();
43  }
```

Això ens retorna un array que guardarem a la nostra variable *classes*, que serà on tindrem tota la informació que fa referència a casa una de les classes del nostre projecte. Farem servir la mateixa petició pels diferents tags com Associacions, Generalitzacions, Notes, etc.

Una vegada tenim cada una de les classes, el que es farà és analitzar-la per extreure tota la informació rellevant. Això ho farem amb una funció pròpia amb la qual guardarem dins d'un array tota la informació separada i ordenada per un millor anàlisi posterior.

```
1  function getItems(i) {
2    attrs = [];
3    opers = [];
4    stereo = [];
5    var elem = [], id, childs, temp, abstract;
6    id = classes[i].attributes.getNamedItem("id").value;
7    elem[0] = id;
8    abstract =
     ↪  classes[i].attributes.getNamedItem("abstract").value;
9    elem[33] = abstract;
10   childs = classes[i].childNodes;
11   for (j = 0; j < childs.length; j = j + 1) {
12     if (childs[j].nodeType === 1) {
13       temp = childs[j].attributes.getNamedItem("id").value;
14       if (temp === "name") {
15         getName(elem, childs, j);
16       }
17       else if (temp === "attributes") {
18         getElements(elem, childs, j, attrs, 2);
19       }
20       else if (temp === "operations") {
21         getElements(elem, childs, j, opers, 3);
22       }
23       else if (temp === "stereotypes") {
24         getElements(elem, childs, j, stereo, 4);
25       }
26     }
27   }
28   elems[i] = elem;
29   elems[i][6] = [];
30 }
```

En aquesta funció es veuen crides a diferents funcions com *getName* o *getElements*, que són semblants a la pròpia getItems, però específica per a cada cas.

Una vegada tenim tota la informació processada, és el moment de començar a generar el codi. Per fer-ho també tenim vàries funcions que s'encarreguen de generar diferents part del mateix, com puguin ser els atributs, operacions, tipus de classe, etc.

```
1  // Write attrs items for every class
2  function writeAttrs(i) {
3    var check = 0;
4    for (j = 0; j < elems[i][2].length; j = j + 1) {
5      var str, n;
6      str = elems[i][2][j];
7      if (str.indexOf("-") !== -1) {
8        result.val += "    private ";
9        writeAttr(n, str, "-");
10       check = 1;
11     }
12     else if (str.indexOf("+") !== -1) {
13       result.val += "    public ";
14       writeAttr(n, str, "+");
15       check = 1;
16     }
17     else if (str.indexOf("#") !== -1) {
18       result.val += "    protected ";
19       writeAttr(n, str, "#");
20       check = 1;
21     }
22     else if (str.indexOf("~") !== -1) {
23       result.val += "    ";
24       writeAttr(n, str, "~");
25       check = 1;
26     }
27     else {
28       result.val += "    private ";
29       writeAttr(n, str, " ");
30     }
31   }
32 }
33
34 //Write attribute
35 function writeAttr(n, str, scope) {
```

```
36    var visib = [];
37    if (scope != " ") {
38      visib = str.split(scope);
39    }
40    else visib[1] = str;
41    n = visib[1].indexOf(":");
42    if (n !== -1) {
43      writeAttrType(visib, str);
44    }
45  }
```

Una vegada s'ha processat tot el codi, només ens queda crear el zip per descarregar-lo. Això es fa mitjançant una llibreria que ens permet tractar amb fitxers zip fent servir JavaScript, la qual ens ha servit per aquest procés. La funció en particular és la següent:

```
1   // Generate .zip with all files
2   function zip(path) {
3     loading();
4     JSZipUtils.getBinaryContent(path, function (err, data) {
5       if(err) {
6         throw err; // or handle the error
7       }
8       JSZip.loadAsync(data)
9         .then(function (zip) {
10        //ADD MY FILES
11        var rootFolder = zip.folder("X2JProject");
12        var files = zip.folder("X2JProject/src/defaultPackage");
13        for (i = 0; i < elems.length; i = i + 1) {
14          files.file(elems[i][1] + ".java", elems[i][42]);
15        }
16        readFile("uploads/file.xml", origFile);
17        rootFolder.file("MyProject.xml", origFile.val);
18        readFile("static_files/license.txt", license);
19        rootFolder.file("license.txt", license.val);
20        readFile("static_files/readme.txt", readme);
21        rootFolder.file("readme.txt", readme.val);
22        readFile("static_files/BaseClass1.txt", baseFile);
23        readTextFields();
24        readFile("static_files/BaseClass2.txt", baseFile);
```

```
25        var url = document.getElementById('baseURL').value;
26        if (url != "") baseFile.val += url;
27        else baseFile.val += "www.tonimiquel.es"
28        readFile("static_files/BaseClass3.txt", baseFile);
29        files.file("ExampleClass.java", baseFile.val);
30        //zip.file("new_file", "new_content");
31        // if you return the zip object, it will be available
          ↪  in the next "then"
32        return zip;
33      }).then(function (zip) {
34        // if you return a promise of a blob, promises will
          ↪  "merge": the current
35        // promise will wait for the other and the next "then"
          ↪  will get the blob
36        return zip.generateAsync({type: "blob"});
37      }).then(function (blob) {
38        saveAs(blob, "X2JProject.zip");
39        loaded();
40      });
41    });
42  }
```

Actualment estan suportades les següents característiques en quant a les classes, atributs i operacions:

- Classes:
  - · public
  - · abstract
  - · extends
  - · import
- Atributs:
  - · public
  - · private
  - · protected
  - · type (String, Integer, etc)
  - · value
- Operacions:
  - · public
  - · private
  - · parameters
  - · return type

El codi complert es pot consultar a l'apartat C del anexes.

6.4 Extres

La part web (html + css) i la part del processament (JavaScript) són les dues més importants, però també tenim altres arxius necessaris pel correcte funcionament. Un d'ells és el que ens permet pujar un fitxer al nostre servidor, i que està creat amb PHP ja que és un llenguatge que ens permet interactuar al costat del servidor, a diferència de JavaScript que ho fa a la part de client. El codi complert es troba a l'apartat D dels anexes.

```php
<?php

$target_dir = "../uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$errorCheck = 0;
$FileType = pathinfo($target_file,PATHINFO_EXTENSION);

/*..altres comprovacions..*/

// Check if $uploadOk is set to 0 by an error
if ($errorCheck != 0) {
  if ($errorCheck == 1) $message = "Please, select a file";
  else if ($errorCheck == 2) $message = "Sorry, your file is too large";
  else $message = "Sorry, only XML files are allowed";
  echo "<script type='text/javascript'>alert('$message');
  window.history.back();</script>";
  // if everything is ok, try to upload file
} else {
  if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"],
  "../uploads/" . "file.xml")) {
    echo "<script type='text/javascript'>window.location='../index.html';</script>";
  } else {
    $message = "Error uploading file";
    echo "<script type='text/javascript'>alert('$message');
    window.history.back();</script>";
  }
}

?>
```

L'altre fitxer PHP és el que ens permet esborrar el fitxer, i el codi és més senzill, ja que només consta d'unes quantes línies, i es pot consultar també a l'apartat E dels anexes.

```
1 <?php
2 if (file_exists('../uploads/file.xml')) {
3   unlink("../uploads/file.xml");
4 }
5 echo "<script type='text/javascript'>alert('All data
  ↪  cleared!');</script>";
6 echo "<script type='text/javascript'>
  ↪  window.location='../index.html';</script>";
7 ?>
```

7 Gestió del projecte

7.1 Planificació temporal

7.1.1 Duració del projecte

El projecte ha tingut una duració d'aproximadament 4 mesos i mig, començant el 20 de Febrer de 2017, coincidint amb l'inici de GEP (Gestió de projectes) i acabant la setmana del 26 de Juny, quan es fa la defensa.

7.1.2 Descripció de les tasques

En aquest punt es descriuen les diferents etapes del projecte, on a les 3 iteracions comentades a l'apartat de "Mètodes de treball", s'hi afegeixen dues més: GEP i Etapa Final.

GEP

En aquesta primera fase es fa un primer estudi del projecte i s'elaboren un seguit de documents que fan referència a molts dels aspectes que s'hauran de tenir en compte en la futura realització d'aquest projecte, com puguin ser una primera planificació temporal, abast, estimació de costos, etc.

La duració d'aquesta etapa és de 4 setmanes, i han hagut de realitzar entregues setmanals per tal de dur un seguiment de l'evolució del treball.

Primera iteració: Generació bàsica de codi

En aquesta primera iteració es prepara tot el codi necessari tant de la web on es pujaran els fitxers com el propi per fer la generació en Java. El que s'espera d'aquesta etapa és obtenir un codi Java coherent i ordenat a partir de l'XML inicial, que podrem veure per pantalla.

La dependència, tot i no ser directe, és la de GEP.

Segona iteració: Descarregar els arxius en format ZIP

La segona iteració afegeix la funcionalitat de poder descarregar els diferents fitxers en un únic ZIP, el que ja permet a l'usuari poder importar-los a un projecte existent de forma més senzilla.

En aquest cas sí que depenem de la primera iteració per poder dur a terme aquesta nova funcionalitat, ja que és imprescindible que la primera "traducció"es faci correctament.

Tercera iteració: Creació d'un projecte

La darrera iteració és on acabem de polir els detalls per poder incorporar a aquest ZIP els fitxers i dependències necessàries per poder importar-lo a un IDE com a projecte sencer. A més a més, s'han afegit diverses opcions on l'usuari pot escollir què vol incloure i què no dins d'aquest ZIP. Això ens ha suposat pre-carregar en el nostre servidor tots els arxius necessaris i afegir-los al ZIP que es genera depenent de les opcions escollides.

Aquí també tenim una dependència total de la iteració anterior, i ens permet completar el procés amb alguns extres que fan el projecte més interessant.

Tot i així, queda oberta la possibilitat d'anar ampliant funcionalitats depenent de l'ús que els usuaris en facin i el feedback que es vagi rebent.

Etapa final

Aquesta darrera etapa és la corresponent a la finalització de la memòria (ja que aquesta s'ha anat generant a mesura que s'ha anat avançant en el projecte, i part d'ella ha estat una revisió de GEP) i preparació de la presentació final.

7.1.3 Recursos

Els recursos que s'han tingut en compte per desenvolupar aquest projecte han estat tant recursos hardware, software com humans.

Hardware

Els recursos hardware utilitzats són els següents:

- Ordinador de sobretaula
- Pantalla
- Teclat i ratolí
- Raspberry Pi 3
- Targeta SD de 32GB

Software

Els recursos software utilitzats són els següents:

- Windows 7 Professional: És el sistema operatiu que farem servir per treballar tant en el desenvolupament com en la generació de documentació.
- WAMP (Windows Apache MyQSL PHP) [21]: És el servidor local que farem servir per simular el producte final i poder preparar i desenvolupar tot el que fa referència a la configuració de xarxa, com puguin ser peticions web, tractament d'arxius en el client/servidor, etc.
- Brackets [22]: Finalment hem optat per substituir l'editor de text Atom per Brackets, ja que ens ha aportat certs avantatges pel desenvolupament.
- Git: És el gestor de repositoris que hem escollit per tal de tenir un control de versions i còpies de seguretat. D'aquesta manera podem tenir vàries instàncies de les diferents versions i podem provar "cara a cara" les modificacions abans de decidir incloure-les en el producte final. S'ha optat per fer servir Git degut a la gran facilitat i expansió que té dins del desenvolupament de projectes de software.

- Github: És l'eina que hem escollit per tenir allotjades totes les nostres versions del projecte.
- Raspbian PIXEL [23]: És el sistema operatiu que farem servir a la Raspi, i sobre el qual estarà muntat el nostre producte final.
- Apache Server [24]: Hem instal·lat el servidor Apache ja que és un dels més utilitzats i és molt fàcil trobar qualsevol informació o ajut en cas de tenir algun problema.
- PHP: Hem hagut d'instal·lar també PHP per poder fer servir certes accions a la banda de servidor, ja que per defecte no ve instal·lat en el SO.
- LaTeX[25]: Per generar la documentació hem fet servir LaTeX.

Humans

En quant als recursos humans, aquest projecte ha estat desenvolupat per una sola persona però ha desenvolupat els següents rols:

- Cap de projecte
- Desenvolupador
- Tester

7.1.4 Estimació del temps

A la següent taula es detalla el temps invertit a cada etapa:

Tasca	Hores
Gestió de Projectes (GEP)	**50**
Abast i contextualització	15
Planificació temporal	7,5
Gestió econòmica	7,5
Presentació	10
Lliurable final	10
Primera iteració: Generació de codi	**125**
Primera generació bàsica del codi	50
Millora en la generació del codi	50
Testeig	25
Segona iteració: Descarregar els arxius en format ZIP	**100**
Preparar i dissenyar funcionalitat	50
Adaptació del projecte	40
Testeig	10
Tercera iteració: Creació d'un projecte	**100**
Creació bàsica d'un projecte	55
Adaptació específica	35
Testeig	10
Etapa final	**50**
Documentació i Memòria	35
Preparació de la presentació oral	15
Total	**425**

Taula 1: Resum del temps invertit en el projecte

7.1.5 Valoració d'alternatives i pla d'acció

En tot projecte és imprescindible tenir un pla d'acció per intentar solucionar (o evitar) possibles contratemps que puguin aparèixer.

En el nostre projecte en tenim un que potser és el que més ens pot influir, i és que tot hi que existeixen algunes eines per generar codi a partir d'un UML (com algun plugin per eclipse), no n'hi ha que ho facin a partir d'un XML amb una estructura com la que requerim. Per això, aquest pot ser un problema a l'hora de poder portar a terme la totalitat del projecte.

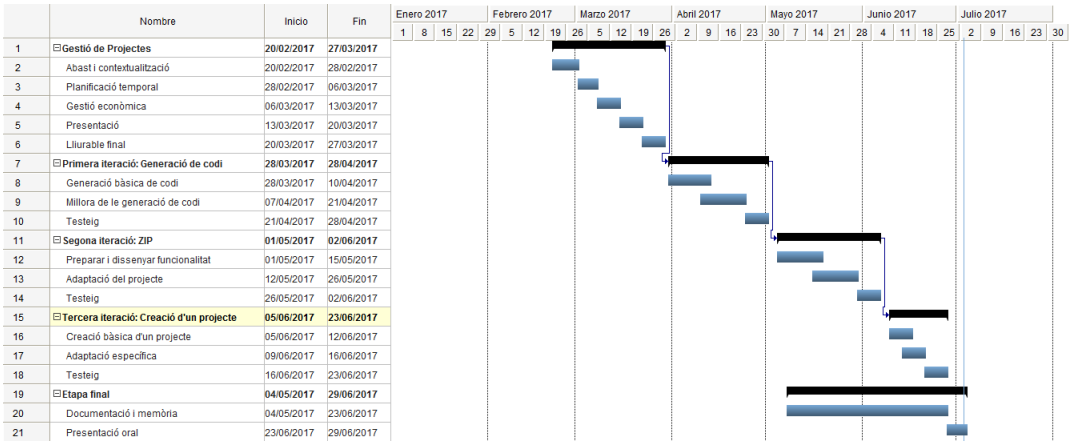
Tot i així, tenim pensat algunes alternatives com anar presentant els resultats de manera progressiva per tal que tot i que el resultat final no sigui l'esperat, sí que pugui ser funcional requerint una petita interacció amb l'usuari. Un exemple podria ser que el codi generat es mostra en format de text pla i no en un arxiu per importar directament dins d'un projecte, el que podria permetre a l'usuari copiar i enganxar aquest codi al seu projecte.

Un altre problema menys important és la dependència d'haver de disposar d'un projecte per realitzar les proves, però que intentarem solventar amb un projecte fictici en cas que arribat el moment límit d'haver de començar la part de desenvolupament no ens ha arribat.

La familiarització amb les diferents eines de treball també pot suposar un impediment per completar el projecte a temps, i per això s'intentarà que les dependències entre elles siguin mínimes. Així, en cas que una eina ens porti més temps del previst, o ens trobem amb un obstacle difícil de solventar, es pugui seguir treballant amb les demés eines o, donat un cas extrem, prescindir d'aquesta eina.

7.1.6 Diagrama de Gantt

A continuació s'adjunta el diagrama de Gantt previst. Si es vol veure a tamany complert, es pot fer des de http://tonimiquel.es/tfg/Gantt.png.



	Nombre	Inicio	Fin
1	**Gestió de Projectes**	**20/02/2017**	**27/03/2017**
2	Abast i contextualització	20/02/2017	28/02/2017
3	Planificació temporal	28/02/2017	06/03/2017
4	Gestió econòmica	06/03/2017	13/03/2017
5	Presentació	13/03/2017	20/03/2017
6	Lliurable final	20/03/2017	27/03/2017
7	**Primera iteració: Generació de codi**	**28/03/2017**	**28/04/2017**
8	Generació bàsica de codi	28/03/2017	10/04/2017
9	Millora de le generació de codi	07/04/2017	21/04/2017
10	Testeig	21/04/2017	28/04/2017
11	**Segona iteració: ZIP**	**01/05/2017**	**02/06/2017**
12	Preparar i dissenyar funcionalitat	01/05/2017	15/05/2017
13	Adaptació del projecte	12/05/2017	26/05/2017
14	Testeig	26/05/2017	02/06/2017
15	**Tercera iteració: Creació d'un projecte**	**05/06/2017**	**23/06/2017**
16	Creació bàsica d'un projecte	05/06/2017	12/06/2017
17	Adaptació específica	09/06/2017	16/06/2017
18	Testeig	16/06/2017	23/06/2017
19	**Etapa final**	**04/05/2017**	**29/06/2017**
20	Documentació i memòria	04/05/2017	23/06/2017
21	Presentació oral	23/06/2017	29/06/2017

Figura 13: Gantt

7.2 Gestió econòmica

7.2.1 Identificació i estimació dels costos

Aquest projecte es durà a terme per una sola persona, la qual haurà d'agafar els rols dels diversos recursos humans que componen un equip de projecte. A rel d'això els recursos tant materials com de software seran també més reduïts, ja que no faran falta més que un ordinador per treballar i algun altra element més.

En les següents seccions es detallaran cada un dels costos, diferenciats per costos en RRHH, materials, software i indirectes.

7.2.2 Recursos humans

Pel que fa als costos de Recursos Humans, com hem dit només hi haurà una persona encarregada de portar el projecte, però aquesta haurà de desenvolupar les diferents tasques. Per això, l'estimació que farem en quant als costos serà com si hi hagués cada un dels components de l'equip.

Així, el que necessitarem serà un cap de projecte, un desenvolupador i un tester, i el repartiment d'hores i costos estimats és com segueix a la següent taula.

Recursos Humans			
Rol	Salari	Hores	Total
Cap de projectes	50€/h	56 h	2.800 €
Desenvolupador	30€/h	225 h	6.750 €
Tester	25€/h	94 h	2.350 €
Total estimat		375 h	11.900 €

Taula 2: Recursos Humans

7.2.3 Recursos materials

En quant al recursos materials, tampoc seran molts, però sí que s'han de tenir en compte. Dividirem aquests recursos en 2 petits grups, ja que un serà el que contindrà els recursos pel desenvolupament i l'altre on hi haurà el producte final. Aprofitarem també per fer una estimació del temps de vida de cada un d'ells i poder fer una estimació de l'amortització que suposarà el temps que el farem servir.

Pel desenvolupament ens farem servir d'un ordinador de sobretaula "ThinkCenter", una pantalla Dell i un teclat i ratolí també Dell. En quant al grup de material on allotjarem el producte final es tracta d'una Rapsberry Pi model 3B, que farà servir una targeta SD de 32GB.

Grup de desenvolupament			
Producte	Cost	Vida útil	Amortització
Ordinador	300 €	4 anys	25 €
Pantalla	135 €	4 anys	11,25 €
Teclat + ratolí	25 €	4 anys	2,08 €
Grup del producte final			
Raspberry Pi 3B	40 €	2 anys	6,66 €
MicroSD 32GB	15 €	2 anys	2,5 €
Total estimat	500 €		47,49 €

Taula 3: Recursos materials

7.2.4 Recursos software

La pràctica totalitat del software és de programari lliure, però encara així l'inclourem dins de la taula per tenir un resum de tot el que farem servir pel projecte.

Software	
Producte	Cost
Windows 7	149 €
Brackets	-
WAMP	-
RaspbianOS	-
Apache	-
LaTeX	-
Total estimat	149 €

Taula 4: Recursos Software

7.2.5 Costos indirectes

Pel que fa a la resta de recursos, n'hi ha alguns dels que no podem prescindir, i degut a això també cal tenir-los en comtpe. Molts d'ells són recursos que no depenen del projecte en si, però que són necessaris. Per exemple tenim les despeses d'electricitat, connexió a Internet, etc.

Per calcular el consum total suposarem que l'ordinador i la pantalla estan engegats durant 8 hores al dia, mentre que la Raspberry ho està les 24h, ja que és el nostre servidor.

Costos d'electricitat Preu: 0,11621 €/KWh			
Recurs	Consum	hores/dia	total
Ordinador	350W	4	13,015 €
Pantalla	25W	4	0,925 €
Raspberry	1,8W	24	0,4 €
Despeses fixes			
Internet	35€/mes		140 €
Total estimat			154,34 €

Taula 5: Costos indirectes

7.2.6 Resum dels costos

Per tal de resumir els costos en una sola taula, podem veure el resum com segueix

Resum dels costos	
Concepte	Cost
Recursos Humans	11.900 €
Hardware	47,49 €
Software	149 €
Altres despeses	154,34 €
Total	12.250,83 €

Taula 6: Resum dels costos

7.2.7 Control de gestió

Per tal de controlar els costos de la forma més acurada possible, una vegada estimats intentarem fer un control periòdic per tal de revisar com està evolucionant el projecte.

D'aquesta manera si durant el desenvolupament del mateix hi ha algun imprevist com per exemple que l'ordinador s'espatlli, o hagem de substituir alguna peça ho podríem reflectir al moment a una taula de despeses extres que adjuntaríem al document per tal de tenir-lo actualitzat en tot moment.

En el nostre cas, degut a que el projecte és desenvolupar un software, la durada no és massa gran i que el hardware en el qual estarà allotjat és una Raspberry, no tindrem en compte ni la devaluació del producte ni les desviacions en quant al preu pel simple fet que aquest producte degut al seu reduït cost no varia excessivament.

7.3 Sostenibilitat i compromís social

La sostenibilitat és un punt que cada dia més és més important, sobretot dins del món de les noves tecnologies. Són moltes les grans empreses que intenten millorar el seu impacte social, i és per això que també en el nostre cas en facem un estudi.

Per fer-ho ens servirem d'una matriu de sostenibilitat (veure *Taula 7*), la qual costa de 3 parts, on cada una d'elles analitza diversos punts que fan referència a l'impacte ambiental, econòmic i social:

- Per una banda tenim la part del projecte posat en producció, que ens permet avaluar el projecte en funció del seu consum, el seu cost econòmic i el seu impacte personal.

- Per altra banda s'analitza la vida útil del producte, el que ens permet veure quina és la seva empremta ecològica, quin pla de viabilitat tenim i quin és l'impacte social que provoca.

- Per acabar, tenim l'apartat dels riscos i que reflexa quins poden ser els riscos tant ambientals com econòmics i socials.

Aquesta matriu té una valoració que surt de diverses preguntes relacionades en cada apartat, i ens serveix per veure en conjunt quin és l'estat del nostre projecte i si compleix o no els mínims per ser un projecte sostenible i viable.

	PPP	**Vida útil**	**Riscos**
Ambiental	Consum del disseny (0:10)	Empremta ecològica (0:20)	Riscos ambientals (-20:0)
Econòmic	Factura (0:10)	Pla de viabilitat (0:20)	Riscos econòmics (-20:0)
Social	Impacte personal (0:10)	Impacte social (0:20)	Riscos socials (-20:0)
Rang sostenibilitat	(-60:90)		

Taula 7: Matriu de sostenibilitat general

7.3.1 Impacte Econòmic

- Existeix una avaluació de costos, tant de recursos materials com humans?

  - Sí, s'ha fet una avaluació dels costos tant de Recursos Humans com materials.

- S'ha tingut en compte el cost dels ajustaments / actualitzacions / reparacions durant la vida útil del projecte?

  - S'han tingut en compte possibles problemes relacionats amb el hardware, però degut a la poca quantitat que en tenim, no és extremadament important.

- El cost del projecte ho faria viable si hagués de ser competitiu?

  - La part del cost més elevada és la de Recursos Humans, pel que sí que podem considerar que seria prou competitiu.

- Es podria realitzar un projecte similar en molt menys temps o amb molts menys recursos i, per tant, menor cost?

  - Degut a que és una proposta nova de desenvolupament, no hi ha eines que ens solucionin aquest problema concret.

- Està prevista o hi ha col·laboració amb algun altre projecte (acadèmic, empresa, associació, etc.?

  - L'única col·laboració que hi ha és amb la pròpia empresa on es desenvolupa el projecte, que és la que ha proposat aquest desenvolupament.

7.3.2 Impacte Social

- Hi ha una necessitat real del teu producte / servei?

  - Pel que fa a l'impacte social aquest projecte no està vinculat en cap aspecte ni a cap necessitat més enllà de la utilitat dins del mon de desenvolupament.

Pel que fa a nivell personal, aquest projecte m'ha aportat, fins aquest moment, uns coneixements sobre certs llenguatges de programació que no havia fet servir mai, al igual que a aprendre a desenvolupar des de 0 un projecte mitjanament important, tenint en compte tots els factors i punts del mateix, a diferència de certs projectes que s'han fet a la universitat on les feines es repartien entre vàries persones.

7.3.3 Impacte Ambiental

- Quins recursos es necessitaran en les diferents fases del projecte?

  - Els únics recursos que es faran servir seran els mateixos per a totes les fases del desenvolupament.

- Quin consum tindran aquests recursos durant el desenvolupament del projecte i posteriorment durant la seva posada en marxa i vida útil? Quin és l'impacte ambiental d'aquest consum (mesurat en tones de CO2, per exemple?)

  - El consum està contemplat a les taules anteriors, i una vegada acabat el ser desenvolupament, l'impacte durant la seva vida útil serà la que tingui el consum d'una Raspberry Pi, que és mínim, pel que el seu impacte ambiental serà també mínim, el que fa que sigui un dels punts forts en quant a sostenibilitat ambiental.

- Quin consum i impacte ambiental tindria realitzar la mateixa activitat sense l'existència del teu TFG (estalvi de paper i altres materials i/o energia?)

  - Com s'ha comentat, un dels punts forts és que l'impacte ambiental serà mínim degut a on es durà a terme la utilització del projecte durant la seva vida útil. Si traiem aquesta Raspberry i allotgem el treball a un altre ordinador, el consum seria més elevat, pel que l'impacte ambiental seria també major.

- Durant el desenvolupament del teu producte es generarà algun tipus de contaminació?

- Més enllà de la contaminació que pot provocar el consum d'energia elèctrica, no es generarà cap més tipus de contaminació.

7.3.4 Consum del disseny

Pel que fa al disseny del projecte, l'impacte ambiental que pot tenir és més bé reduït. La major repercussió ambiental es produeix durant el seu desenvolupament ja que és quan més recursos es fan servir, però una vegada completat, el consum energètic (que serà bàsicament l'únic recurs del que necessitarem) serà gairebé inapreciable degut a que una Raspberry Pi té un consum molt reduit.

7.3.5 Empremta ecològica

Actualment no hi ha cap solució al nostre problema concret, però sí que fan coses similars. Degut a que aquestes solucions estan basades en un software, és difícil estimar l'impacte que tenen ja que depen molt d'on es fan servir. L'únic que podem afirmar és que la nostra solució possiblement tingui un consum i impacte menor ja que es farà servir en un entorn molt específic, i la resta potser estan allotjades en servidors molt més grans on el consum és també molt més elevat.

7.3.6 Matriu de sostenibilitat

	PPP	**Vida útil**	**Riscos**
Ambiental	Consum del disseny (0:10) 8	Empremta ecològica (0:20) 18	Riscos ambientals (-20:0) -3
Econòmic	Factura (0:10) 8	Pla de viabilitat (0:20) 17	Riscos econòmics (-20:0) -5
Social	Impacte personal (0:10) 8	Impacte social (0:20) 17	Riscos socials (-20:0) -3
Rang sostenibilitat	(-60:90) 65		

Taula 8: Matriu de sostenibilitat del projecte

8 Conclusions

8.1 Consecució dels objectius

Una vegada finalitzat el projecte, podem concloure que s'ha arribat a assolir amb èxit els objectius establerts, ja que s'ha aconseguit una eina capaç de generar de forma automàtica una estructura bàsica per un projecte d'automatització basat en Selenium partint d'un esquema UML.

8.2 Possible treball futur

Tot i haver arribat a complir les expectatives del projecte, sempre es poden incloure millores o funcions extres per tal de fer-lo més complert. Algunes d'aquestes tasques futures podrien ser:

- **Ampliar compatibilitat XML**: Una millora molt interessant, però potser de les més complexes, seria ampliar la compatibilitat amb més aplicacions que ens generin XML a part de jsUML2.
- **Augmentar el nombre de tags reconeguts**: Un diagrama UML ens permet aplicar moltes definicions, atributs i personalitzacions, pel que afegir-les totes requereix d'un temps extra. Seria una bona ampliació afegir més tags.
- Actualment es pot crear projectes per provar amb Google Chrome, però es podria afegir també compatibilitat amb altres navegadors.

8.3 Conclusions personals

La realització d'aquest projecte ha estat una experiència molt satisfactòria ja que m'ha permès millorar no només els meus coneixements amb tecnologies que ja coneixia, sinó aprendre'n de noves i millorar la meva programació i configuració de servidors degut a la implementació que s'ha portat a terme.

Encara així, degut al temps limitat i el fet d'haver d'aprendre algunes de les tecnologies que s'han fet servir, no ha estat fàcil arribar a concloure el projecte amb un resultat tan positiu, el que fa que hagi estat encara més satisfactori.

A més a més, m'ha servit per veure de primera ma com es treballa en una empresa real i acostumar-me a treballar amb un equip de gent i dependre tots de tots.

En definitiva, estic molt satisfet amb el treball realitzat i per haver pogut aportar una mica de valor al treball que es realitza en una empresa com Sogeti España.

8.4 Competències tècniques

A continuació es llisten les competències tècniques associades al projecte.

CTI1.1: Demostrar comprensió de l'entorn d'una organització i de les seves necessitats en l'àmbit de les tecnologies de la informació i les comunicacions. [Bastant]

Realitzar aquest treball en una empresa dedicada al sector de les tecnologies de la informació, entre d'altres entorns, m'ha servit per entendre con funcionen realment aquest tipus d'empreses, i que poc tenen a veure amb el que es mostra a la facultat (per motius obvis. Entendre les necessitats que tenen aquestes empreses i com abordar els problemes i feines ha estat una experiència molt gratificant i enriquidora.

CTI3.1: Concebre sistemes, aplicacions i serveis basats en tecnologies de xarxa, tenint en compte Internet, web, comerç electrònic, multimèdia, serveis interactius i computació ubiqua. [En profunditat]

Les necessitats de l'empresa pel que fa referència al projecte eren les de poder fer servir un sistema econòmic però al mateix temps funcional. Per això es va decidir dissenyar un sistema que no depengués del propi ordinador de treball, sinó que fos accessible fàcilment des de qualsevol ordinador. Per això es va optar per enfocar aquest projecte com un servei web, on es processa tota la informació en un servidor intern de l'empresa, per evitar així haver-lo d'instal·lar a cada un dels ordinadors que el faran servir. Això ens ha portat a implementar un servidor i treballar tant en la part del client com del propi servidor per poder tractar i processar les dades correctament.

CTI3.4: Dissenyar software de comunicacions. [Una mica]

Inicialment es tenia una idea diferent del projecte i al final aquesta competència, tot i estar marcada com que només es treballaria una mica, no s'hi ha treballat ja que l'interacció i comunicació entre usuaris en el resultat final del projecte no s'ha aplicat.

CTI4: Emprar metodologies centrades en l'usuari i l'organització per al desenvolupament, l'avaluació i la gestió d'aplicacions i sistemes basats en tecnologies de la informació que assegurin l'accessibilitat, l'ergonomia i la usabilitat dels sistemes. [Bastant]

Per treballar en aquest projecte ens hem servit de tot un recull d'eines que ens han ajudat en el seu desenvolupament, així com per la seva gestió. Entre elles hi trobem eines de comunicació interna pel seguiment com Slack o Trello, o Git pel control de versions del projecte.

Referències

[1] *Selenium*. http://www.seleniumhq.org (Consultat el 08/05/20173).

[2] *UML*. https://ca.wikipedia.org/wiki/Llenguatge_unificat_de_modelat (Consultat el 08/05/20173).

[3] *XML*. https://ca.wikipedia.org/wiki/Extensible_Markup_Language (Consultat el 08/05/20173).

[4] *JavaScript*. https://ca.wikipedia.org/wiki/JavaScript (Consultat el 08/05/20173).

[5] *jsUML2*. http://www.jrromero.net/tools/jsUML2 (Consultat el 08/05/2017).

[6] *Papyrus*. https://eclipse.org/papyrus/ (Consultat el 08/05/20173).

[7] *Eclipse IDE for Java*. https://eclipse.org/ (Consultat el 10/05/2017).

[8] *Gherkin*. https://cucumber.io/docs/reference (Consultat el 10/05/2017).

[9] *Cucumber*. https://cucumber.io/ (Consultat el 08/05/20173).

[10] *Històries d'usuari*. https://es.wikipedia.org/wiki/Historias_de_usuario (Consultat el 08/05/20173).

[11] *XStream*. http://x-stream.github.io/index.html (Consultat el 08/05/20173).

[12] *XML to POJO*. http://pojo.sodhanalibrary.com/ (Consultat el 08/05/20173).

[13] *POJO*. https://en.wikipedia.org/wiki/Plain_old_Java_object (Consultat el 08/05/20173).

[14] *PHP*. https://ca.wikipedia.org/wiki/PHP (Consultat el 08/05/2017).

[15] *HTML*. https://ca.wikipedia.org/wiki/Hyper_Text_Markup_Language (Consultat el 08/05/2017).

[16] *CSS*. https://ca.wikipedia.org/wiki/Cascading_Style_Sheets (Consultat el 08/05/2017).

[17] *Metodologia Agile*. https://en.wikipedia.org/wiki/Agile_software_development (Consultat el 08/05/2017).

[18] *Git*. https://en.wikipedia.org/wiki/Git (Consultat el 08/05/2017).

[19] *Github*. https://ca.wikipedia.org/wiki/GitHub (Consultat el 08/05/2017).

[20] *No-IP*. https://www.noip.com/ (Consultat el 17/05/2017).

[21] *WAMP*. http://www.wampserver.com/ (Consultat el 08/05/2017).

[22] *Brackets*. http://brackets.io/ (Consultat el 08/05/2017).

[23] *Raspbian*. https://www.raspberrypi.org/downloads/raspbian/ (Consultat el 08/05/2017).

[24] *Apache*. https://www.apache.org/ (Consultat el 08/05/2017).

[25] *LaTeX*. https://www.latex-project.org/ (Consultat el 17/05/2017).

Anexes

A index.html

Codi corresponent a la pàgina web, en HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>X2J</title>
    <link rel="stylesheet" type="text/css"
    ↪ href="https://cdnjs.cloudflare.com/ajax/libs/semantic
    ↪ -ui/2.2.9/semantic.min.css">
    <link rel="stylesheet" type="text/css"
    ↪ href="css/styles.css">
    <script type="text/javascript"
    ↪ src="js/jszip-utils.js"></script>
    <script src="js/scripts.js"
    ↪ type="text/javascript"></script>
    <script src="js/jszip.js" type="text/javascript"></script>
    <script src="js/FileSaver.js"
    ↪ type="text/javascript"></script>
    <script src="https://code.jquery.com/jquery-3.2.1.js"
    ↪ integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3o
    ↪ i35EouyE="
            crossorigin="anonymous"></script>
  </head>
  <body onload="initCheck();">
    <div class="ui segment" align="center" id="main">
      <div class="ui clearing segment" id="header">
        <div class="content">
          <h2 class="ui center aligned header">X2J - XML 2
          ↪ Java</h2>
          <div class="description">
            <p>X2J is a tool to generate JavaCode files from
            ↪ an XML created from a UML.<br>
              Only (for now) XML files generated with
```

```
23              <a href="http://www.uco.es/users/in1rosaj/tools
                ↪ /jsUML2/editor/index.html" target="_blank">
                ↪ <i><b>jsUML2</b></i></a> tool are valid.
24            </p>
25            <p>
26              Follow <a href="example.html"
                ↪ target="_blank">this link</a> if you want
                ↪ to see an UML example and download XML file
                ↪ to try it.
27            </p>
28          </div>
29        </div>
30      </div>
31      <div class="ui clearing segment" id="process">
32        <h3>Step 1: Select XML file</h3>
33        <div class="ui form" id="step1">
34          <div class="field">
35            <form action="php/upload.php" method="post"
                ↪ enctype="multipart/form-data" class="ui form">
36              <div id="upload">
37                <input type="file" name="fileToUpload"
                  ↪ id="fileToUpload" accept=".xml">
38              </div>
39              <button class="ui orange button" type="submit"
                ↪ id="uploadButton">Submit</button>
40            </form>
41          </div>
42        </div>
43        <div id="parse">
44          <div class="ui divider"></div>
45          <h3>Step 2: Choose options</h3>
46          Insert your baseURL: <br><br>
47          <div class="ui labeled input">
48            <div class="ui label">
49              http://
50            </div>
51            <input type="text" placeholder="tonimiquel.es"
                ↪ id="baseURL">
52          </div><br>
53          <div class="ui horizontal divider">
```

```
54              Add up to 5 "By" elements
55            </div>
56            Add "By" elements like <i>email</i>, <i>buy</i>,
               ↪  <i>registerButton</i>, etc. to ExampleClass and
               ↪  use them in your project.
57            <div id="fieldExtras">
58              <form method="POST">
59                <div id="dynamicInput"></div><br>
60                <input class="ui button" id="addField"
                   ↪  type="button" value="Add 'By' element"
                   ↪  onClick="addInput('dynamicInput');">
61                <input class="ui button" id="delField"
                   ↪  type="button" value="Delete Last Field"
                   ↪  onClick="deleteElem();"><br>
62              </form>
63            </div>
64            <div class="ui horizontal divider">
65              Add .jar Files and Chromedriver
66            </div>
67            If you just want download your class files, uncheck
               ↪  the next options or combine them as you
               ↪  want<br><br>
68            <div>
69              <div id="toggles">
70                <div id="jarToggle">
71                  <div class="ui toggle checkbox">
72                    <input type="checkbox" name="jar"
                       ↪  id="jarFiles" onchange="doalert(this)"
                       ↪  checked>
73                    <label>Add Selenium .jar files to my
                       ↪  project</label>
74                  </div>
75                  <div class="ui left pointing red basic label"
                     ↪  id="jarAlert">
76                    You will need to add ".jar" files manually!
77                  </div><br><br>
78                </div>
79                <div id="chromeToggle">
80                  <div class="ui toggle checkbox">
```

```
81                  <input type="checkbox" name="chrome"
                     ↪  id="chromeDriver"
                     ↪  onchange="doalert(this)" checked>
82                  <label>Add ChromeDriver to my
                     ↪  project</label>
83                </div>
84                <div class="ui left pointing red basic label"
                   ↪  id="chromeAlert">
85                  You will need to add "chromedriver"
                     ↪  manually!
86                </div>
87              </div>
88              <div class="ui negative message"
                 ↪  id="downloadAlert">
89                <div class="header">
90                  Only class files will be downloaded!
91                </div>
92              </div>
93            </div>
94          </div>
95          <div class="ui horizontal divider">
96            Select XML format
97          </div>
98          This option is not implemented yet<br><br>
99          <div>
100           <div class="ui form" id="xmlFormat">
101             <div class="inline fields">
102               <label></label>
103               <div class="field">
104                 <div class="ui radio disabled checkbox">
105                   <input type="radio" name="frequency"
                       ↪  disabled="disabled" checked="checked">
106                   <label>jsUML2</label>
107                 </div>
108               </div>
109               <div class="field">
110                 <div class="ui radio disabled checkbox">
111                   <input type="radio" name="frequency"
                       ↪  disabled="disabled">
112                   <label>StarUML</label>
```

```
113                   </div>
114                 </div>
115               </div>
116             </div>
117           </div>
118           <div id="errors">
119             <p id="errorMsg"></p>
120           </div><br>
121           <div class="ui animated green button" tabindex="0"
              ↪ onclick='parse()'>
122             <div class="visible content">Generate</div>
123             <div class="hidden content">
124               <i class="right arrow icon"></i>
125             </div>
126           </div><br><br>
127         </div>
128         <div id="download">
129           <div class="ui divider"></div>
130           <h3>Step 3: Generating files...</h3>
131           <div>
132             <p id="files"></p>
133             <div class="ui active centered inline loader"
                ↪ id="loader"></div>
134           </div>
135           <div>
136             <h4></h4>
137           </div>
138         </div>
139         <div class="ui divider"></div>
140         <div id="clear">
141           <h4>Clear All Data</h4>
142           <form action="php/clearData.php" >
143             <button class="ui red button" type="submit">Clear
                ↪ all data</button>
144           </form>
145           <br>
146         </div>
147         <div id="log">
148           <p id="last" align="center"></p>
```

```
149             <p align="center"><a href="logs.html"
                ↪  target="_blank">LogUpdates</a></p>
150         </div>
151       </div>
152       </div>
153     </body>
154 </html>
```

B styles.css

Codi CSS per millorar el disseny de la pàgina web

```
#main{
  margin-top: 1%;
  margin-left: 20%;
  margin-right: 20%;
  margin-bottom: 1%;
}

#upload{
  padding: 2%;
  margin-left: 20%;
  margin-right: 20%;
  max-width: 40%;
}

#parse{
  display: none;
  background: #cde9ab;
}

#clear, #download, #loader, #errors, #downloadAlert{
  display: none;
}

#addField{
  display: inline;
}

#toggles, #xmlFormat{
  display: inline-block;
  text-align: left;
}

#delField{
  background: #f0b4ad;
  display: inline;
  display: none;
```

```
}

#fileToUpload{
  background: #deefca;
}

#header{
  background: #f0f0f0;
}

#process, #example{
  background: #cde9ab;
}

#logBody {
  text-align: center;
}

#list{
  text-align: left;
}

#jarAlert, #chromeAlert{
  display: inline-block;
  float: right;
  display: none;
  position: absolute;
}

body {
  background: #ebebeb;
}
```

C scripts.js

Codi principal en JavaScript que ens permet generar el projecte en Java a partir de l'XML

```
/*
****************************************
**********GLOBAL VARIABLES**************
****************************************
*/

// Result of parsed class
var result = { val : "" };

// Array with download link for each class generated
var urls = [];

/*Array with all classes from XML, their attributes and operations
* elems[i] = all info about class "i"
* elems[i][0] = class id
* elems[i][1] = class name
* elems[i][2] = all class attributes
* elems[i][2][j] = each attribute (j) from the class "i"
* elems[i][3] = all class operations
* elems[i][3][j] = each operation (j) from the class "i"
* elems[i][4] = all stereotypes
* elems[i][4][j] = each stereotype (j) from the class "i"
* elems[i][5] = note id
* elems[i][6] = notes
* elems[i][7] = generalization
* elems[i][33] = abstract value
* elems[i][42] = result parse class
*/
var elems = [];
var attrs = [];
var opers = [];
var stereo = [];
var class_association = [];
var class_notes = [];
```

```
35 var class_generalization = [];
36 var classes;
37 var license = { val : "" };
38 var mainClass = { val : "" };
39 var readme = { val : "" };
40 var origFile = { val : "" };
41 var baseFile = { val : "" };
42 var error = "";
43
44
45 // lib data
46 var lib_files = [];
47 var lib_content = "";
48
49 // Variables
50 var i, j, k;
51 var counter = 0;
52 var limit = 5;
53 var jarCheck = 1;
54 var chromeCheck = 1;
55
56
57 /*
58 ****************************************
59 **************INIT CHECKS*****************
60 ****************************************
61 */
62
63 function checkLogFile() {
64   var xhReq = new XMLHttpRequest();
65   xhReq.open("HEAD", "js/scripts.js", false);
66   xhReq.send(null);
67   var lastModified = xhReq.getResponseHeader("Last-Modified");
68   var date = new Date(lastModified);
69   var weekday = new Array(7);
70   weekday[0] = "Sun";
71   weekday[1] = "Mon";
72   weekday[2] = "Tue";
73   weekday[3] = "Wed";
74   weekday[4] = "Thu";
```

```
75    weekday[5] = "Fri";
76    weekday[6] = "Sat";
77    var month = new Array(12);
78    month[0] = "Jan";
79    month[1] = "Feb";
80    month[2] = "Mar";
81    month[3] = "Apr";
82    month[4] = "May";
83    month[5] = "Jun";
84    month[6] = "Jul";
85    month[7] = "Aug";
86    month[8] = "Sep";
87    month[9] = "Oct";
88    month[10] = "Nov";
89    month[11] = "Dec";
90    var newDate = weekday[date.getDay()] + ", " + date.getDate() + " " +
91        month[date.getMonth()] + " " + date.getFullYear() + " at " +
92        date.getHours() + ":" + date.getMinutes() + ":" + date.getSeconds();
93    document.getElementById("last").innerHTML = "Code was last update on: " +
94      newDate;
95  }
96
97  // Check if exists xml file on server to parse
98  function fileExists() {
99    var http = new XMLHttpRequest();
100   http.open('HEAD', "uploads/file.xml", false);
101   http.send();
102   return http.status !== 404;
103 }
104
105 // Check some things on load page
106 function initCheck() {
107   // Check for the various File API support.
108   if (window.File && window.FileReader && window.FileList && window.Blob) {
109     var parse = document.getElementById('parse');
110     var clear = document.getElementById('clear');
111     // Check if any file are uploaded into server
112     if (fileExists()) {
113       parse.style.display = 'block';
114       parse.style.display = 'yes';
```

```
115       clear.style.display = 'block';
116       clear.style.display = 'yes';
117     }
118   }
119   else {
120     alert('The File APIs are not fully supported in this browser.');
121   }
122   checkLogFile();
123 }
124
125
126 /*
127 ******************************
128 **********MAIN FUNCTIONS*******
129 ******************************
130 */
131
132 // Print on page function
133 function cout(text) {
134   document.getElementById("errorMsg").innerHTML += text + "<br>";
135 }
136
137 function cout2(text) {
138   document.getElementById("files").innerHTML += text + "<br>";
139 }
140
141 function addToPage(sectionId, text) {
142   document.getElementById(sectionId).innerHTML += "<br>" + text + "<br>";
143 }
144
145 // Get file to parse and call main function fillElems
146 function parse() {
147   if (counter != 0) {
148     for (var i = 1; i < (counter+1); ++i) {
149       if (!checkTextElements(i)) {
150         alert("Please, fill all By element names/html value");
151         return;
152       }
153       if (!checkRadioButtons(i)) {
154         alert("Some By types are not defined");
```

```
155         return;
156       }
157     }
158   }
159   if (fileExists()) {
160     var xhttp = new XMLHttpRequest();
161     xhttp.onreadystatechange = function () {
162       if (this.readyState === 4 && this.status === 200) {
163         fillElems(this);
164       }
165     };
166     xhttp.open("GET", "uploads/file.xml", true);
167     xhttp.send();
168   }
169   else {
170     alert("Please, upload a file");
171   }
172 }
173
174 // Analize all classes into elems array. Main function
175 function fillElems(xml) {
176   var xmlDoc;
177   xmlDoc = xml.responseXML;
178   if (xmlDoc == null) {
179     alert("Please, upload a valid XML file");
180     return;
181   }
182   classes = xmlDoc.getElementsByTagName("UMLClass");
183   class_association = xmlDoc.getElementsByTagName("UMLAssociation");
184   if (class_association.length == 0) {
185     class_association = xmlDoc.getElementsByTagName("UMLLine");
186   }
187   class_notes = xmlDoc.getElementsByTagName("UMLNote");
188   class_generalization = xmlDoc.getElementsByTagName("UMLGeneralization");
189   //Analize classes
190   for (i = 0; i < classes.length; i = i + 1) {
191     getItems(i);
192   }
193   //Prepare generalization "pointer"
194   for (var n = 0; n < elems.length; ++n) {
```

```
195       elems[n][7] = "";
196     }
197     //Analize Generalizations
198     for (i = 0; i < class_generalization.length; ++i) {
199       getGeneralizations(i);
200     }
201     //Initializate notes to []. For prevent undefined pointers
202     for (var n = 0; n < elems.length; ++n) {
203       elems[n][6] = [];
204     }
205     //Analize associations
206     for (i = 0; i < class_association.length; ++i) {
207       getNotes(i);
208     }
209     //Write elements
210     for (i = 0; i < elems.length; i = i + 1) {
211       result = { val : "" };
212       writeItems(i);
213       elems[i][42] = result.val;
214     }
215     preZip();
216   }
217
218   // Get items for each class and save into elems array
219   function getItems(i) {
220     attrs = [];
221     opers = [];
222     stereo = [];
223     var elem = [], id, childs, temp, abstract;
224     id = classes[i].attributes.getNamedItem("id").value;
225     elem[0] = id;
226     abstract = classes[i].attributes.getNamedItem("abstract").value;
227     elem[33] = abstract;
228     childs = classes[i].childNodes;
229     for (j = 0; j < childs.length; j = j + 1) {
230       if (childs[j].nodeType === 1) {
231         temp = childs[j].attributes.getNamedItem("id").value;
232         if (temp === "name") {
233           getName(elem, childs, j);
234         }
```

```
235         else if (temp === "attributes") {
236           getElements(elem, childs, j, attrs, 2);
237         }
238         else if (temp === "operations") {
239           getElements(elem, childs, j, opers, 3);
240         }
241         else if (temp === "stereotypes") {
242           getElements(elem, childs, j, stereo, 4);
243         }
244       }
245     }
246     elems[i] = elem;
247     elems[i][6] = [];
248   }
249
250   // Get all attributes for all classes
251   function getElements(elem, childs, j, elements, n) {
252     var elements_temp, temp;
253     elements_temp = childs[j].childNodes;
254     for (k = 0; k < elements_temp.length; k = k + 1) {
255       if (elements_temp[k].nodeType === 1) {
256         temp = elements_temp[k].attributes.getNamedItem("value").value;
257         elements.push(temp);
258       }
259     }
260     elem[n] = elements;
261   }
262
263   // Write all info from every class
264   function writeItems(i) {
265     writeClass(i);
266     writeAttrs(i);
267     result.val += "\r\n";
268     writeOpers(i);
269     result.val += "}" + "\r\n\r\n";
270   }
271
272
273   /*
274   ***************************************
```

```
275 **************CLASS FUNCTIONS***********
276 ************************************
277 */
278
279 function checkImports(i) {
280   var imports = [];
281   result.val += "package defaultPackage;" + "\r\n\r\n";
282   for (var j = 0; j < elems[i][6].length; ++j) {
283     var temp = elems[i][6][j].split(".");
284     if (imports.indexOf(temp[0]) == -1) {
285       imports.push(temp[0]);
286     }
287   }
288   if (imports.length > 0) result.val += "\r\n";
289 }
290
291 // Get class name
292 function getName(elem, childs, i) {
293   var name = childs[i].attributes.getNamedItem("value").value;
294   elem[1] = name;
295 }
296
297 function writeClass(i) {
298   checkImports(i);
299   result.val += "public ";
300   if (elems[i][33] == "true") {
301     result.val += "abstract ";
302   }
303   result.val += "class " + elems[i][1];
304   if (elems[i][7] != "") {
305     result.val += " extends " + elems[i][7];
306   }
307   result.val += " {" + "\r\n\r\n";
308 }
309
310 function writeStereotypes(i) {
311   if (elems[i][4].length > 0) {
312     var ext = elems[i][4][0];
313     ext = ext.split("≪");
314     ext = ext[1].split("≫");
```

```
315     result.val += " stereotype " + ext[0];
316   }
317 }
318
319 /*
320 ************************************
321 **********ATTRUBUTE FUNCTIONS*********
322 ************************************
323 */
324
325 // Write attr value (if exists)
326 function writeValueAttr(type) {
327   var value = type[1].split("=");
328   result.val += value[0] + " ";
329   result.val += type[0] + " = ";
330   result.val += value[1] + ";" + "\r\n";
331 }
332
333 // Write attr type
334 function writeAttrType(visib, str) {
335   var type, n;
336   type = visib[1].split(":");
337   n = type[1].indexOf("=");
338   if (n !== -1) {
339     writeValueAttr(type);
340   }
341   else {
342     result.val += type[1] + " ";
343     result.val += type[0] + ";" + "\r\n";
344   }
345 }
346
347 //Write attribute
348 function writeAttr(n, str, scope) {
349   var visib = [];
350   if (scope != " ") {
351     visib = str.split(scope);
352   }
353   else visib[1] = str;
354   n = visib[1].indexOf(":");
```

```
355   if (n !== -1) {
356     writeAttrType(visib, str);
357   }
358 }
359 
360 // Write attrs items for every class
361 function writeAttrs(i) {
362   var check = 0;
363   for (j = 0; j < elems[i][2].length; j = j + 1) {
364     var str, n;
365     str = elems[i][2][j];
366     if (str.indexOf("-") !== -1) {
367       result.val += "    private ";
368       writeAttr(n, str, "-");
369       check = 1;
370     }
371     else if (str.indexOf("+") !== -1) {
372       result.val += "    public ";
373       writeAttr(n, str, "+");
374       check = 1;
375     }
376     else if (str.indexOf("#") !== -1) {
377       result.val += "    protected ";
378       writeAttr(n, str, "#");
379       check = 1;
380     }
381     else if (str.indexOf("~") !== -1) {
382       result.val += "    ";
383       writeAttr(n, str, "~");
384       check = 1;
385     }
386     else {
387       result.val += "    private ";
388       writeAttr(n, str, " ");
389     }
390   }
391 }
392 
393 
394 /*
```

```
395 ****************************************
396 **********OPERATION FUNCTIONS**********
397 ****************************************
398 */
399
400 function writeOperParams(params) {
401   if (params.length != 0) {
402     var list = params.split(",");
403     for (var x = 0; x < list.length; ++x) {
404       var elem = list[x].split(":");
405       result.val += elem[1] + " ";
406       result.val += elem[0];
407       if (x < (list.length-1) && list.length > 1) {
408         result.val += ", ";
409       }
410     }
411   }
412   result.val += ")" + " {" + "\r\n";
413   result.val += "        // TODO Auto-generated method stub" + "\r\n";
414 }
415
416 // Write operation
417 function writeOper(str, i) {
418   var params_type = str.split("(");
419   var name = params_type[0];
420   params_type = params_type[1].split(")");
421   var params = params_type[0];
422   var type = params_type[1];
423   if (type.length == 0) type = "void";
424   else {
425     type = type.split(":");
426     type = type[1];
427   }
428   result.val += type + " ";
429   result.val += name + " (";
430   writeOperParams(params);
431   insertNotes(i);
432   if (type != "void") result.val += "\r\n" + "        return null;";
433   result.val += "\r\n" + "    }" + "\r\n\r\n";
434 }
```

```
435
436 // Write opers for every class
437 function writeOpers(i) {
438   for (j = 0; j < elems[i][3].length; j = j + 1) {
439     var str, n;
440     str = elems[i][3][j];
441     if (str.indexOf("+") !== -1) {
442       result.val += "    public static ";
443       var val = str.split("+");
444       writeOper(val[1], i);
445     }
446     else if (str.indexOf("-") !== -1) {
447       result.val += "    private ";
448       var val = str.split("-");
449       writeOper(val[1], i);
450     }
451     else if (str.indexOf("#") !== -1) {
452       result.val += "    protected ";
453       var val = str.split("#");
454       writeOper(val[1], i);
455     }
456     else if (str.indexOf("~") !== -1) {
457       result.val += "    ";
458       var val = str.split("~");
459       writeOper(val[1], i);
460     }
461     else {
462       result.val += "    public static ";
463       var val = [];
464       val[1] = str;
465       writeOper(val[1], i);
466     }
467   }
468 }
469
470
471 /*
472 ***********************************************
473 ****NOTES, ASSOCIATION AND GENERALIZATIONS****
474 ***********************************************
```

```
475 */
476
477 //
478 function getNoteValues(noteValues, j) {
479   var childs = class_notes[j].childNodes;
480   for (var k = 0; k < childs.length; ++k) {
481     if (childs[k].nodeType === 1) {
482       temp = childs[k].attributes.getNamedItem("id").value;
483       if (temp === "description") {
484         noteValues = childs[k].attributes.getNamedItem("value").value;
485       }
486     }
487   }
488   return noteValues;
489 }
490
491 //Add notes to class file
492 function insertNotes(i) {
493   for (var j = 0; j < elems[i][6].length; ++j) {
494     result.val += "\r\n" + "        " + elems[i][6][j] + ";";
495   }
496 }
497
498 //Read all notes associated to each class
499 function getNotes(i) {
500   var id1, id2;
501   id1 = class_association[i].attributes.getNamedItem("side_A").value;
502   id2 = class_association[i].attributes.getNamedItem("side_B").value;
503   class_association[i][0] = "";
504   class_association[i][1] = "";
505   for (var n = 0; n < elems.length; ++n) {
506     elems[n][5] = "";
507     if (id1 == elems[n][0]) {
508       class_association[i][0] = id2;
509       class_association[i][1] = id1;
510       elems[n][5] = id2;
511       processNotes(n);
512       return;
513     }
514     else if (id2 == elems[n][0]) {
```

```
515         class_association[i][0] = id1;
516         class_association[i][1] = id2;
517         elems[n][5] = id1;
518         processNotes(n);
519         return;
520       }
521     }
522   }
523
524   //Process all notes associated to each class
525   function processNotes(n) {
526     var noteValues = "";
527     for (var j = 0; j < class_notes.length; ++j) {
528       var id = class_notes[j].attributes.getNamedItem("id").value;
529       if (id == elems[n][5]) {
530         noteValues = getNoteValues(noteValues, j);
531       }
532     }
533     var values = noteValues.split(";");
534     for (var i = 0; i < values.length; ++i) {
535       if (values[i] != "") {
536         elems[n][6].push(values[i]);
537       }
538     }
539   }
540
541   //Process generalization tag
542   function getGeneralizations(i) {
543     var id1, id2, name1, name2, j;
544     id1 = class_generalization[i].attributes.getNamedItem("side_A").value;
545     id2 = class_generalization[i].attributes.getNamedItem("side_B").value;
546     name1 = "";
547     name2 = "";
548     for (var n = 0; n < elems.length; ++n) {
549       if (id1 == elems[n][0]) {
550         name1 = elems[n][1];
551         j = n;
552       }
553       if (id2 == elems[n][0]) {
554         name2 = elems[n][1];
```

```
555       }
556       if (name1 != "" && name2 != "") {
557         elems[j][7] = name2;
558       }
559     }
560   }
561
562   /*
563   ******************************************
564   ******PROCESS FILES AND EXTRAS**************
565   ******************************************
566   */
567
568   //Add new field for "By" elements
569   function addInput(divName){
570     var newCounter = counter + 1;
571     var newdiv = document.createElement('div');
572     newdiv.setAttribute("id", "By" + newCounter);
573     newdiv.setAttribute("name", "myInputs[]");
574     newdiv.innerHTML = " <br>Name: <input type='text' id='name_" + newCounter +
575       "'>   html value: <input type='text' id='html" +
576       newCounter + "'>   <input type='radio' id='id" + newCounter +
577       "' name='ByType" + newCounter +
578       "' value='id'> id <input type='radio' id='xpath" + newCounter +
579       "' name='ByType" + newCounter +
580       "' value='xpath'> xpath <input type='radio' id='name" + newCounter +
581       "' name='ByType" + newCounter +
582       "' value='name'> name <input type='radio' id='css" + newCounter +
583       "' name='ByType" + newCounter + "' value='css'> css<br>";
584     document.getElementById(divName).appendChild(newdiv);
585     counter++;
586     if (counter == limit)  {
587       var element = document.getElementById("addField");
588       element.style.display = 'none';
589     }
590     if (counter > 0) {
591       var element = document.getElementById("delField");
592       element.style.display = 'inline';
593       element.style.display = 'yes';
594     }
```

```
595 }
596
597 //Delete last "By" element added
598 function deleteElem() {
599   var elem = document.getElementById('By'+counter);
600   elem.parentNode.removeChild(elem);
601   counter--;
602   if (counter == 0) {
603     var element = document.getElementById("delField");
604     element.style.display = 'none';
605   }
606   if (counter < limit) {
607     var element = document.getElementById("addField");
608     element.style.display = 'inline';
609     element.style.display = 'yes';
610   }
611   return false;
612 }
613
614 //Check if a radiobutton are checked for all "By" elements
615 function checkRadioButtons(i) {
616   if (document.getElementById("id"+i).checked) return true;
617   else if(document.getElementById("xpath"+i).checked) return true;
618   else if(document.getElementById("name"+i).checked) return true;
619   else if(document.getElementById("css"+i).checked) return true;
620   return false;
621 }
622
623 //Check name of all "by" elements
624 function checkTextElements(i) {
625   var name = document.getElementById("name_"+i).value;
626   var html = document.getElementById("html"+i).value;
627   html = html.split(" ").join("");
628   if (html == "") return false;
629   name = name.split(" ").join("");
630   if (name == "") return false;
631   return true;
632 }
633
634 //Read wich "By" elements to add
```

```
635 function readTextFields() {
636   for (var i = 1; i <= counter; ++i) {
637     var name = document.getElementById("name_"+i).value;
638     var html = document.getElementById("html"+i).value;
639     baseFile.val += "    private By " + name + " = ";
640     if (document.getElementById("id"+i).checked) {
641       baseFile.val += "By.id('" + html + "');" + "\r\n";
642     }
643     else if(document.getElementById("xpath"+i).checked) {
644       baseFile.val += "By.xpath('" + html + "');" + "\r\n";
645     }
646     else if(document.getElementById("name"+i).checked) {
647       baseFile.val += "By.name('" + html + "');" + "\r\n";
648     }
649     else if(document.getElementById("css"+i).checked) {
650       baseFile.val += "By.cssSelector('" + html + "');" + "\r\n";
651     }
652   }
653 }
654
655 //Create a prompt with pre-defined text to copy
656 function copyToClipboard() {
657   var xmlText = { val : "" };
658   readFile("static_files/example.txt", xmlText);
659   window.prompt("Copy to clipboard: Ctrl+C, Enter", xmlText.val);
660 }
661
662 //Show "loading" animation while building zip file
663 function loading() {
664   var files, loading, parse, download;
665   files = document.getElementById('files');
666   files.style.display = 'none';
667   parse = document.getElementById('parse');
668   parse.style.display = 'none';
669   download = document.getElementById('download');
670   download.style.display = 'block';
671   download.style.display = 'yes';
672   loading = document.getElementById('loader');
673   loading.style.display = 'block';
674   loading.style.display = 'yes';
```

```
675 }
676
677 //When zip file is created, hide "loading" animation
678 function loaded() {
679   var files, loading, parse, download;
680   files = document.getElementById('files');
681   files.style.display = 'block';
682   parse = document.getElementById('parse');
683   parse.style.display = 'block';
684   parse.style.display = 'yes';
685   download = document.getElementById('download');
686   download.style.display = 'block';
687   download.style.display = 'none';
688   loading = document.getElementById('loader');
689   loading.style.display = 'none';
690 }
691
692 // Read data from file
693 function readFile(file, variable) {
694   var rawFile = new XMLHttpRequest();
695   rawFile.open("GET", file, false);
696   rawFile.onreadystatechange = function () {
697     if (rawFile.readyState === 4) {
698       if (rawFile.status === 200 || rawFile.status === 0) {
699         variable.val += rawFile.responseText;
700       }
701     }
702   };
703   rawFile.send(null);
704 }
705
706 // Show/Hide toggle buttons alert
707 function doalert(checkboxElem) {
708   if (!checkboxElem.checked) {
709     if (checkboxElem.name == "jar") {
710       var jar = document.getElementById("jarAlert");
711       jar.style.display = 'inline-block';
712       jar.style.display = 'yes';
713       jarCheck = 0;
714     }
```

```
715       else {
716         var chrome = document.getElementById("chromeAlert");
717         chrome.style.display = 'inline';
718         chrome.style.display = 'yes';
719         chromeCheck = 0;
720       }
721     }
722     else {
723       if (checkboxElem.name == "jar") {
724         var jar = document.getElementById("jarAlert");
725         jar.style.display = 'none';
726         jarCheck = 1;
727       }
728       else {
729         var chrome = document.getElementById("chromeAlert");
730         chrome.style.display = 'none';
731         chromeCheck = 1;
732       }
733     }
734   }
735
736   // Check toggle buttons to decide wich zip file create
737   function preZip() {
738     if (document.getElementById("jarFiles").checked) {
739       if (document.getElementById("chromeDriver").checked)
740         zip("uploads/X2JProject.zip");
741       else zip("uploads/X2JProject_lite2.zip");
742     }
743     else if(document.getElementById("chromeDriver").checked)
744       zip("uploads/X2JProject_lite.zip")
745     else zip("uploads/X2JProject_lite3.zip");
746   }
747
748   // Generate .zip with all files
749   function zip(path) {
750     loading();
751     JSZipUtils.getBinaryContent(path, function (err, data) {
752       if(err) {
753         throw err; // or handle the error
754       }
```

```
755     JSZip.loadAsync(data)
756       .then(function (zip) {
757       //ADD MY FILES
758       var rootFolder = zip.folder("X2JProject");
759       var files = zip.folder("X2JProject/src/defaultPackage");
760       for (i = 0; i < elems.length; i = i + 1) {
761         files.file(elems[i][1] + ".java", elems[i][42]);
762       }
763       readFile("uploads/file.xml", origFile);
764       rootFolder.file("MyProject.xml", origFile.val);
765       readFile("static_files/license.txt", license);
766       rootFolder.file("license.txt", license.val);
767       readFile("static_files/readme.txt", readme);
768       rootFolder.file("readme.txt", readme.val);
769       readFile("static_files/BaseClass1.txt", baseFile);
770       readTextFields();
771       readFile("static_files/BaseClass2.txt", baseFile);
772       var url = document.getElementById('baseURL').value;
773       if (url != "") baseFile.val += url;
774       else baseFile.val += "www.tonimiquel.es"
775       readFile("static_files/BaseClass3.txt", baseFile);
776       files.file("ExampleClass.java", baseFile.val);
777       //zip.file("new_file", "new_content");
778       // if you return the zip object, it will be available in the next "then"
779       return zip;
780     }).then(function (zip) {
781       // if you return a promise of a blob, promises will "merge": the current
782       // promise will wait for the other and the next "then" will get the blob
783       return zip.generateAsync({type: "blob"});
784     }).then(function (blob) {
785       saveAs(blob, "X2JProject.zip");
786       loaded();
787     });
788   });
789 }
```

D upload.php

Codi en PHP que ens permet pujar un fitxer XML al servidor

```php
<?php

$target_dir = "../uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$errorCheck = 0;
$FileType = pathinfo($target_file,PATHINFO_EXTENSION);

// Check file selected
if ($target_file == "../uploads/") {
  $errorCheck = 1;
}

// Check file size
else if ($_FILES["fileToUpload"]["size"] > 500000) {
  $errorCheck = 2;
}

// Allow certain file formats
else if($FileType != "xml") {
  $errorCheck = 3;
}

// Check if $uploadOk is set to 0 by an error
if ($errorCheck != 0) {
  if ($errorCheck == 1) $message = "Please, select a file";
  else if ($errorCheck == 2) $message = "Sorry, your file is too large";
  else $message = "Sorry, only XML files are allowed";
  echo "<script type='text/javascript'>alert('$message');
  window.history.back();</script>";
  // if everything is ok, try to upload file
} else {
  if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"],
  "../uploads/" . "file.xml")) {
    echo "<script type='text/javascript'>window.location='../index.html';</script>";
  } else {
    $message = "Error uploading file";
```

```
37      echo "<script type='text/javascript'>alert('$message');
38      window.history.back();</script>";
39    }
40  }
41
42  ?>
```

E clearData.php

Codi en PHP per eliminar el fitxer pujat al servidor

```
1 <?php
2 if (file_exists('../uploads/file.xml')) {
3   unlink("../uploads/file.xml");
4 }
5 echo "<script type='text/javascript'>alert('All data cleared!');</script>";
6 echo "<script type='text/javascript'> window.location='../index.html';</script>";
7 ?>
```