10. Gestió de sessions: GnomeClient

Gràcies al protocol X Session Management (XSMP)\footnote{http://refsspecs.freestandards.org/X11/SMlib.pdf}, les aplicacions d’X-Window poden disposar d’un sistema estandarditzat per a la gestió de sessions. El protocol XSMP permet notificar a les aplicacions la finalització de la sessió actual (mitjançant biblioteques de la part client) perquè puguin ser reiniciades de nou quan l’usuari torni a entrar de nou (mitjançant un servei de gestió de sessions). La biblioteca de desenvolupament d’aplicacions d’X, Xlib, incorpora un mòdul per treballar amb el protocol XSMP, anomenat SMlib. Aquesta biblioteca pot ser emprada per qualsevol aplicació X, però requereix disposar d’experiència prèvia en la programació d’aquest entorn.

L’escriptori Gnome fa servir el protocol de gestió de sessions XSMP. Es proporcionen els següents components:

- **GnomeClient**, una classe GObject que embolcalla el funcionament del protocol XSMP. Aquesta classe disposa de mètodes perquè les aplicacions puguin rebre notificacions en tancar i iniciar la sessió, i simplifica el treball amb el protocol XSMP gràcies a una API orientada a objectes com les de GTK+/Gnome.

- **gnome-session**, el gestor de sessions del Gnome. Tan bon punt un usuari inicia una nova sessió, aquest dimoni executa les aplicacions predeterminades de l’escriptori (p.e. els quadres i el gestor de fitxers Nautilus) i torna a executar les aplicacions que estaven obertes la darrera vegada que es va sortir de la sessió.

10.1. Inicialització de la gestió de sessions

Per poder suportar el treball amb sessions, les aplicacions han d’obtenir un identificador del gestor de sessions. Amb aquest identificador s’informa el gestor de com s’ha de reinstallar la nostra aplicació, i permet que es rebin notificacions quan es produeixen canvis en la sessió. El gestor gnome-session s’inicia automàticament quan l’usuari fa login, de manera que les aplicacions només han d’obtenir-ne una.
Al Gnome, les aplicacions fan servir la classe `GnomeClient` per disposar d’aquests serveis. Hi ha dues maneres d’obtenir un objecte `GnomeClient`:

- Les aplicacions del Gnome solen cridar `gnome_program_init()` al començament per inicialitzar les biblioteques de l’entorn. En aquest cas, `gnome_program_init()` haurà inicialitzat un gestor de sessions predeterminat per nosaltres. Per obtenir-lo, només cal fer servir la crida `gnome_master_client()`.

- En cas contrari, es pot obtenir un nou objecte `GnomeClient` mitjançant la crida `gnome_client_new()`.

Una vegada es disposa d’una connexió amb el gestor de sessions, cal notificar com s’ha de cridar la nostra aplicació perquè es pugui reiniciar adequadament:

- Amb `gnome_client_set_restart_style (GnomeClient *client, GnomeRestartStyle style)` s’informa al gestor de sessions de quan cal reiniciar l’aplicació. L’enumeració `GnomeRestartStyle` pot prendre els palms `GNOME_RESTART_IF_RUNNING` (es reinicia si s’estava executant), `GNOME_RESTART_ANYWAY` (reinicia sempre), `GNOME_RESTART_IMMEDIATELY` (es reinicia tan bon punt l’aplicació surti o mori), `GNOME_RESTART_NEVER` (no s’ha de tornar a iniciar).

- Amb `gnome_client_set_restart_command (GnomeClient *client, gint argc, gchar* argv[])` s’estableix l’ordre que cal executar per reiniciar l’aplicació. El vector `argv[]` ha de contenir com a mínim el camí de l’aplicació.

Aquests mètodes se solen cridar des de la funció `main()`. `GnomeClient` no estableix cap mètode per desar la configuració de l’estat actual de l’aplicació; correspon a les aplicacions fer-ne servir el que trobin més convenient (per exemple, amb paràmètres especials passats a la crida `gnome_client_set_restart_command()` o mitjançant claus `GConf`).

### 10.2. Notificació de la finalització de la sessió

Els objectes `GnomeClient` emeten els següents signals per informar del fi de sessió:

- "save-yourself": aquest signal és una petició del gestor de sessions perquè l’aplicació desitja el seu estat actual. La signatura del seu callback conté força paràmètres:

```c
gboolean user_function (GnomeClient *client,
                      gint arg1,
                      GnomeSaveStyle arg2,
                      gboolean arg3,
                      GnomeInteractStyle arg4,
                      gboolean arg5,
                      gpointer user_data);
```

---

50 [http://developer.gnome.org/doc/API/2.0/libgnomeui/GnomeClient.html](http://developer.gnome.org/doc/API/2.0/libgnomeui/GnomeClient.html)

51 Tot i que `GnomeClient` permet obtenir un prefix per desar la sessió mitjançant l’API `gnome-config`, es desaconseixa fer servir aquest mètode. En aquests casos, és preferible fer servir `GConf` per emmagatzemar les dades de la sessió.
Algums d'aquests paràmetres rebuts són innecessaris per a la majoria d'aplicacions. Un dels paràmetres importants és `GnomeInteractStyle`, que indica com es pot interaccionar amb l'usuari (per exemple, per preguntar-li si vol desar els fitxers oberts abans de sortir).

- "die": indica que l'aplicació ha de tancar-se immediatament.

### 10.3. Recuperació de l'estat anterior en iniciar la sessió

Les aplicacions han de demanar explícitament si han sigut reiniciades pel gestor de sessions o iniciades per l'usuari. Es poden fer servir aquests dos mitjans:

- amb paràmetres d'inici especials que es van establir en la crida `gnome_client_set_restart_command()`

- demanar-ho al gestor de sessions mitjançant `gnome_client_get_flags(GnomeClient* client)`, que retorna una màscara de valors `GnomeClientFlags`. Aquesta enumeració té els següents valors:

```c
typedef enum
{
    GNOME_CLIENT_IS_CONNECTED = 1 << 0,
    GNOME_CLIENT_RESTARTED   = 1 << 1,
    GNOME_CLIENT_RESTORED    = 1 << 2
} GnomeClientFlags;
```

En el cas que la nostra aplicació hagi estat reiniciada pel gestor de sessions, el flag `GNOME_CLIENT_RESTARTED` valdrà 1. En canvi, si l'aplicació s'ha executat des d'un menú o des d'una terminal, el flag estarà inhabilitat.

### 10.4. Exemple d'aplicació amb gestió de sessions

La següent aplicació mostra la interacció amb el gestor de sessions mitjançant `GnomeClient`. S'ha ignorat els detalls d'emmagatzematge de la configuració de l'estat:

```c
#include <gnome.h>

gboolean
on_save_session (GnomeClient *client,
                        gint arg1,
                        GnomeSaveStyle arg2,
                        gboolean arg3,
                        GnomeInteractStyle arg4,
                        gboolean arg5,
                        gpointer user_data)
{
    g_assert (SALEENA_IS_TABS_GUI (obj));

    /* Desa l'estat actual de l'aplicació */
    return TRUE;
}
```
void
on_die_session (GnomeClient *client, gpointer data)
{
    /* Tanca l’aplicació immediatament */

gtk_main_quit ();
}

int main (int argc, char* argv)
{
    gnome_program_init ("session",
        "0.1",
        LIBGNOMEUI_MODULE,
        argc, argv,
        GNOME_PROGRAM_STANDARD_PROPERTIES,
        NULL);

    GnomeClient* client = gnome_master_client ();

    gnome_client_set_restart_style (client, GNOME_RESTART_IF_RUNNING);

    gnome_client_set_restart_command (client, 1 , argv);

    /* Comprova si l’aplicació ha sigut reiniciada pel gestor de sessions */

    GnomeClientFlags flags = gnome_client_get_flags (client);

    if (flags & GNOME_CLIENT_RESTARTED)
    {
        /* Recupera la sessió anterior */
    }
    else
    {
        /* Inicia l’aplicació sense recuperació */
    }

    /* Connecta el signal per desar l’estat actual de l’aplicació */

    g_signal_connect (G_OBJECT (client),
        "save-yourself",
        G_CALLBACK (on_save_session),
        NULL);

    /* Connecta el signal per tancar l’aplicació quan ho digui el gestor de sessions */

    g_signal_connect (G_OBJECT (client),
        "die",
        G_CALLBACK (on_die_session),
        NULL);
}
11. Saleena: lector de canals RSS/RDF per a l’escriptori Gnome

La sindicació de continguts és una de les aplicacions més emprades per distribuir continguts a Internet. Mitjançant senzills fitxers XML, els usuaris poden obtenir informació actualitzada a diari dels seus serveis de notícies, actualitzacions de seguretat o blogs personals. Aquests fitxers de continguts, anomenats feeds (canals) es troben allotjats en un servidor web i s’actualitzen a la vegada que les pròpies pàgines. Tot i ser una tecnologia desenvolupada fa anys la seva popularització és força recent, principalment gràcies a dues de les seves aplicacions: els podcasts i els resums del contingut dels blogs.

Com a exemple d’aplicació de les tecnologies de desenvolupament disponibles a l’escriptori Gnome, s’ha desenvolupat un software per a la visualització i navegació de canals de sindicació RSS/RDF, anomenat Saleena. Amb aquesta aplicació, els usuaris defineixen un conjunt de categories sota les quals agrupar els canals (p.e. Informació, Entreteniment, Blogs personals) i poden navegar pels continguts dels canals mitjançant una interfície HTML generada dinàmicament, basada en els continguts dels canals descarregats. D’aquesta manera, s’accedeix a la informació de manera semblant als populars portals d’Internet, amb la diferència que és el propi usuari qui selecciona les seves fonts d’informació:

![Pantalla principal de l'aplicació, mostrant les categories definides per l'usuari. El contingut de canals i categories es pot visualitzar en pestanyes separades.](image-url)
Actualment, existeixen tres tecnologies diferents per a la sindicació de continguts:

- **RSS**, que fa servir un vocabulari XML per especificar el contingut. Existeixen diferents revisions d’RSS, però les més populars són **RSS (Rich Site Summary) 0.91** i **RSS (Really Simple Indication) 2.0**.

- **RSS 1.0 (RDF Site Summary)**, fa servir part del vocabulari **RDF** (basat en XML) per especificar el contingut. Tot i ser semblant a **RSS**, tenen certes diferències menors que els fan incompatibles. El principal avantatge sobre **RSS** és la seva extensiblitat mitjançant vocabularis **RDF**.

- **Atom**, una altra tecnologia per a la sindicació de continguts, desenvolupada com a alternativa a **RSS** i **RDF**. Fa servir un vocabulari XML totalment diferent als anteriors.

Tot i la diversitat de formats, la seva simplicitat fa que molt llocs web optin per oferir el seu contingut en més d'un format. Saleena implementa la lectura dels formats **RSS i RDF**, ja que es tracten dels formats més populars. Aquesta aplicació ha estat desenvolupada en C, fent servir les biblioteques pròpies de l'escriptori i del sistema operatiu GNU/Linux.

### 11.1. Descripció de les tecnologies per a la sindicació de continguts

#### 11.1.1. Casos d’ús per a la sindicació de continguts

Gràcies a la senzillesa conceptual de la sindicació de continguts (una llista de canvis actualitzada periòdicament), es troben força aplicacions a aquesta tecnologia. Un dels aspectes més interessants introduïts per la sindicació de continguts és el canvi d'un model tradicional d'obtenció d'informació des d'un tipus *push* (on l'usuari reb la informació) a un model tipus *pull* (l'usuari escull la informació que realment l'interessa). Com es veurà, molts d'aquests usos tenen un fort component socialitzador, ja que permeten als usuaris crear les seves fonts d'informació personalitzada amb tot allò que realment els interessa.

- **Blogs**. El fenomen dels blogs és un dels més importants des de la popularització de la web. Milions de persones en tot el món fan servir blogs personals per explicar les seves aficions o les novel· tats relacionades amb la seva activitat professional. També, moltes organitzacions els i companyies els fan servir per informar dels avenços diaris en els seus productes. Els blogs han col·laborat en gran mesura en les capacïtats socialitzadores de la xarxa, i gairebé qualsevol portal de serveis (p.e. MSN, Google) o web amb gran nombre de visitants (vilaweb.cat, 20minutos.es) ofereixen aquest servei gratuïtament a les seves comunitats d'usuaris.

Els blogs fan servir la sindicació de continguts per permetre obtenir les entrades més recents. L'autor sol disposar d’alguns paràmetres de configuració, com ara el nombre d’entrades a mostrar o el contingut de cada entrada (abreviat o complet). A banda d’un feed per al contingut principal, també es proporcionen múltiples feeds per cada categoria, fins i tot es pot obtenir un feed d’un fil de comentaris d’una entrada.

- **Podcasts**. Juntament amb els blogs, els podcasts han demostrat ser un dels serveis més emprats pel internauta. Un podcast és semblant a un feed d’un blog, però en comptes de distribuir el contingut en forma de text es fan servir fitxers d’àudio. De manera semblant existeixen podcasts per a vídeo. Aquesta tecnologia, popularitzada pel servei iTunes d'Apple, permet que els usuaris
cre'n els seus propis canals de ràdio o de televisió. A banda dels usuaris particulars, els podcasts són emprats per les grans companyies com una forma més d'informar dels nous aspectes de les seves tecnologies. També, moltes emissoress de ràdio permeten accedir a la programació de dies anteriors mitjançant podcasts.

Un podcast s’implementa mitjançant un feed de notícies, que conté un element especial per identificar que el contingut és un fitxer multimèdia en comptes d’un text. Els reproductors multimèdia solen permetre subscriure’s a un podcast, i en reproduir en els fitxers com si es tractés d’una llista de reproducció.

- **Webs d’informació.** Les webs de notícies van ser les primeres en fer servir la sindicació de continguts per distribuir informació actualitzada al minut. Aquestes webs permeten subscriure’s a diferents categories (esports, societat, internacional, etc.), i sovint proporcionen senzilles miniaplicacions que descarreguen els titulars i els mostren en una finestra de l’escriptori.

- **Planetes.** Un planeta té una funció socialitzadora: agrupar tots els blogs dels membres d’una comunitat per crear un meta-blog. Cada membre registra quina categoria del seu blog es mostarà en el planeta de la seva comunitat, i el software s’encarrega de descarregar el contingut més recent de tots els subscriptors. Les entrades es poden visualitzar mitjançant una pàgina web o descarregant el feed del planeta.

- **Llistes d’actualitzacions.** Molts softwares fan servir un feed per llistar els darrers canvis que s’han produït en el sistema. Per exemple, els wikis permeten obtenir un feed que llista els canvis recents en una pàgina, i els blogs disposen de feeds per seguir el fil de comentaris d’una entrada. Un altre software que es beneficia d’aquestes llistes de canvis són els sistemes de control de versions, que notifiquen els darrers commits en un repositori mitjançant feeds. Un altre dels usos habituals és la distribució d’avisos de seguretat i actualitzacions de programari.

### 11.1.2. Introducció als formats per a la sindicació

Tot i existir varius formats per a la sindicació de continguts, el cert és que tres s’assemblen força. Tanmateix, petites diferències els fan ser incompatibles. Les tres tecnologies tenen els següents aspectes comuns:

- **XML.** Els tres formats fan servir el llenguatge XML. Mentre que **RSS 0.91/2.0** i **Atom** tenen un vocabulari XML propi, **RSS 1.0** agafa molts elements del vocabulari **RDF**, com ara els atributs "rdf:about" per indicar la URL de la notícia original.

- **Estructura del document.** En general, els tres formats fan servir una estructura semblant: una capçalera i un cos. La capçalera conté el nom del lloc web originador del feed, la url de la web, la data de creació i algunes metadades més. El cos del document està format per múltiples entrades d’un canal; cada entrada sol contenir el títol, l’enllaç a la URL original i el contingut (HTML o text).

- **Extensibilitat mitjançant espais de noms.** Emprant les capacitat que proporcionen els espais de noms XML, cada format disposa d’extensions per a certs usos específics. Per exemple, **RSS 2.0** fa servir els tags addicionals d’*<itunes>* per implementar els podcasts d’Itunes, mentre que **RSS 1.0** fa servir un conjunt d’extensions anomenades **Dublin Core** que permeten especificar molta més metainformació.
11.1.3. Descripció dels formats RSS 0.91, RSS 1.0, RSS 2.0 i Atom

Un dels aspectes més problemàtics de les tecnologies de sindicació és la manca d’un format únic, un fet difícil d’entendre tenint en compte la gran semblança conceptual entre els quatre. Cada format disposa de vàries revisions que miren de corregir defectes i ambigüitats; tanmateix aquestes modificacions també van provocar incompatibilitats entre les versions. Actualment, els llocs web fan servir aquestes tecnologies de sindicació:

- **RSS 0.91 (Rich Site Summary)**, creat per Userland Software, és la primera versió de RSS que es va fer servir ampliament. Es va desenvolupar sobre el format **RSS 0.90**, originalment creat per Netscape per al portal My Netscape l’any 1999. La versió 0.91 tenia petites diferències amb la 0.90, principalment n’eliminava l’ús d’RDF en alguns elements. Un feed RSS 0.91 té la següent sintaxi:

```xml
<rss version="0.91">
  <channel>
    <title>Canal d’exemple</title>
    <link>http://exemple.com/</link>
    <description>Un canal d’exemple</description>
    <language>ca</language>
    <webmaster>wm@example.com</webmaster>
    <pubDate>Dec 27th, 16:00:00</pubDate>
    <image>
      <url>http://example.com/imatge0.png</url>
      <title>Logotip</title>
      <link>http://example.com</link>
    </image>
    <textinput>
      <title>Cerca en aquesta web</title>
      <description>Cerca</description>
      <name>consulta</name>
      <link>http://exemple.com/cerca</link>
    </textinput>
    <skipHours>
      <hour>24</hour>
    </skipHours>
    <item>
      <title>1 &lt; 2</title>
      <link>http://example.com/1_menor_que_2.html</link>
      <description>1 &lt; 2, 3 &lt; 4.
En HTML, &lt;b&gt;comença una frase en negreta, i &lt;a href= comença un enllaç.</description>
    </item>
  </channel>
</rss>
```

Un element `<channel>` conté un conjunt d’elements que proporcionen metainformació quant al canal, per exemple `<title>`, `<link>`, `<description>`, `<language>`, `<pubDate>`, `<image>`. Hi ha d’altres elements que gairebé mai no es fan servir, com ara `<textInput>`, que hauria de ser visualitzar en l’agregador com a un `<INPUT TYPE=TEXT>` d’HTML per poder disposar de formularis senzills, o `<skipHours>`, que indica quant de temps esperar fins a obtenir el feed de nou.

Un `<channel>` té múltiples `<item>`, cadascun per a un dels elements que formen el feed de notícies. Cada item té un camp `<title>`, `<link>` i `<description>`. Ni `<title>` ni `<description>` especificuen quin tipus de dades contenen, però habitualment es tracta d’HTML.

52 [http://backend.userland.com/rss091](http://backend.userland.com/rss091)
Paradoxalment, va existir una altra versió d’**RSS 0.91** desenvolupada per Netscape. La diferència principal es trobava en la definició d’entitats addicionals en el DTD que es podien fer servir en el document. Tanmateix la versió de Netscape no va tenir èxit i es va fer servir la de Userland.

Posteriorment, Userland va crear tres revisions d’**RSS 0.91**: **RSS 0.92**, que especificava que el camp `<description>` s’havia de considerar HTML; **RSS 0.93**, que agafia un element `<expirationDate>`, i **RSS 0.94**, que eliminava l’element `<expirationDate>` i afegia un atribut `type` a l’element `<description>` per discriminar el tipus MIME del contingut. Tanmateix, aquestes revisions s’han fet servir molt rarament; la majoria de llocs web fan servir **RSS 0.91**.

- **RSS 1.0 (RDF Site Summary)** va ser creat a partir d’**RSS 0.90**, i fa servir elements del vocabulari RDF. RDF és un llenguatge XML que permet expressar propietats i relacions entre recursos, identificats mitjançant una URL. El vocabulari RDF és molt extens, però **RSS 1.0** només en fa servir una part. Un feed **RSS 1.0** té els següents aspectes:

```xml
<?xml version="1.0"?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns="http://purl.org/rss/1.0/"
 <channel rdf:about="http://www.xml.com/news.rss">
 <title>XML.com</title>
 <link>http://xml.com/pub</link>
 <description>XML.com features a rich mix of information and services for the XML community.</description>
 <items>
   <rdf:Seq>
   </rdf:Seq>
   </items>
 </channel>
 <title>Processing Inclusions with XSLT</title>
 <description>Processing document inclusions with general XML tools can be problematic. This article proposes a way of preserving inclusion information through SAX-based processing.</description>
 </item>
</rdf:RDF>
```

**RSS 1.0** es diferencia d’**RSS 0.91** en els següents aspectes:

- L’arrel del document és un element `<RDF>`, a diferència d’**RSS 0.91**, que fa servir un element `<rss>`.

- També es diferent l’ús de l’element `<channel>`, que en **RSS 1.0** només conté la informació de la capçalera.

53 [http://web.resource.org/rss/1.0/](http://web.resource.org/rss/1.0/)
Visualitzador i organitzador de canals RSS/RDF per a l'escritori GNOME

- <channel> conté un nou element, <items>, que conté una llista dels elements <item> del feed. <items> fa servir l'element <rdf:Seq> del vocabulari RDF per a crear còl·leccions d'elements. A RDF, els elements es referencien mitjançant una URL que els identifica unívocament (tot i que no cal que existeixi realment), per això cada element de la llista <rdf:Seq> referencia una url mitjançant l'atribut "resource".

- Els elements <channel>, <image>, <item> i <textinput> han de contenir un nou atribut rdf:about. Al vocabulari RDF, els elements s'identifiquen unívocament mitjançant una URL (tot i que no cal que existeixi realment). L'atribut rdf:about conté aquesta URL.

- RSS 1.0 permet ser extès mitjançant espais de noms XML. L'extensió més popular és el vocabulari de Dublin Core54, que proporciona metadades addicionals per establir l'autor, el llenguatge, la data de creació, el copyright, etc. També, mitjançant el vocabulari Syndication55 els agregadors poden obtenir informació precisa sobre el període d'actualització dels feeds. El següent feed fa servir força vocabularis addicionals per definir el seu contingut:

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:taxo="http://purl.org/rss/1.0/modules/taxonomy/"
 xmlns:rdfs="http://purl.org/rss/1.0/modules/syndication/"
 xmlns:wbns="http://webns.net/mvcb/"
 <channel rdf:about="http://www.20minutos.es/" usercontent="true">
   <title>20Minutos</title>
   <link>http://www.20minutos.es</link>
   <description>Diario de información general y local más leído de España, noticias de última hora de España, mundo, local, deportes, noticias curiosas y más</description>
   <pubDate>2006-12-27T17:10:31+01:00</pubDate>
   <lastBuildDate>2006-12-27T17:10:31+01:00</lastBuildDate>
   <language>es</language>
   <syndicationBase>http://www.20minutos.es/noticia/1861590</syndicationBase>
   <syndicationTitle>Tropas etíope desplazadas por Somalia</syndicationTitle>
   <items>
     <item>
       <rdf:Seq>
         <rdf:li>
          <rdf:li rdf:resource="http://www.20minutos.es/noticia/186159/0/etiopia/somalia/tropas/"

         </item>
       </rdf:Seq>
     </item>
   </items>
 </channel>
```

54 http://web.resource.org/rss/1.0/modules/dc/
55 http://web.resource.org/rss/1.0/modules/syndication/
rss:about="http://www.20minutos.es/noticia/186144/0/aerolinea/alemana/air-madrid/"
<title>La aerolínea alemana LTU se reúne con la directiva de Air Madrid para comprar la compañía</title>
<link>http://www.20minutos.es/noticia/186144/0/aerolinea/alemana/air-madrid/</link>
<description>Le interesa adquirirla &quot;total&quot; o &quot;parcialmente&quot; para cubrir vuelos desde Madrid y Barcelona a Latinoamérica. También el Grupo Marsans ha mostrado interés.</description>
<dc:creator>2006-12-27T14:05:13+01:00</dc:creator></item></rdf:RDF>

- RSS 2.0 (Really Simple Syndication). Tot i que el número de versió pugui indicar el contrari, RSS 2.0 no és una actualització de la versió RSS 1.0, sinó una actualització de la versió RSS 0.94. Per tant, no fa servir el vocabulari RDF. Amb l'increment del número de versió, també es va canviar el significat de l'acrònim, per tal de fer palesa la pretensió simplificadora de la nova versió (per exemple, es va suprimir l'element &lt;textInput&gt;). Un feed RSS 2.0 té el següent aspecte:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?><rss version="2.0"
<channel>
  <title>art.gnome.org releases</title>
  <image>
    <link>http://art.gnome.org</link>
    <url>http://art.gnome.org/images/site/art-icon.png</url>
  </image>
  <description>A list of recent backgrounds and themes released on art.gnome.org</description>
  <webMaster>thom@nospam.gnome.org</webMaster>
  <item>
    <title>[Application Themes] Aquarius</title>
    <link>http://art.gnome.org/themes/gtk2/1325</link>
    <guid>http://art.gnome.org/themes/gtk2/1325</guid>
    <pubDate>Thu, 28 Dec 2006 19:19:51 -0500</pubDate>
    <description>
      <![CDATA[<table border="0" style="margin-bottom:1em;">
      </table></td></tr></table>]]>
    </description>
  </item>
</channel>
</rss>
```

Com es pot veure, la sintaxi no es diferencia gaire del format RSS 0.91, i no fa servir cap espai de noms addicional per descriure el seu contingut, com fa RSS 1.0 amb RDF. Algunes dels elements addicionals inclosos a RSS 2.0 inclouen:

- <enclosure>: aquest element permet referenciar un recurs multimèdia. Conté tres atributs: length (mida), url (url del recurs) i type (tipus MIME). Els podcasts fan servir aquest element per referenciar els fittxes de so pertanyents a una entrada. Tanmateix, només es pot especificar un element <enclosure> per <item>.

56 http://blogs.law.harvard.edu/tech/rss
• <guid>: una cadena que identifica unívocamente un elemento en el feed. Permet als lectors de feeds determinar quins elements són nous. Té un funcionament semblant als atributs "rdf:about" que es fan servir a RSS 1.0.

• <ttl>: el nombre de segons que romandrà vàlid, des que es va crear.

• <category>: un nom de categoria a la qual associar un <item>. Rarament emprat.

• <cloud>: l’adreça d’un servei web SOAP on connectar-se per rebre notificacions quan es produeixin actualitzacions en el feed. Rarament emprat.

RSS 2.0 permet ser estès mitjançant d’altres espais de noms, sempre que l’espai de noms es declarei prèviament en l’element <rss>. Un dels espais de noms més populars és d’iTunes (xmlns:iTunes="http://www.itunes.com/dtds/podcast-1.0.dtd" version="2.0"), i permet implementar podcasts mitjançant feeds RSS 2.0. Els elements <itunes> afegeixen informació adicional, com ara l’autor d’un podcast, la durada del fitxer de so a un <enclosure> o vàries paraules clau per a ser cercat. Un element d’un podcast té el següent aspecte:

```xml
<item>
  <title>Socket Wrench Shootout</title>
  <itunes:author>Jane Doe</itunes:author>
  <itunes:subtitle>Comparing socket wrenches is fun</itunes:subtitle>
  <itunes:summary>This week we talk about metric vs. old english socket wrenches. Which one is better? Do you really need both? Get all of your answers here.</itunes:summary>
  <enclosure
    url="http://example.com/podcasts/everything/AllAboutEverythingEpisode2.mp3"
    length="5650889"
    type="audio/mpeg" />
  <guid>http://example.com/podcasts/archive/aae20050608.mp3</guid>
  <pubDate>Wed, 8 Jun 2005 19:00:00 GMT</pubDate>
  <itunes:duration>4:34</itunes:duration>
  <itunes:keywords>metric, socket, wrenches, tool</itunes:keywords>
</item>
```

• Atom 1.0. Atom és un format de sindicació sense relació amb cap de les famílies RSS. Va ser creat amb la intenció de corregir molts dels defectes i indefinicions que es troben a RSS 2.0. Entre d’altres aspectes, Atom estableix dos elements per al contingut: un per a la descripció abreujada i un altre per al contingut sencer; disposa d’un atribut per establir com s’ha d’interpretar el contingut (text, html, xhtml, xml o base64); el format de les dates ha de ser RFC 3339 (a RSS no s’estableix clarament quin és el format vàlid); permet especificar més d’un fitxer de recurs multimèdia en una entrada, i fa obligatori l’ús d’un identificador per a cada entrada (a RSS és opcional).

Els feeds Atom tenen el següent aspecte:

```xml
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title>Example Feed</title>
  <subtitle>A subtitle.</subtitle>
  <link href="http://example.org/"/>
  <updated>2003-12-13T8:30:02Z</updated>
  <author>
    <name>John Doe</name>
    <email>johndoe@example.com</email>
  </author>
  <id>urn:uuid:60a76c80-d399-11d9-b91c-0003939e0af6</id>
```
Com es pot veure, l'estructura del document és molt semblant als formats RSS. Els camps de la cappalera proporcionen informació de la web original, nom de l'autor, data de creació etc. Igualment, les entrades disposen dels camps habituals: títol, enllaç i data de creació. Quant a contingut, es disposa de dos camps: un resum breu (<summary>) i un contingut complet (<content>). Els elements com ara <title>, <summary> i <content> permeten especificar el format de les dades mitjançant l'atribut type.

Un dels aspectes més crítics d'Atom és que, tot i especificar el format dels seus feeds sense les ambigüïtats de la família RSS, realment no aporta res de nou que no es pogués implementar amb RSS mitjançant extensions adicionals. Per tant, la creació d'un nou format aporta més confusió i no ajuda realment a solucionar el problema. Recentment, Atom ha estat proposat com a estàndard per l'IETF. Es pot consultar l'especificació completa en el seu RFC²⁷.

Des d'un punt de vista tecnològic, els quatre formats són pràcticament equivalents i no proporcionen cap avantatge clar sobre els altres. Mentre que RSS 1.0 i Atom estableixen els formats i els continguts sense les ambigüïtats que RSS 0.91 i RSS 2.0, l'ús més extès d'aquest darrer sembla restar-li importància a aquest defeccte de la seva especificació. D'entre tots els formats de sindicació, actualment el més emprat és RSS 2.0. Molts llocs web proporcionen feeds simultàniament en formats RSS 1.0 / 2.0, en canvi Atom no gaudeix de massa popularitat entre els llocs web, tot i que disposa del recolzament d'importantes empreses, com ara Google, que el fa servir al seu servei gratuït de blogs Blogger.

11.2. Aplicacions existents per a l’agregació de feeds

11.2.1. Aplicacions d’escriptori

Els següents agregadors gràtuits de canals es fan servir en l’escriptori Gnome:

- **Liferea**\(^{58}\): una aplicació Gnome que suporta els formats RSS/RDF, a més de CDF, Atom, OCS, i OPML. S’ha desenvolupat sobre les biblioteques C de l’escriptori. Liferea agrupa els canals en jerarquies de carpetes, i proporciona una vista amb tres subfinestres per consultar els continguts:

- **RSSOwl**\(^{59}\): una sofisticada aplicació Java multiplataforma que fa servir els widgets GTK+. Fa servir un sistema de navegació pels feeds similar al de Liferea, però disposa de més opcions, com ara cercar text a tots els feeds, exportar una categoria o grup a PDF o HTML, importació de feeds des de fitxers OPML o descobriment automàtic de feeds a partir d’una URL HTTP:

- **Blam!**\(^{60}\): una aplicació per al Gnome que fa servir Mono .NET i els bindings GTK#. A diferència de les aplicacions anteriors, fa servir una llista plana de canals en comptes d’agrupar-los per categories. La interfície gràfica fa servir una típica organització en tres subfinestres: llista de canals, elements d’un canal i contingut de l’element.

---

58 http://liferea.sourceforge.net/
59 http://www.rssowl.org
60 http://developer.imendio.com/projects/blam
- **Firefox.** Una de les característiques més interessants d'aquest navegador gratuït són les nombroses extensions disponibles, que permeten millorar les seves funcionalitats de manera senzilla. Una de les extensions més conegudes és **Sage**\(^1\), que converteix Firefox en un agregador de canals. **Sage** permet escollir diferents estils per representar el contingut dels canals, i gràcies al sistema de notificació d'actualitzacions es pot obtenir la versió més recent amb senzillesa.

- **Thunderbird.** El popular lector de correu també permet visualitzar feeds **RSS/RDF**. L'usuari crea nous “comptes” **RSS**, de la mateixa manera que es creen comptes **IMAP** o **POP3**. Una vegada hagi afegit els feeds que li interessin, només ha de fer clic en el botó de descarregar missatges: els feeds apareixeran en la finestra de missatges, com si es tractessin de correus electrònics. En visualitzar el contingut d’un feed, apareixeran tots els elements del canal a al finestra de missatges.

### 11.2.2. Serveis on-line

Els portals web més populars solen integrar un agregador de canals per als seus usuaris com a un servei personalitzat més. Aquestes aplicacions proporcionen una interfície web per configurar els feeds i visualitzar el contingut mitjançant un navegador. Els més destacats són els següents:

- **Bloglines**\(^2\). Bloglines es un servei web gratuït d’agregació de canals **RSS/RDF**. Permet importar i exportar canals mitjançant fitxers **OPML**. Els canals subscrits s’actualitzen periòdicament per assegurar que es disposa de la informació més recent.

---

\(^1\) [http://sage.mozdev.org/](http://sage.mozdev.org/)

\(^2\) [http://www.bloglines.com](http://www.bloglines.com)
Google Homepage. Des de la pàgina principal d’un compte de Google, els usuaris poden configurar els feeds que consideren més interessants. També, Google disposa d’una aplicació web semblant a Bloglines, Google Reader⁶³, que fa servir la tecnologia AJAX emprada amb tant d’èxit al seu servei de correu GMail.

MyYahoo! El portal ofereix als usuaris la possibilitat de subscriure’s a canals dintre de la seva pàgina personal.

La majoria de serveis grats de creació de blogs permeten que els usuaris mostren els feeds d’altres webs. D’aquesta manera els bloggers poden crear comunitats i oferir informació que considerin interessant per als seus lectors.

Sovint, les webs solen incloure botons per afegir ràpidament un feed a un d’aquests serveis. Aquests botons redirigeixen a la URL del servei i presenten una pàgina per configurar les propietats del canal seleccionat.

11.3. Agregació de canals i navegació pels continguts amb Saleena

En el present projecte s’ha desenvolupat un agregador de canals RSS per a l’escriptori Gnome, anomenat Saleena, que fa servir les biblioteques C emprades més comunament en l’escriptori. Saleena fa servir una aproximació als lectors de feeds que combina les aplicacions d’escriptori i els serveis on-line. La interfície s’assembleja força a un navegador web amb pestanyes, on cada pestanya és una pàgina HTML generada a partir dels continguts dels feeds descarregats localment. Els passos habituals per visualitzar els feeds amb Saleena són els següents:

1. Creació de categories. L’usuari ha de crear categories per agrupar temàticament els canals. A banda de visualitzar els canals individualment, l’aplicació mostrarà un resum general dels continguts de cada categoria, que permetrà als usuaris fer-se una idea ràpida dels continguts de cada canal. Opcionalment es pot assignar una icona a una categoria, que es mostrarà a les pàgines i a les pestanyes de la interfície.

Seguint el consell de simplicitat de les Gnome Human Interface Guidelines, el Saleena no permet crear subcategories dintre d’una categoria. Aquesta decisió de disseny es va prendre per simplificar la forma de navegar pels continguts emprada per altres aplicacions. Quan es vol navegar per un canal, disposar de múltiples subcategories dificulta trobar-lo amb rapidesa. En fer

⁶³ http://www.google.com/reader
servir només un nivell de categories, els usuaris poden fer servir definicions més generals per a les seves categories i trobar els canals amb facilitat. En aquest aspecte, les categories a Saleena tenen una certa semblança amb els sistemes de "tags" que s'han popularitzat recentment.

2. **Assignació de canals.** Cada categoria pot contenir múltiples canals, referenciats per l'URI del d'on es descarregaran. Saleena fa servir GnomeVFS per obtenir els feeds, de manera que es poden descarregar des de de molts tipus d'ubicacions, tant locals com HTTP, FTP o SMB.

Un dels aspectes que diferencia la navegació pels feeds amb Saleena és l'ús d'una categoria especial denominada "Principal". Aquesta categoria és mantinguda per l'aplicació, i el contingut dels seus canals es mostra quan es crea una pestanya nova. La categoria "Principal" és el punt d'inici de la navegació pels continguts, i permet mostrar de bon començament aquells canals que l'usuari considera més interessants. Per assignar un canal a la categoria Principal, s'ha de marcar la casella "Afegir el canal a la categoria Principal" en el diàleg de configuració dels canals. En desplegar la categoria "Principal" a l'editor de categories, es poden visualitzar els canals assignats a aquesta categoria.

3. Descàrrega dels continguts. Per raons d'eficiència en l'accés als continguts, els canals s'han de descarregar localment al disc de l'usuari. Mitjançant el diàleg "Descàrrega del contingut", l'aplicació permet descarregar un canal individual, tots els canals d'una categoria o tots els canals existents. Aquest diàleg manté informat l'usuari de l'estat de les descàrregues i permet interrompre i/o cancel·lar la descàrrega dels canals.

En el cas que es produeixin errors durant la descàrrega, l'usuari pot visualitzar un registre de les descàrregues per esbrinar el motiu de l'error (per exemple, una URL incorrecta).

Una vegada es descarreguen els feeds, l'usuari accedeix els canals mitjançant la finestra principal. En certs aspectes, Saleena es comporta com un navegador web ja que permet obrir pestanyes individuals per
navegar pels continguts, i avançar i retrocedir per l’historial de navegació. A l’igual que els serveis on-line de lectura de feeds, l’aplicació construeix dinàmicament una interfeïce HTML basant-se en les categories que ha creat l’usuari i el contingut dels canals.

Quant a la navegació pels continguts, Saleena fa servir una organització en tres nivells: pàgina principal, categories i canals d’una categoria:

1. **Pàgina principal.** Quan l’usuari obre una nova pestanya, es mostra per defecte el contingut de la categoria “Principal”. En aquesta pàgina es mostra un breu resum de les notícies dels canals assignats en aquesta categoria principal, així com la llista de categories definides per l’usuari. Des d’aquesta pàgina l’usuari pot accedir ràpidament als canals que considera més importants.

![Pàgina principal de Saleena](image)

En fer clic en un enllaç d’una categoria o d’un canal, l’aplicació en mostrarà el contingut. En el cas que es faci clic amb el botó central del ratolí, el contingut es mostrarà en una nova pestanya. Tots els enllaços a recursos externs (com ara enllaços HTTP o adreces de correu electrònic) s’obriran externament amb l’aplicació predeterminada de l’escriptori.

2. **Categories.** En fer clic en l’enllaç d’una categoria, es mostra un resum individual de cadascun dels seus canals. Cada resum mostra les tres notícies més recents i la resta de titulars anteriors. Gràcies a aquest petit resum, l’usuari pot decidir si vol llegint el contingut sencer del canal; moltes vegades aquest resum pot resultar suficient.
3. **Canals.** En accedir el contingut d’una categoria, l’usuari pot visualitzar el contingut complet dels canals mitjançant els enllaços sota “Canals”, o bé des del cos del resum el canal, clicant “Vés al canal ...”. Una nova pàgina mostra totes les entrades dels canals, amb l’enllaç original a la web:

Saleena reconeix els canals que contenen metainformació referida a podcasts, i permet obrir els fitxers multimèdia relacionats amb l’aplicació predeterminada de l’entorn:
11.4. Comparativa entre aplicacions existents i Saleena

- **Aplicacions d’escriptori.** En comparació amb les altres aplicacions, el disseny de la interfeix del Saleena és més proper als requeriments de simplicitat de les Gnome Human Interface Guidelines. Mentre les altres aplicacions fan servir una vista amb tres subfinestres (arbre de feeds, llista de continguts del canal i visualització del contingut), el funcionament de la nostra aplicació s’assembla a un navegador web amb pestanyes, de manera que els usuaris poden familiaritzar-se-hi més ràpidament. La navegació mitjançant pàgines HTML permet consultar els continguts intuitivament, i gràcies als resums de tots els canals d’una categoria, els usuaris poden estalviar temps a l’hora de revisar les noves notícies.

Des del punt de vista de la base tecnològica, **Saleena** i **Liferea** fan servir les biblioteques estàndard de l’escriptori **Gnome**, mentre que **RSSOwl** requereix instal·lar l’entorn **Java**, i **Blam!** el **Mono .NET**. Per tant, fer servir la nostra aplicació requereix menys recursos d’espai a disc, ja que totes les seves biblioteques es trobaran preinstall·lades. En el cas de l’entorn **Mono .NET**, el seu ús és força popular (i des de la versió Gnome 2.16 ha esdevingut una dependència de l’escriptori), mentre que **Java** és rarament emprat en aplicacions de l’entorn.

De totes les aplicacions per a la lectura de feeds, sense dubte **RSSOwl** és la més completa de totes. Aquest agregador disposa de funcionalitats molt sofisticades que no estan suportades en **Saleena**, com ara detecció automàtica dels feeds disponibles sota una web, importació/exportació de llistats de feeds mitjançant fitxers en format OPML i cerques als continguts dels canals descarregats. Tanmateix, tot i la seva utilitat no es tracta de funcionalitats emprades gaire habitualment.

En comparació amb les extensions per a **Firefox** i **Thunderbird**, la integració en una aplicació existent permet que l’usuari no hagi de fer servir una aplicació addicional per llegir feeds. A més, ambdues aplicacions disposen d’una gran comunitat d’usuaris i desenvolupadors que asseguren la qualitat de les futures versions, i es troben disponibles en múltiples plataformes.

- **Serveis on-line.** **Saleena** fa servir un sistema de navegació semblant al dels serveis on-line, ja que ambdós fan servir una interfeix HTML per mostrar els continguts dels canals. Un dels avantatges de la nostra aplicació és que permet llegir els canals off-line, ja que es descarreguen localment en la carpeta de l’usuari, i d’aquesta manera s’estalvia ample de banda cada vegada que es vol consultar una nova pàgina. En canvi, els serveis on-line requereixen tenir una connexió per consultar els continguts, amb la corresponent ocupació de l’ample de banda de l’usuari.

Un dels principals avantatges dels serveis on-line és la mobilitat: poder llegir els continguts dels feeds des que qualsevol indret, ja que només cal un navegador per accedir aquestes aplicacions. Això també suposa no haver d’instal·lar aplicacions addicionals en el sistema, amb el conseguent estalvi d’espai i de recursos. Molts d’aquests serveis formen part de l’infraestructura d’un portal, i això permet proporcionar als usuaris una experiència més senzilla a l’hora de realitzar certes tasques, com ara enviar per correu el contingut d’unanova notícia, o afegir un nou feed al nostre compte amb un sol clic.
12. Disseny de l’aplicació

"Let me tell you about my boat"

The Life Aquatic with Steve Zissou (2004)

12.1. Ús del patró arquitectònic en tres capes

Per dissenyar l’aplicació s’ha emprat un patró arquitectònic tipic de tres capes. Aquest patró és emprat comunament en els sistemes d’informació que han de tractar amb dades de formats diversos i proporcionar-ne diferents mètodes de visualització. En el cas de la nostra aplicació, gràcies a aquest patró arquitectònic s’aconsegueix una gran flexibilitat a l’hora d’haver de treballar amb feeds RSS de diferents formats. Per exemple, mentre la capa de dades abstraiu la sintaxi dels diferents formats, la capa de domini fa servir un vocabulari XML propi per definir el contingut de les pàgines de feeds i la capa de presentació ho visualitza en format HTML.

El disseny intern de cada capa s’ha realitzat seguint el patró d’orientació a objectes. Tot i que l’aplicació s’hagi escrit en el llenguatge C, això és tècnicament possible gràcies a que les classes s’han implementat fent servir la tecnologia GObject. Tant GTK+ com libglade fan servir classes GObject, la qual cosa ens permet disposar d’un disseny orientat a objectes tot i que el llenguatge no disposa d’aquest paradigma. També, mitjançant les classes GtkTreeModel i GtkTreeView de GTK+ s’ha pogut emprar en el disseny el patró model-vista-controlador. En el nostre cas, l’aplicació fa servir el component model GtkTreeStore per definir les relacions entre canals i categories de manera senzilla, a la vegada que es disposen de vàries visualitzacions diferents de les mateixes dades.

Per simplicitat, les següents seccions descriuen el contingut de cadascuna de les capes. Quan s’escaigui, es mostrarà la relació amb les classes de capes veïnes.

12.2. Capa de presentació

En aquesta capa es troben els diàlegs de l’aplicació, implementats mitjançant widgets GTK+. El seu disseny és força senzilla, ja que la majoria de classes es corresponen amb finestres de diàleg.
12.2.1. Disseny UML de la capa de presentació

Restriccions d’integritat:

- El conjunt d’instàncies de SaleenaTabTitleGUI de l’associació “Té” entre SaleenaTabsGUI i SaleenaTabTitleGUI coincideix amb el conjunt format per les instàncies de SaleenaTabTitleGUI de cada instància de SaleenaTabBrowserGUI que forma part de l’associació “mostra” entre SaleenaTabBrowserGUI i SaleenaTabsGUI.

12.2.2. Descripció de les classes de la capa de presentació

- SaleenaGUI. Aquesta classe representa la finestra principal de l’aplicació, des de la qual els usuaris realitzen la majoria d’accions possibles. SaleenaGUI disposa també de la majoria dels diàlegs emprats per l’aplicació, com ara l’editor de categories (CategoryEditorGUI) i l’editor de Preferències (PreferencesGUI):

  - UpdateContentGUI. Aquesta classe implementa el diàleg de descàrrega dels feeds. Mitjançant la seva finestra, l’usuari reb notificació del percentatge de transferències completat, i permet interrompre l’acció actual o cancel·lar el procediment. La classe UpdateContent de la capa de domini fa servir aquesta classe per interactuar amb l’usuari.

  - UpdateContentErrorLogGUI. Aquesta classe mostra un quadre de diàleg amb els errors que s’han produït durant la descàrrega. La llista de canals que han fallat és un model de la classe GtkTreeModelFilter derivat de CategoryModel, aprofitant així les facilitats que proporciona el patró model-vista-controlador de GTK+ per poder disposar de diferents submodels fills a partir d’un model pare.

En el cas que no existeixi cap error a mostrar, el diàleg simplement informa l’usuari que tot ha anat correctament. Això és més intuitiu que mostrar una llista d’errors buida, i segueix més d’aprop els consells de les Gnome HIG.

- PreferencesGUI. Aquesta classe implementa el diàleg de preferències de l’aplicació. Per obtenir els valors de la configuració fa servir un objecte PreferencesData, que interacciona amb el sistema de configuració GConf.
Visualitzador i organitzador de canals RSS/RDF per a l'escritori Gnome

- **CategoryEditorGUI.** Aquesta classe implementa l'editor de categories. Mitjançant la seva interfície es poden crear categories, assignar-hi canals i editar-ne les propietats. Aquesta classe actua com a controlador del model **CategoryModel**, a més de visualitzar-ne el contingut mitjançant una vista **GtkWidget**. Addicionalment fa servir els següents diàlegs:

  - **CategoryPropertiesGUI.** Una senzilla interfície per afegir / modificar categories. Permet establir el nom i la icona.
  
  - **ChannelPropertiesGUI.** Una senzilla interfície per afegir / modificar canals. Permet establir el nom del canal, la URL del feed i si s'ha de mostrar a la pàgina Principal.
  
  - **CategorySelectorGUI.** Aquesta classe representa una finestra de selecció de categories. Per llistar les categories disponibles es fa servir un objecte **GtkWidget** FilteredModel, que proporciona un model filtrat (només les categories) sobre l'objecte **CategoryModel**.

- **SaleenaTabsGUI.** Aquesta classe, derivada de **GtkWidget**, proporciona a **SaleenaGUI** les pestanyes per visualitzar els navegadors de contingut. Cada pestanya creada conté un objecte de la classe **SaleenaTabBrowserGUI**, i la banda seleccionable fa servir un objecte **SaleenaTabTitleGUI**, que mostra la icona de la categoria actual i la descripció.

  - **SaleenaTabBrowserGUI.** Aquesta classe, derivada de **GtkWidget**, permet visualitzar el contingut dels feeds mitjançant una pàgina web. Els continguts d'aquesta pàgina són generats per la classe **SaleenaHTMLContent** de la capa del domini.

- **SaleenaTabTitleGUI.** Una senzilla classe que agrupa els elements que es visualitzen en el selector de pestanyes: una icona (**GtkImage**), una etiqueta de text (**GtkLabel**) i un botó per tancar la pestanya (**GtkButton**). Quan es carrega un nou contingut, **SaleenaTabBrowserGUI** n'actualitza els widgets.

Un dels problemes de disseny que presenta la navegació mitjançant pestanyes són els canvis d'estat que s'han de produir a la interfície gràfica cada vegada que es mou a una pestanya diferent.

L'aplicació ha d'adaptar-se dinàmicament al contingut de la pestanya activa; això inclou modificacions senzilles, com ara canviar el títol de la finestra, però d'altres més complexes, com canviar el contingut dels botons de la barra d'eines de l'historial de navegació. Aquests botons permeten anar endavant i endarrere, i a més mostren un submenú amb l'historial d'ubicacions visitats.

**SaleenaTabBrowserGUI** disposa d'un objecte de la classe **NavigationHistory** de la capa domini per gestionar l'historial de navegació. Aquest objecte permet obtenir un menú **NavigationHistoryMenuGUI** amb les ubicacions per les que s'ha navegat. En canviar de pestanya, l'objecte **SaleenaTabBrowserGUI** actua modificant els botons **NavigationHistoryMenuGUI** amb els seus menús.

---

64 De fet, les Gnome HIG desaconseillen dissenyar les interfície de les aplicacions fent servir pestanyes. En el cas que es vulgui treballar amb múltiples documents, es recomana fer servir finestres individuals per a cadascun. Un dels motivs que fan desaconsellable l'ús és la confusió entre les accions que s'apliquen sobre totes les pestanyes i les que només s'apliquen a una. Per això, en dissenyar interfeïncies d'aquest tipus, és important informar l'usuari de l'abast de les aplicacions que s'apliquen sobre el contingut de les pestanyes.
- **NavigationHistoryMenuGUI.** Aquesta classe, derivada de `GtkMenu`, presenta un menú amb les entrades de l'històrial de navegació, que es mostrarà quan l'usuari faci clic en el botó de menú de l'objecte `NavigationHistoryButtonGUI`. En seleccionar una entrada, s'emet un signal "go-to"; la signatura del signal inclou l'element `CategoryElement` o `ChannelElement` a on navegar:

```c
void go_to (GtkWidget* wg,
            GObject* element,
            gpointer user_data)
```

La classe `NavigationHistory` en modifica el contingut segons les accions de navegació de l'usuari.

- **NavigationHistoryButtonGUI.** GTK+ disposa de la classe d'objectes `GtkMenuToolButton`, que implementa els botons amb submenú de les barres d'unes. La classe `NavigationHistoryButtonGUI` n'estic el funcionament; fa servir menús de la classe `NavigationHistoryMenuGUI` per mostrar l'històrial de navegació, i cada vegada que se selecciona una entrada del menú (o es prem el botó per anar endavant/endarrere) emet un signal amb l'objecte `CategoryElement`/`ChannelElement` a on navegar.

### 12.3. Capa de domini

A la capa de domini es troba un dels objectes més importants de l'aplicació: `CategoryModel`, el model de dades en forma d'arbre que conté les categories i els canals. Un altre de les classes més importants és `SaleenaHTMLContent`, que s'encarrega de construir la descripció HTML de canals i categories per poder visualitzar els seus continguts.

#### 12.3.1. Disseny UML de la capa de domini
Restriccions d’integritat:

- Existeix una única instància de **CategoryModel**.

### 12.3.2. Descripció de les classes de la capa de domini

- **CategoryModel.** Aquesta classe és filla de **GGtkTreeStore**, el component model de GTK+ per al patró model-vista-controlador. **CategoryModel** fa servir el model d’arbre per emmagatzemar les dades de les categories i els canals. Així, les files que es troben a profunditat 0 són les categories, i les files filles, els seus canals. A banda de les categories creades per l’usuari, **CategoryModel** conté una categoria derivada especial: Principal. Aquesta categoria està formada per aquells canals que s’han de visualitzar a la pàgina principal. En activar aquesta propietat, el canal s’afegeix sota la categoria Principal, i en desactivar-la s’elimina. Per tant, aquesta categoria derivada és mantinguda internament per **CategoryModel**.

Mitjançant **CategoryModel** és realitzen totes les operacions relacionades amb categories i canals; per això, es pot considerar que aquest objecte també implementa el patró façana. Aquesta classe inclou les operacions per crear canals i categories, canviar-ne les propietats, moure/copiar canals entre categories i obtenir una llista de totes les categories.

El model fa servir les següents columnes de dades:

<table>
<thead>
<tr>
<th>Identificador</th>
<th>Tipus</th>
<th>Descripció</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATEGORY_PIXBUF_COLUMN</td>
<td>GdkPixbuf</td>
<td>Icona renderitzada de la categoria</td>
</tr>
<tr>
<td>CATEGORY_NAME_COLUMN</td>
<td>gchar*</td>
<td>Nom de la categoria</td>
</tr>
<tr>
<td>CATEGORY_ICON_COLUMN</td>
<td>gchar*</td>
<td>Camí de la icona de la categoria</td>
</tr>
<tr>
<td>CHANNEL_CHECKED_COLUMN</td>
<td>gboolean</td>
<td>Si s’ha de mostrar a Principal</td>
</tr>
<tr>
<td>CHANNEL_NAME_COLUMN</td>
<td>gchar*</td>
<td>Nom del canal</td>
</tr>
<tr>
<td>CHANNEL_URL_COLUMN</td>
<td>gchar*</td>
<td>URL d’on obtenir el canal</td>
</tr>
<tr>
<td>CHANNEL_CACHED_COLUMN</td>
<td>gchar*</td>
<td>Camí al feed descarregat localment</td>
</tr>
<tr>
<td>CHANNEL_DOWNLOAD_ERROR_VALUE_COLUMN</td>
<td>gboolean</td>
<td>Si s’ha produït un error en descarregar el feed</td>
</tr>
<tr>
<td>CHANNEL_DOWNLOAD_ERROR_MSG_COLUMN</td>
<td>gchar*</td>
<td>Una cadena que descriu el motiu de l’error de la descàrrega del feed</td>
</tr>
</tbody>
</table>

Una de les característiques més sofisticades de **GtkTreeModel** és la creació de models derivats mitjançant la classe **GtkTreeModelFilter.** Aquesta classe crea un model virtual, que filtra les dades d’un model pare, de la mateixa manera que els SGBD proporcionen vistes sobre les dades d’una taula. **CategoryModel** proporciona dos submodels filtrats **GtkTreeModelFilter**, emprats per diàlegs de la capa de presentació:

- Un model que mostra només les categories creades per l’usuari (excepte Principal), emprat pel diàleg **CategorySelectorGUI.** Fa servir una funció de filtratge per decidir si la fila que ha
de mostrar pertany a una categoria o a un canal.

- Un model emprat pel diàleg `UpdateContentErrorDialogGUI`. Aquest model mostra els canals i categories on s’han produït errors durant la descàrrega. El model filtrat mostra només les files que tenen el valor `TRUE` a la columna `CHANNEL_DOWNLOAD_ERROR_VALUE_COLUMN`.

Com hem vist en parlar dels widgets GTK+, la classe `GtkTreeModel` fa servir estructures `GtkTreePath` i `GtkTreeIter` per obtenir i modificar el contingut de les files del model. Atès que treballar directament amb aquests objectes és poc pràctic, `CategoryModel` fa servir les classes `CategoryElement` i `ChannelElement`, que proporcionen un millor nivell d’abstracció, a més d’un disseny de les operacions millor orientat als requeriments del model:

  - **CategoryElement**. Representa una categoria existent en l’objecte `CategoryModel`. Internament, conté un objecte `GtkTreeRowReference`; encara que la fila que representa la categoria és mogui de posició, `CategoryElement` seguirà apuntant a la fila correcta. Mitjançant aquest objecte, es poden obtenir els atributs d’una categoria i una llista amb els seus canals (`ChannelElement`).

  - **ChannelElement**. Representa un canal existent en l’objecte `CategoryModel`. Internament, conté un `GtkTreeRowReference`; encara que la fila que representa el canal es mogui de posició, `ChannelElement` seguirà apuntant a la fila correcta. Mitjançant aquest objecte, es pot obtenir els atributs d’un canal, esbrinar la categoria a la qual pertany i obtenir un objecte `ChannelElementData` per llegir elements del feed RSS/RDF.

- **UpdateContent**. Aquesta classe controla el procés de descàrrega dels canals subscriïts. Permet descarregar un sol canal, tots els canals d’una categoria o tots els canals de totes les categories. Mitjançant el diàleg `UpdateContentGUI`, l’usuari ordena `UpdateContent` que cancel·li/atauri la descàrrega. Les operacions finals d’entrada sortida, són realitzades per un objecte de la classe `CacheManager`.

Com la descàrrega dels canals s’ha de realitzar asincronament, `UpdateContent` crea el seu propi flux `GThread` per no haver de bloquejar el bucle principal de GTK+ (la qual cosa provocaria que l’aplicació deixés de respondre fins que la descàrrega finalitzés. La sincronització entre fluxos es realiza mitjançant una cua de missatges asincrona, `GAsyncQueue`, compartida amb la classe `UpdateContentGUI`.

- **CacheManager**. Aquesta classe permet controlar les operacions d’entrada/sortida necessàries per descarregar els feeds des de l’URI d’un `ChannelElement`. Internament, fa servir un objecte de la classe `CacheManagerData`, que realitzarà les operacions d’entrada/sortida asincrones. Si la descàrrega finalitza correctament, s’estableix en la columna `CHANNEL_CACHED_COLUMN` de `CategoryModel` el camí on s’ha descarregat localment el feed.

- **NavigationHistory**. Mitjançant aquesta classe, cada objecte `SaleenaTabBrowserGUI` manté el seu historial de navegació. Aquesta classe conté una llista amb les ubicacions de navegació, a més de la la ubicació actual. Internament, el contingut de les llistes està vinculat un menú de la classe `NavigationHistoryMenuGUI`. 
Atès que la navegació pels continguts es realitza mitjançant pàgines web, és necessari incorporar el concepte d'URL per als canals i categories. Per a la nostra aplicació, s'ha definit una URL especial, saleena://, mitjançant la qual es pot referenciar un canal o una categoria pel seu nom. Això permet, per exemple, que en fer clic en un enllaç HTML que conté la URL d'una categoria de Saleena, es generi el contingut de la categoria en una nova pàgina.

La classe **SaleenaNavigation** s'encarrega d'interpretar les URL per poder generar els continguts corresponents, proporcionant així l' experiència de navegació pels continguts de canals i categories:

- **SaleenaNavigation.** Aquesta classe té mètodes per analitzar el tipus d'URL i realitzar una acció específica. Si el tipus és http://, ftp:// o mailto:, l'acció a realitzar serà obrir l'aplicació predeterminada per gestionar aquestes URLs. En canvi, si el prefix és saleena://, genera el contingut HTML de la categoria o canal que apareix en aquest enllaç i el mostra en un objecte **SaleenaTabBrowserGUI.** D'aquesta manera s'aconsegueix navegar per continguts com si fos una pàgina web qualsevol.

Hi ha d'altres URL interpretades internament per **SaleenaNavigation.** Per exemple, les URL amb prefix saleena-podcast:// indiquen que s'inicia l'aplicació predeterminada per obrir un fitxer multimèdia que es troba en la URL referenciada.

- **SaleenaHTMLContent.** Aquesta classe és l'encarregada de crear el contingut HTML navegable de l'aplicació. Segons el tipus de contingut a visualitzar (categoria o canal) es crea un objecte que implementi la interfície **SaleenaHTMLContentInterface** per obtenir una descripció XML del contingut. La generació final del codi HTML es realitza mitjançant l'aplicació d'un full d'estil XSLT. Finalment, **SaleenaHTMLContent** escriu el contingut de la pàgina web dintre de l'objecte **SaleenaTabBrowserGUI.** Per evitar que l'aplicació es pugi quedar bloquejada durant la generació del contingut, tot aquest procés es realitza en un **GThread** secundari.

- **SaleenaHTMLContentInterface.** Aquesta interfície disposa d'un sol mètode, que retorna la descripció XML del contingut relacionat amb una categoria o un canal. És implementada per les següents classes:
  - **SaleenaMainHTMLContent.** Genera el contingut XML de la categoria Principal. Aquesta descripció inclou la capçalera de la pàgina, la llista de canals de Principal, una llista de categories i els 5 titulars més recents de cada canal. Per obtenir el contingut d'un feed, fa servir objectes de la classe **ChannelElementData.**
  - **SaleenaCategoryHTMLContent.** Genera el contingut XML d'una categoria, excepte Principal. Aquesta descripció inclou la capçalera de la pàgina, la llista de canals d'una categoria, la llista de totes les categories i un resum de cada canal, que mostra les tres notícies més recents del canal i els titulars de la resta. Per obtenir el contingut d'un feed, fa servir objectes de la classe **ChannelElementData.**
  - **SaleenaChannelHTMLContent.** Genera el contingut XML d'un canal. Aquesta descripció inclou la capçalera de la pàgina, la llista de canals de la seva mateixa categoria, la llista de totes les categories i el contingut de totes les entrades del canal. En el cas que el canal contingui referències a fitxers multimèdia, es generarà també el contingut que descrigui aquests
12.4. Capa de dades

En aquesta capa es troben les classes que proporcionen accés a dades emmagatzemades en diferents sistemes: claus de configuració (GConf), fitxers XML (libxml2) i servidors web (GnomeVFS).

12.4.1. Disseny UML de la capa de dades

![Diagrama UML](image)

12.4.2. Descripció de les classes de la capa de dades

- **PreferencesData.** Aquesta classe proporciona els mètodes per escriure i llegir claus de configuració GConf.

- **CategoryModelData.** Aquesta classe importa / exporta el contingut d’un objecte CategoryModel a un fitxer XML. Fa servir un vocabulari propi, i alguns elements del vocabulari RDF.

- **CacheManagerData.** Aquesta classe realitza les operacions d’entrada/sortida per obtenir un feed d’una URL. Tecnològicament, fa servir les operacions de transferència de fitxers de GnomeVFS per obtenir el feed remot i copiar-lo en un directori de l’usuari.

Tot i existir una API GnomeVFS asincròna per realitzar aquestes operacions asincrònimament, existeixen certs problems a l’hora de cancel·lar efectivament una transferència en curs. Per això s’ha obat per crear un flux GThread propi. La sincronització amb el flux de la classe CacheManager es realitza mitjançant una cua de missatges GAsyncQueue, que permet a CacheManager interrompre o cancel·lar la transferència, a petició de l’usuari.

- **ChannelElementData.** Aquesta classe proporciona mètodes abstractes i estructures de dades per iterar per les entrades d’un feed, independentment del format que faci servir. El disseny és semblant al d’un iterator, amb mètodes per obrir el feed, obtenir elements d’un en un i consultar si s’ha arribat al final. ChannelElementData disposa d’estructures de dades que permeten treballar amb feeds de diferents formats de manera unificada: ChannelElementDataInfo (que representa les metadades d’un feed) i ChannelElementDataItem (que representa un element d’un feed).
- **ChannelElementDataRSS**. Aquesta classe implementa els mètodes abstractes de la classe **ChannelElementData**, i permet obtenir el contingut de feeds RSS 0.91, RSS 0.92, RSS 0.93 i RSS 2.0.

- **ChannelElementDataRDF**. Aquesta classe implementa els mètodes abstractes de la classe **ChannelElementData**, i permet obtenir el contingut de feeds RSS 1.0.

### 12.5. Aplicació del patró model-vista-controlador de GTK+

Com es va veure en parlar de les classes de GTK+, aquesta biblioteca proporciona un sofisticat component per poder emprar el patró model-vista-controlador. En el següent diagrama UML mostra el disseny de les classes GTK+ d’aquest patró:

- **GtkTreeModel**. Una interfície que defineix mètodes, signals i estructures de dades compartits per tots els tipus de models.

- **GtkTreeStore**. Un **GtkTreeModel** que emmagatzema la informació en forma de files i columnes, on cada fila pot tenir un pare. De manera semblant, **GtkListStore**, emmagatzema les dades en files sense pares.

- **GtkTreeModelFilter**. Un **GObject** que implementa la interfície **GtkTreeModel**. En comptes d’emmagatzemar les dades directament, filtra el contingut d’un altre **GtkTreeModel**. **GtkTreeModelFilter** permet crear models derivats d’un altre sense duplicar dades, un mecanisme semblant a les vistes dels SGBD.

- **GtkTreeView**. Un component per visualitzar objectes **GtkTreeModel**. Escolla els canvis que es produeixen en el model i n’actualitzia la visualització.

- **GtkTreeSelection**. Un objecte que permet seleccionar files d’un **GtkTreeModel** mitjançant un objecte **GtkTreeView**.

Aprofitant la facilitat per integrar aquest patró en un disseny amb tres capes, la nostra aplicació fa servir el component **GtkTreeModel** a la capa de domini i **GtkTreeView** a la capa de presentació. Les classes **CategoryEditorGUI** i **CategorySelectorGUI** es comporten com a controladors del model, a la vegada que en realitzen la visualització:
**CategoryModel.** Una classe derivada de GtkTreeStore que emmagatzema la informació de categories i canals. Disposa de dues operacions per obtenir-ne dos models GtkTreeModelFilter: category_model_get_model_categories_filtered() (mostra només les files que es corresponen amb categories) i category_model_get_model_errors_filtered() (mostra només els canals on s’han produït errors durant la descàrrega dels continguts).

**CategoryEditorGUI.** Una subclasse de GtkWidget que implementa un editor de categories. Es comporta com a controlador de CategoryModel, ja que permet crear noves categories i canals i modificar-ne el contingut. També conté un objecte GtkTreeView per visualitzar les dades del model.

**CategorySelectorGUI.** Una subclasse de GdkDialog que permet seleccionar una categoria creada per l’usuari. Fa servir el model obtingut amb category_model_get_model_categories_filtered() per visualitzar les categories en un GtkTreeView.

**UpdateContentErrorLogGUI.** Una subclasse de GdkDialog que permet visualitzar els canals on s’han produït errors durant la descàrrega dels feeds. Fa servir el model obtingut amb category_model_get_model_errors_filtered() per visualitzar-lo en un GtkTreeView.

**CategoryModelData.** Aquesta classe permet importar i exportar el contingut d’un CategoryModel en un fitxer XML.

12.6. Integració de GladeXML en la construcció d’interfícies gràfiques

La construcció d’una interfície gràfica sol involucrar dues etapes: la definició d’una nova classe que estenguï el comportament d’una classe finestra, i l’addició posterior d’elements (contenidors, botons, menús, etc.). En GTK+, la creació d’una interfície comença per definir una nova classe que hereta de GtkWindow o GdkDialog, i posteriorment s’hi afegeixen els altres widgets (GtkVBox, GtkButton, GtkMenu...).

Si es fa servir libglade com a assistent per construir les interfícies gràfiques, llavors ja no és possible fer-ho...
d'aquesta manera, ja que GladeXML construeix la interfície en un GtkWidget nou, en comptes de fer-ho sobre l'objecte que estem creant. En aquest cas, el disseny de la nostra classe s'ha de modificar per poder emprar un objecte GladeXML. Es poden fer servir els següents patrons de disseny:

a) Creació d'una classe que no sigui un GtkWidget. La classe MyClass és un objecte GObject que obté el GtkWidget a mostrar mitjançant un objecte GladeXML. Totes les operacions sobre GtkWidget es realitzaran mitjançant mètodes façana de la classe MyClass.

Aquest disseny té l'inconveni que la nostra classe MyClass no és un objecte GtkWidget, per això aquells mètodes de GtkWidget que hàgin de fer servir s'hauran de reimplementar mitjançant mètodes wrappers a MyClass. És el cas del mètode gtk_widget_show(), que mostra el GtkWidget per pantalla: MyClass ha de proporcionar un mètode show() per poder mostrar el seu GtkWidget:

Haver d'implementar mètodes wrapper sobre els mètodes de GtkWidget pot suposar un problema si se n'han de fer servir moltes crides. En canvi, la inicialització del GtkWidget és senzilla, ja que GladeXML el construeix completament.

La classe CategoryEditorGUI fa servir aquest disseny per proporcionar un editor de categories a l'usuari. CategoryEditorGUI és un GObject que conté l'objecte GtkWidget creat per GladeXML, i disposa de mètodes façana que operen sobre el diàleg, per exemple category_editor_gui_show().

b) Creació d'una classe GtkWidget. En aquest cas, la nostra classe MyClass és una finestra GtkWidget, que obté el seu contingut mitjançant l'objecte GladeXML.

GladeXML construeix els widgets interns de la finestra GtkWidget, que s'hi afegeixen amb una
crida al mètode `gtk_container_add()`). En aquest cas, només s'ha de demanar a `libglade` que construeixi el widgets del contingut, no la finestra sencera. És possible que alguns elements i propietats del nostre `GtkDialog` s'hagin de modificar posteriorment, ja que `GladeXML` només retorna una part de la finestra creada:

```
MyClass

<create>

myWindow

<create>

gtk_container_add (myWindow)

gtk_widget_show()
```

Aquest mètode s’assembla més a la construcció habitual d’interfícies gràfiques, i permet que la nostra classe hereti de `GtkDialog`, de manera que es conserva la jerarquia completa de GTK+. Tanmateix la inicialització és més complicada que en el disseny anterior, atès que el `GtkWidget` obtingut de l’objecte `GladeXML` és només una part de la finestra, i pot caldre afegir elements addicionals o canviar atributs del nostre objecte `GtkDialog`.

La classe `PreferencesGUI` fa servir aquest disseny per construir la seva interfície. `PreferencesGUI` és un `GtkDialog` que en comptes de construir tota la interfície, n’obté els widgets (un `GtkNotebook` amb els widgets de configuració) mitjançant l’objecte `GladeXML`. La resta de la interfície s’ha de crear manualment (botons inferiors del diàleg, establir el títol de la finestra, etc.)
13. Planificació

"En este reino adecuentamos ayer y planeamos adecentar mañana, pero nunca adecuentamos hoy".
Alicia en el país de las maravillas, Lewis Carroll

13.1. Investigació

<table>
<thead>
<tr>
<th>Investigació</th>
<th>Feina 1</th>
<th>Feina 2</th>
<th>Feina 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tecnologies sindicació</td>
<td>15d</td>
<td>1d</td>
<td>1d</td>
</tr>
<tr>
<td>RSS, RDF i Atom</td>
<td>3d 4h</td>
<td>1d 4h</td>
<td>1d 4h</td>
</tr>
<tr>
<td>Aplicacions existents</td>
<td>1d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uso per a la sindicació</td>
<td>1d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bibliotques Gnome</td>
<td>10d 4h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLib</td>
<td>1d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GObject</td>
<td>3d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GTK+ i libglade</td>
<td>3d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GlibHTML i GtkMaxEmbed</td>
<td>1d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gnome HIG</td>
<td>4h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GnomeVFS</td>
<td>4h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Libxml2 / libxslt</td>
<td>4h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GCCF</td>
<td>4h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Creació de projectes</td>
<td>1d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVS</td>
<td>4h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automake/autoconf</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- L'etapa d'investigació involucra familiaritzar-se amb les tecnologies que s'hauran d'emprar en l'aplicació. En el cas del Gnome, els dos components més complexos són GObject i GTK+ / libglade. Per a ambdós, és recomenable desenvolupar proves de concepte que ajudin a entendre els aspectes més complicats.

- **Riscos**: tot i que en general la documentació és extensa, en alguns casos les API són massa confoses (libxml2/libxslt, automake/autoconf) o bé inexistentes (GtkHTML3). El període d'investigació pot involucrar la cerca d'altres projectes que facin servir aquests components, com a exemple d'ús.
13.2. Configuració del projecte

<table>
<thead>
<tr>
<th>Nom</th>
<th>Semana 4, 2007</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuració</td>
<td>22 23 24 25 26 27 28</td>
</tr>
<tr>
<td>Repoitoris CVS local</td>
<td>5d</td>
</tr>
<tr>
<td>Creació entorn build</td>
<td>5d</td>
</tr>
<tr>
<td>Instal·lació biblioteques desenvolupament</td>
<td>4h</td>
</tr>
<tr>
<td>Documentació adicional</td>
<td>4h</td>
</tr>
<tr>
<td>Suport per a Internacionalització</td>
<td>4h</td>
</tr>
<tr>
<td>Aplets de suport</td>
<td>4h</td>
</tr>
<tr>
<td>Prove &quot;Hola món&quot;</td>
<td>5d</td>
</tr>
</tbody>
</table>

- Per simplicitat, es contempla la creació del projecte en un repoitoris CVS local. Això simplifica les tasques inicials de manteniment, a l'hora que permet fer proves amb el repoitoris sense haver de preocupar-se per la configuració de l'entorn que allotgi el repoitoris.

- Al final de l'etapa, és recomenable desenvolupar un petit exemple tipus "Hola món!". Aquesta aplicació ha de ser una senzilla prova de concepte, que faci servir qualsevol crida de les API que es faran servir. Això ens assegura que l'entorn de desenvolupament està ben configurat i que es disposa de totes les biblioteques necessàries. També és important comprovar que l'entorn de build i instal·lació s'ha creat correctament.

- A més de les eines habituals per al desenvolupament (gcc, gdb, etc.) és recomenable disposar d'altres aplicacions que ens ajudaran a accelerar la fase de desenvolupament. A l'apèndix A es recull una col·lecció d'aplicacions d'ús comú per al desenvolupament d'aplicacions al Gnome.

- Riscos: la creació d'un entorn de build amb automake/autoconf sol ser difícil al començament. És preferible basar-se en l'entorn d'una altra aplicació Gnome, o bé fer servir el generador de projectes de l'IDE Anjuta.

13.3. Desenvolupament
La tasca més costosa en l’etapa de desenvolupament és la creació del model de categories. Aquest model ha de disposar de moltes operacions de creació i modificació de canals; també implementa les restriccions d’integritat i una categoria derivada, ’Principal’, que ha de ser mantinguda pel propi model. L’exportació del model en fitxer XML fa servir RDF, de manera que cal familiaritzar-se amb les nocions pròpies d’aquest vocabulari de descripció de recursos.

Juntament amb la creació del model de categories, la descàrrega de categories resulta una de les etapes més complexes. Això es deu al fet d’haver d’emplair fluxos en cadascuna de les tres capes (presentació, domini i dades) per tal d’assegurar que aquesta operació es realitza asincronament.

El disseny dels continguts a mostrar suposa la definició d’un document XML per a cadascuna de les planes de continguts (Principal, Categoria i Canal), així com diferents fulls d’estil XSLT per generar les pàgines HTML. En aquesta etapa s’han de tenir en compte les limitacions del widget GtkHTML, que només suporta HTML 4.01 sense JavaScript ni CSS. També pot caldre dissenyar imatges per fer servir en la decoració.

La construcció del contingut HTML s’ha de realitzar asincronament mitjançant un altre flux, per permetre que es puguin carregar múltiples pestaunes a la vegada. L’ús de fluxos pot complicar aquesta etapa, per tal d’assegurar l’exclusió mútua entre els dos fluxos que han d’escrivir als widgets simultàniament.

El disseny de l’historial de navegació té la complicació addicional d’haver de modificar els botons de navegació endavant/endarrere cada vegada que es canviï de pestaña. Això suposa haver de considerar un disseny més dinàmic per a aquests widgets, a diferència de la resta de widgets de la interfície.

L’etapa de millores addicionals inclou tasques a realitzar que milloren la usabilitat de l’aplicació com ara suport per a podcasts, integració amb el gestor de sessions i traducció de l’aplicació.
També és molt important assegurar-se de que les nostres interfacies segueixen els criteris de les
Gnome HIG, per assegurar la coherència amb les altres aplicacions de l'escriptori.

- **Riscos**: com s'ha comentat en l'etapa d'investigació, un dels components que suposen un
aprenentatge més difícil és GObject. Tot i el gran avantatge de poder disposar d'orientació a
objectes, la codificació necessària per definir i implementar classes i interfacies resulta més
complexa que fer servir un llenguatge realment orientat a objectes. És recomanable desenvolupar
les aplicacions Gnome en C++, Python o C#, aïllat que disposen de bindings que permeten
treballar amb GObject de manera més eficient i ràpida.

El disseny del model de categories resulta força complex, ja que ha de disposar de mètodes
emprats en moltes classes, però també d'altres específics d'alguns components. Les operacions
de verificació de restriccions d'integritat en un GtkTreeModel resulten complicades, a causa de la
forma d'accèdir a les dades del model. L'ús de les classes ajudants per modificar el model, com
ara CategoryElement i ChannelElement, permeten simplificar alguns d'aquests aspectes.

També l'ús de fluxos en les operacions d'E/S asíncrones, com ara en la descàrrega dels canals i
la construcció dels continguts HTML, suposen un risc degut a les complicacions d'haver
d'assegurar l'exclusió mútua en GTK+. Aquest tipus d'errors no sempre són deguts a nostre codi:
també es poden trobar biblioteques que no implementen adequadament l'exclusió mútua de
GTK+. Així ha passat en el desenvolupament d'aquest projecte, on un bug d'aquest tipus en el
component GtkHTML3 va produir un retard considerable en l'avanç65.

Tot i estar assignada com a tasca de millora, la correcció de memory leaks amb Valgrind hauria de
ser una tasca realitzada al final de totes les subtasques majors. Cal tenir en compte que els
memory leaks són un dels defectes més comuns en les aplicacions i més difícils de detectar
mitjançant l'anàlisi visual del codi. També permet detectar un ús excessiu de memòria dinàmica
resident, que pot implicar un redisseny de certes operacions. Atès que el llenguatge C no disposa
d'una garbage collector automatitzat, aquesta tasca ha de ser tinguda molt en compte.

### 13.4. Avaluació dels costos de desenvolupament

Gràcies a la gratuïtut de les biblioteques del Gnome, no cal considerar cap cost addicional per poder fer
servir aquesta tecnologia. De la mateixa manera, les aplicacions per al desenvolupament i utilitats més
comunes es poden obtenir gratuïtament, mitjançant distribucions o des dels propis projectes a Internet. En
el cas que es vulgués emprar un entorn de hosting extern per realitzar el desenvolupament, també
existeixen opcions gratuïtes que proporcionen un conjunt d'aplicacions predeterminades per a les llistes de
correu, repositori de canvis i bugzilla, i en faciliten força l'administració.

Quant al nombre de treballadors, un sol programador pot afrontar totes les tasques necessàries. Fins i tot
el disseny HTML dels continguts, que no requereix uns grans coneixements en arts gràffiques. L'etapa de
desenvolupament es pot reduir força si es disposen de més recursos per repartir les tasques.

---

65 Més informació quant a l'error i la seva solució: [http://bugzilla.gnome.org/show_bug.cgi?id=365258](http://bugzilla.gnome.org/show_bug.cgi?id=365258)
Apèndix A. Aplicacions d’ajuda al desenvolupament

- **Anjuta IDE**. Anjuta és un entorn integrat de desenvolupament que facilita les tasques més habituals dels programadors. Algunes de les seves característiques inclouen:

  - **Generació automàtica de diferents tipus de projecte. Anjuta** és capaç de crear un esquelet bàsic de projectes GTK+/Libglade/Gnome, i genera els fitxers necessaris per construir l’aplicació amb *automake/autoconf*. La seva interfície permet modificar-ne alguns aspectes, com ara les biblioteques suportades per a la compilació, la inclusió de símbols per a depuració o macros d’automake personalitzades.

  - Anjuta permet realitzar les tasques més habituals de muntatge, com ara executar l’script configure per generar els automakes, fer la instal·lació de la nostra aplicació o generar el tarball distribuïble amb el codi font. Des de l’IDE podem iniciar la compilació directament, i en cas que es produeixin errors o avisos només caldrà fer un clic en els missatges perquè s’obri el fitxer en el punt exacte on s’ha produït.

- **Integració amb gdb. Anjuta** fa servir gdb per a depurar la nostra aplicació, i proporciona una forma senzilla de definir breakpoints directament sobre els fitxers de codi font. Es disposen de diferents vistes per consultar l’estat del programa durant la depuració, com ara la traça de la pila, els fluxos que està fent servir la nostra aplicació o el valor actual de les variables del programa.

- **Integració amb CVS. Mitjançant el menú “CVS” tenim accés a les tasques més comunes realitzades amb aquest gestor de versions: obtenir les diferències del fitxer que s’està editant respecte a la còpia del repositori, fer commit dels canvis més recents de tot el projecte o fer el checkout d’un nou projecte.

- **Integració amb Devhelp. Devhelp** és el navegador de les API de programació de l’escriptori Gnome. Per obtenir ajuda sobre una funció o una classe, només cal situar el cursor de l’editor sobre el text i prèmer Ctrl-H. Devhelp s’obrirà automàticament mostrant-

---

66 La versió estable més recent d’Anjuta és la 1.2.4, i es pot descarregar des de [www.anjuta.org](http://www.anjuta.org).
Valgrind\(^{67}\). Un dels problemes més difícils de diagnosticar en les aplicacions C/C++ són els acceders incorrectes a memòria. Bé mitjançant els perilllosos atacs de buffer overflow, bé per l'esgotament dels recursos del sistema provoca pels memory leaks (memòria dinàmica assignada però no aliberada), aquests bugs solen ser ignorats pels programadors ja que requereixen una anàlisis detallada de totes les crides emprades, tant de la nostra aplicació com d'altres biblioteques. Valgrind permet detectar memory leaks, acceders a regions de memòria sense inicialitzar i executar que emprin variables buides o aliberades anteriorment.

A diferència de les biblioteques C existents per trobar errors d'aquest tipus (com ara ElectricFence), Valgrind no necessita que les aplicacions es recompilin per fer servir crides especials. Internament, implementa una CPU x86 sintètica sobre la qual executar una aplicació compilada per a aquesta arquitectura (per tant no es pot fer servir en altres arquitectures). Valgrind decodifica i executa les instruccions de la nostra aplicació, i traça els acceders a memòria del processador sintètic. També reemplaça les crides malloc() / free() de glibc per comptabilitzar la memòria dinàmica demanada i trobar memory leaks. Per tot això, Valgrind és l'aplicació més completa per detectar aquest tipus de problemes comuns.

En finalitzar l'execució d'una aplicació amb Valgrind, es genera un fixter de text amb informació dels errors que s'han trobat. En aquest fixter s'informa dels següents errors:

- **Memòria no aliberada perduda.** Valgrind ha trobat memory leaks en la nostra aplicació, i li llista la traça de crides de la pila on es va demanar aquesta memòria:

  11021 (9086 direct, 1935 indirect) bytes in 491 blocks are definitely lost in loss record 214 of 243
  at 0x1b009222: malloc (vg_replace_malloc.c:130)
  by 0x1c3749ff: g_malloc (gmem.c:137)
  by 0x1c386c3a: g_strdup (getrfuncs.c:91)
  by 0x1c3f329b: rsrv_state_reinherit (in /usr/lib/librsvg-2.0.so.2.9.5)
  by 0x1c3f3488: rsrv_state_inherit (in /usr/lib/librsvg-2.0.so.2.9.5)
  by 0x1c3f34c1: rsrv_state_push (in /usr/lib/librsvg-2.0.so.2.9.5)
  by 0x1c3f2826: (within /usr/lib/librsvg-2.0.so.2.9.5)
  by 0x1c3f2962d: rsrv_defa_drawable_draw (in /usr/lib/librsvg-2.0.so.2.9.5)
  by 0x1c3f393c5: rsrv_handle_get_pixbuf (in /usr/lib/librsvg-2.0.so.2.9.5)
  by 0x1c3eab5b: (within /usr/lib/gtk-2.0/2.4.0/loaders/swg_loader.so)
  by 0x1c254a10: gdk_pixbuf_loader_close (gdk-pixbuf-loader.c:744)
  by 0x1b9a9810: html_image_factory_end_pixbuf (htmlimage.c:1118)

- **Memòria no aliberada però referenciada.** Valgrind ha trobat memòria que no s'ha aliberat en sortir de l'aplicació, però encara n'existien punters que la referenciaven. El programador ha de decidir si es tracta o no d'un memory leak, ja que és habitual que les aplicacions no aliberin la memòria abans de sortir, ja que el nucli la reclamarà igualament:

  170266 bytes in 188 blocks are still reachable in loss record 217 of 222
  at 0x1b0090222: malloc (vg_replace_malloc.c:130)
  by 0x1c9c22c2: (within /usr/lib/libfreetype.so.6.3.7)
  by 0x1c9c71ec: FT_Alloc (in /usr/lib/libfreetype.so.6.3.7)
  by 0x1c9f2b26: (within /usr/lib/libfreetype.so.6.3.7)
  by 0x1c9f2d5c: (within /usr/lib/libfreetype.so.6.3.7)

\(^{67}\) La majoria de distribucions disposen de Valgrind entre els seus paquets d'instal·lació. La seva web és [http://valgrind.kde.org/](http://valgrind.kde.org/)
Memòria alliberada parcialment. Només s'ha alliberat una part d'un bloc de memòria dinàmica:

11021 (9086 direct, 1935 indirect) bytes in 491 blocks are definitely lost
in loss record 214 of 243
at 0x1b909222: malloc (vg_replace_malloc.c:130)
by 0x1c7499ff: g_malloc (gmem.c:137)
by 0x1c36c3a: g_strdup (gstrfuncs.c:91)
by 0x1cf3129b: rsvg_state_reinherit (in /usr/lib/librsvg-2.so.2.9.5)
by 0x1cf34bb: rsvg_state_inherit (in /usr/lib/librsvg-2.so.2.9.5)
by 0x1cf34c41: rsvg_state_push (in /usr/lib/librsvg-2.so.2.9.5)
by 0x1cf29826: (within /usr/lib/librsvg-2.so.2.9.5)
by 0x1cf296d2: rsvg_def_drawable_draw (in /usr/lib/librsvg-2.so.2.9.5)
by 0x1cf199c5: rsvg_handle_get_pixbuf (in /usr/lib/librsvg-2.so.2.9.5)
by 0x1ce4c5cb: (within /usr/lib/gtk-2.0/2.4.0/loaders/svg_loader.so)
by 0x1c254a10: gdk_pixbuf_loader_close (gdk-pixbuf-loader.c:744)
by 0xb9a9fb10: html_image_factory_end_pixbuf (htmlimage.c:1118)

Traces de funcions que fan servir variables sense inicialitzar. Per exemple, una funció
printf() que fa servir una variable char* que no s’ha inicialitzat. Pot provocar un error
de seguretat mitjançant atacs de buffer overflow:

Syscall param write(buf) points to uninitialised byte(s)
at 0x1c19108b: (within /lib/libpthread-2.3.5.so)
by 0x1c57246: _X11TransWrite (in /usr/X11R6/lib/libX11.so.6.2)
by 0x1c56180a: (within /usr/X11R6/lib/libX11.so.6.2)
by 0x1c561925: _XReply (in /usr/X11R6/lib/libX11.so.6.2)
by 0x1c54d9b4: XInternAtom (in /usr/X11R6/lib/libX11.so.6.2)
by 0x1c1f5658: gdk_x11_atom_to_xatom_for_display (gdkproperty-
x11.c:197)
by 0x1c1f63a4: gdk_x11_get_xatom_by_name_for_display (gdkproperty-
x11.c:415)
by 0x1c1fc871: gdk_window_set_role (gdkwindow-x11.c:2704)
by 0xb7f341b: ??? (gtkwindow.c:4113)
by 0xc135816: _gclosure_marshall_VOID_VOID (gmarshal.c:77)
by 0xc13949a9: ??? (gclosure.c:514)
by 0xc130a284: _g_closure_invoke (gclosure.c:437)

Traces de funcions que basen un salt condicional en variables sense inicialitzar. Per
exemple, una construcció if que fa servir una variable no definida. Pot provocar un error de
seguretat mitjançant atacs de buffer overflow:

Conditional jump or move depends on uninitialised value(s)
at 0x806769d: saleena_tab_title_gui_set (SaleenaTabTitleGUI.c:186)
by 0x8067aae: saleena_tab_title_gui_set_from_element
(SaleenaTabTitleGUI.c:276)
by 0x806a65: saleena_tab_browser_gui_set_state
(SaleenaTabBrowserGUI.c:383)
by 0x806a260: saleena_tab_browser_gui_new (SaleenaTabBrowserGUI.c:265)
by 0x80662bb: saleena_tabs_guinit_tab (SaleenaTabsGUI.c:737)
by 0x80668bb: saleena_tabs_gui_new (SaleenaTabsGUI.c:245)
by 0x8066a01: saleena_tabs_gui_new (SaleenaTabsGUI.c:288)
by 0x8063f0b: saleena_gui_new (SaleenaGUI.c:862)
by 0x8055d55: main (main.c:118)
A més de trobar memory leaks, el core de Valgrind permet desenvolupar nous tipus de perfiladors, com ara cacheGrind, que simula accessos a una cache sintètica per comprovar l'eficiència de la nostra aplicació en aquest aspecte. També existeixen frontends gràfics per fer servir Valgrind: Alleyoop68, una aplicació GTK+ que permet executar l'aplicació des d'una interfície gràfica, i visualitzar les estadístiques que generades.

Entre els seus principals inconvenients, cal tenir en compte que executar una aplicació sota Valgrind requereix força memòria addicional (la mida del codi s'incrementa fins a 12 cops). Un altre problema és la velocitat d'execució, ja que les aplicacions s'executen força més lentament sobre la CPU sintètica (fins a 40 vegades més lent). Per això és millor combinar Valgrind amb algun tipus de sistema de test automatitzat que alliberi al programador d'aquesta tasca.

- Devhelp69. Devhelp permet navegar per les pàgines de documentació HTML de les API de programació del Gnome. També permet fer cerques sobre els només de funció, tipus de dades i classes.

- Dia70. Dia és una aplicació GTK+ per a l'edició de diagrames. Entre els símbols de dibuix disponibles, es troben objectes per a la creació de diagrames UML de classe i de sequència. Permet exportar el resultat a multitud de formats, des de PNG a SVG.

- Planner71. Planner és un senzill planificador de projectes per al Gnome. Amb aquesta aplicació podem crear diagrames de Gantt que mostren la progressió de les diferents tasques involucrades en un projecte. Proporciona una gran flexibilitat a l'hora de definir calendaris laborals, i permet imprimir els diagrames en format PDF.

---

68 http://alleyoop.sourceforge.net/
69 http://www.imendio.com/projects/devhelp/
70 http://www.gnome.org/projects/dia
71 http://www.imendio.com/projects/planner/
Apèndix B. Infraestructura tècnica per a la gestió de projectes de programari lliure

Els projectes de programari lliure fan servir un conjunt de sistemes per a la gestió efficient dels seus recursos. A continuació veurem un resum general d'aquestes tecnologies, totes de sobres conegudes. Tot i que podem crear aquesta infraestructura en el nostre propi servidor, és molt recomanable emprar algun servei de hosting gratuït, com ara SourceForge72, Savannah73 o BerliOS74. A banda d'estalviar-nos problemes amb la instal·lació i l'administració, disposen d'una millor integració entre els seus components. Lluny de tractar-se de subsistemes aïllats, per obtenir els millors resultats cal fer servir conjuntament aquestes aplicacions. La creació d'un projecte de programari lliure involucra la configuració de tots aquests sistemes:

- **Control de versions.** Una de les primeres decisións que s'han de prendre en començar un projecte és quin sistema es farà servir per a la gestió dels canvis. L'aplicació més popular i veterana és sense dubte CVS75, que proporciona les funcionalitats més bàsiques d'aquest sistema. Tot i ser l'opció més estesa entre els projectes de programari lliure, poc a poc SVN76 ha anat guanyant-li terreny, principalment per la facilitat per a treballar amb branques. Per a projectes molt grans on el control en un repositori centralitzat resulta massa conflictiu, existeixen els sistemes de control de versions distribuïts: Bazaar-NG/bzr77 i git78 en són els més populars. Aquests sistemes permeten fer commits off-line, retardant la sincronització amb el repositori fins més endavant. A l'hora de decidir-nos per un sistema en particular, cal tenir en compte que la majoria de desenvolupadors només han treballat amb CVS, i l'ús de sistemes pot coneguts pot suposar una barrera massa gran a l'hora de participar en un projecte.

Habitualment, només se sol posar el codi font de les aplicacions sota els sistemes de control de versions. Tanmateix, és una bona pràctica emprar-lo per a qualsevol tipus de document que sigui

---

72 [http://sourceforge.net/](http://sourceforge.net/)
74 [https://developer.berlios.de/](https://developer.berlios.de/)
75 [http://www.nongnu.org/cvs](http://www.nongnu.org/cvs)
76 [http://subversion.tigris.org/](http://subversion.tigris.org/)
77 [http://bzr.or.cz/](http://bzr.or.cz/)
78 [http://bazaar-ng.org/](http://bazaar-ng.org/)
ediat sovint. Això inclou, per exemple, documents referits al disseny, les ajudes de l'aplicació i les pàgines web. L'únic excepció a aquesta regla són els fitxers generats automàticament, com ara els makefiles o les plantilles de traducció .pot.

Una de les maneres més populars d'accédir als repositoris és mitjançant una interfície web. Això permet consultar-los anònimament, per exemple per obtenir la versió més recent dels fitxers PO de traducció. Aquestes intercies estalvien als usuaris haver d'instal·lar una nova aplicació, i en molts casos eviten les complicacions d'haver de conèixer els detalls més tècnics de la gestió de versions, com ara fer diffs entre branques o fer el checkout de tot un projecte. També, exposar el repositori mitjançant una web permet que els serveis d'indexació d'informació, com ara Google o Yahoo!, puguin fer-hi cerques.

Quan es fa un commit, aquests sistemes permeten executar scripts i aplicacions per processar les dades del canvi. Això permet notificar immediatament dels canvis en un repositori mitjançant diferents medis: correus electrònics, missatges en xats o feeds RSS. Si a més es disposa d'una interfície web per visualitzar el repositori, es pot incloure un enllaç que identifiqui univocament aquest commit, de manera que altres persones poden visualitzar els canvis sense haver de disposar d'una còpia del repositori localment.

- **Llistes de correu.** A banda dels fòrums i els xats, les llistes de correu són el punt d'accés públic més habitual en els projectes de programari lliure. Ja sigui com a participant actiu o com a observador extern, milers de persones se subscriuen a les llistes de correu per estar al dia de les últimes decisións que es produeixen al voltant d'un projecte.

Tot projecte hauria de disposar almenys de les següents llistes públiques: usuaris, desenvolupadors, anuncis d'alliberaments, traductors i anuncis de correccions de seguretat. Per a certes discussions, és preferible fer servir una llista privada, a la qual només ens podrem subscriure per invitació (per exemple, per notificar un projecte d'errors de seguretat).

Existeix abundant programari per a la gestió de llistes de correu. Se sol disposar de senzilles intercies web per controlar-ne la majoria d'aspectes, de manera que la configuració i administració no ens ha de suposar cap problema. Amb aquests programes disposem de les següents funcionalitats:

a) **Gestiò de subscripcions.** Habitualment, una persona que vulgui subscriure's a una llista només haurà d'emplenar la seva adreça de correu en un formulari web. El sistema realitzarà les tasques necessàries per afegir-los a les llistes, i fins i tot ens permetrà escolir entre rebre un resum al final del dia en comptes de missatges individuals. També, es poden gestionar les subscripcions mitjançant correus amb continguts especials, com ara un cos de missatge amb paraules clau per donar-se de baixa.

b) **Filtratge anti-spam.** Actualment, l'spam és el problema més greu d'Internet. Els gestors de llistes de correu disposen de mesures per evitar l'enviament d'aquest tipus de missatges a les llistes, com ara usar fíltres anti-spam o publicar només els missatges procedents d'adreces subscriutes. Tanmateix, els missatges rebutjats haurien de ser revisats per l'administrador de la llista de correu abans de ser esborrats; sovint, persones no subscriutes s'adrecen directament a les llistes per fer pregунtes i suggeriments. En aquests casos, cal reenviar-les a la llista perquè puguin ser respostes.
Una altra de les tasques que poden realitzar automàticament els gestors de llistes és ofuscar les adreces de correu. S’ha demostrat empíricament que, fins i tot fent servir esquemes força simples (com ara nom punt cognom arroba servidor punt com) es pot reduir el nombre de missatges d’spam rebut en una adreça.

c) **Arxivatge.** Els missatges rebuts a les llistes de correu se solen emmagatzemar en arxius, accessibles mitjançant una pàgina web. Els arxius són un altre dels mètodes emprats per seguir una llista externament, i estalvia el problema d’haver de rebre molts missatges de correu a diari. Un altre dels avantatges és que proporcionen un històrial complet de converses, decissions, patches, etc. des de l’inici del projecte. Sovint, els nouvinguts a un projecte solen fer les mateixes preguntes; és habitual redirigir-los cap als arxius, on molt probablement trobaran les respostes que necessiten.

Un dels aspectes més importants de la configuració dels arxius és que l’adreça web que rebin els missatges sigui permanent. Això permet emprar efectivament els missatges arxivats en les respostes dels nous missatges o en el cos d’un entrada del bug tracker, amb la seguretat que no es perdrà el fil de la conversa encara que passin molts anys. També, cal que el període d’actualització dels arxius sigui breu, per tal que no hi hagi grans diferències entre els fils de discussió de correu i les seves rèpliques arxivades.

d) **Cerques.** Els gestors de llistes de correu han de proporcionar algun mecanisme senzill per fer cerques entre els missatges, bé permetent que Google cerqui en el nostre arxiu, bé amb motors de cerca propis per obtenir resultats més acurats (cerca en el cos del text, en el missatge, pel remitent, etc.)

- **Bugzilla.** Els sistemes de seguiment d’errors (bug trackers) són un dels sistemes que necessiten més control per part dels gestors dels projectes. Un dels principals “enemics” per a fer un ús eficient d’aquests sistemes són els errors duplicats, que fan que la base de dades s’ompli de notificacions innecessàries. Almenys un membre del projecte hauria de revisar les noves incidències, per tal d’eliminar les innecessàries ben aviat.

A banda de notificacions d’errors i enviament de patches, els bug trackers també se solen emprar per enviar suggeriments de millors. Aquests sistemes també ajuden en la planificació de les properes versions del nostre projectes, ja que permeten assignar una data orientativa per a la resolució de les incidències.

El cicle de creació i resolució d’una incidència conta amb les següents etapes:

1. **Entrada de la incidència.** Depenent de la quantitat d’informació que ens calgui per poder reproduir un error, aquest formulari pot ser una sola pàgina web (títol, resum, fitxers addicionals/traces del depurador) o varíen en forma d’assistent (el sistema emprat en el bugzilla del Gnome). L’entrada d’incidències també es pot produir per altres mitjans: per exemple, el projecte Gnome fa servir l’aplicació bug-buddy, que recull automàticament les dades d’un programa que ha fallat (la traça de la pila, la versió de les biblioteques emprada, etc.) i introduceix la incidència en bugzilla.gnome.org mitjançant un missatge XML-RPC.
2. **Reproducció de la incidència.** Aquest és un dels punts més crítics del procés de resolució, i pot involucrar posar-se en contacte amb la persona que ha introduït la incidència per obtenir informació addicional. Algunes projectes consideren obligatori que almenys una altra persona informi de la mateixa incidència, fins i tot abans de notificar-la.

3. **Diagnosi del problema.** Tan bon punt es conegui el motiu de l’error, cal introduir aquesta informació en la instància del bug tracker. Altres desenvolupadors poden veure la causa de l’error, i segons la seva experiència poden proporcionar una solució ells mateixos. També resulta important disposar d’aquesta informació quan un error anteriorment solucionat es reobre (el que es coneix com a regressió). En aquesta etapa, cal establir qui serà l’encarregat de resoldre la incidència.

4. **Planificació de la solució.** Depenent de la gravetat de la incidència, cal realitzar una estimació del temps necessari per desenvolupar la solució. Aquesta informació també és molt important per a les persones que van introduir la incidència en el sistema.

5. **Resolució de la incidència.** A banda de tancar una incidència una vegada solucionada, l’administrador pot decidir tancar-la per altres motius (un duplicat d’una altra, un suggeriment descartat o un error que es considera provocat per un motiu al·liat). Si la solució s’ha integrat en el repositori, afegir un enllaç web que identifiqui aquest commit ens permetrà recuperar aquesta informació en el futur.

Quan es produeix un canvi en una incidència, aquests sistemes solen enviar un missatge de correu a la llista de desenvolupament. Tot i que sovint es fa servir el bug tracker com a mitjà de discussió públic, és preferible restringir aquest tipus de comunicació a les llistes de correu, ja que no tothom vigila els canvis que es produeixen en les incidències. En aquests casos, és recomanable emprar els arxius de les llistes juntament amb el bug tracker, incloent l’enllaç web al fil de discussió arxivat en la descripció de la incidència.

- **Wiki.** El wiki és el sistema de creació de continguts web més dinàmic, ja que permet que qualsevol persona en modifiqui el contingut. Els projectes els fan servir per a tot tipus de propòsit, des de pàgina principal fins a bloc de notes on escriure idees i suggeriments. Gràcies a la facilitat perquè múltiples persones puguin editar-ne les pàgines, sovint es fa servir com a mitjà de treball col·laboratiu. És especialment útil quan cal escriure documents complexes, com ara les ajudes de les aplicacions o notes informatives referides als nous alliberaments del projecte.

Tot i això, la llibertat de participació que proporcionen els wikis també comporta els seus problemes. Un dels aspectes més crítics dels wikis és la manca d’homogeneïtat que solen mostrar els seus continguts. Per resoldre aquest problema, els projectes han de definir des del començament unes regles d’estil per als textos, que estableixin el to a emprar a l’hora d’adreçar-se als lectors, així com l’orientació dels continguts. Un altre dels problemes, derivat de l’anterior, és la pobra navegabilitat: sense un criteri clar per organitzar les pàgines, un wiki serà poc més que una col·lecció desordenada d’informació.

- **Xats.** Els projectes solen fer servir els xats per la seva vessant socialitzadora, ja que permet que els programadors i els usuaris interaccionin a un nivell més personal. Com es considera un mitjà menys formal que les llistes de correu, és força habitual adreçar-s’hi per fer consultes ràpides. Molts desenvolupadors solen ser-hi bona part del dia, així doncs és bastant probable obtenir una
La gran majoria de projectes tenen una sala de xat a Freenode®, un servidor de xats gratuït. La web del projecte ha de llistar les sales de xat disponibles (és recomanable separar les converses d’usuaris i desenvolupadors). No és habitual arxivar les converses dels xats, ja que els usuaris solen donar per fèr la volatilitat del mitjà a l’hora d’expressar les seves opinions.
Apèndix C. Llistat codi GObject d'exemple

first iface.h:

```c
#include <glib.h>

#define FIRST_IFACE_TYPE
#define FIRST_IFACE_TYPE(obj) (first_iface_get_type (obj))
#define FIRST_IFACE_CLASS(vtable, FIRST_IFACE_TYPE) (G_TYPE_CHECK_INSTANCE_CAST ((obj),
                                                  (vtable),
                                                  FIRST_IFACE_TYPE, FIRST_IFACE_CLASS))
#define IS_FIRST_IFACE(obj) (G_TYPE_CHECK_INSTANCE_TYPE ((obj),
                                                      FIRST_IFACE_TYPE))
#define IS_FIRST_IFACE_CLASS(vtable) (G_TYPE_CHECK_CLASS_TYPE ((vtable),
                                                  FIRST_IFACE_TYPE))
#define FIRST_IFACE_GET_CLASS(inst) (G_TYPE_INSTANCE_GET_INTERFACE ((inst),
                                                     FIRST_IFACE_TYPE, FirstIfaceClass))

typedef struct _FirstIface FirstIface; /* dummy object */
typedef struct _FirstIfaceClass FirstIfaceClass;

struct UserInfo
{
    gchar * firstName;
    gchar * surname;
    gint age;
    gchar * email;
};

struct _FirstIfaceClass
{
    GType interface parent;
    /* Interface methods */
    gboolean (* set_info) (FirstIface * self, UserInfo * info, GError ** error);
    gboolean (* get_info) (FirstIface * self, UserInfo ** info, GError ** error);
};

GType
first_iface_get_type (void);
/* Public methods */
```
gboolean
first_iface_set_info (FirstIface *self, UserInfo* info, GError** error);

gboolean
first_iface_get_info (FirstIface *self, UserInfo** info, GError** error);

first_iface.c:

#include "first-iface.h"

static void
first_iface_init (gpointer g_class)
{
    static gboolean initialized = FALSE;
    if (!initialized) {
        /* create interface signals here. */
        initialized = TRUE;
    }
}

GType
first_iface_get_type (void)
{
    static GType type = 0;
    if (type == 0) {
        static const GTypeInfo info = {
            sizeof (FirstIfaceClass),
            first_iface_init, /* base_init */
            NULL, /* base_finalizer */
            NULL, /* class_init */
            NULL, /* class_finalize */
            NULL, /* class_data */
            0,
            0, /* n_preallocs */
            NULL /* instance_init */
        };
        type = g_type_register_static (G_TYPE_INTERFACE, "FirstIface", &info, 0);
    }
    return type;
}

gboolean
first_iface_set_info (FirstIface *self, UserInfo* info, GError** error)
{
    return FIRST_IFACE_GET_CLASS (self)->set_info (self, info, error);
}

gboolean
first_iface_get_info (FirstIface *self, UserInfo** info, GError** error)
{
    return FIRST_IFACE_GET_CLASS (self)->get_info (self, info, error);
}

first-object-private.h:

typedef struct _FirstObjectPrivate FirstObjectPrivate;
first-object.h:

#include <gnome.h>

#include "first-object-private.h"

#define FIRST_OBJECT_TYPE (first_object_get_type())
#define FIRST_OBJECT(obj) (G_TYPE_CHECK_INSTANCE_CAST ((obj), FIRST_OBJECT_TYPE, FirstObject))
#define FIRST_OBJECT_CLASS(klass) (G_TYPE_CHECK_CLASS_CAST ((klass),
FIRST_OBJECT_TYPE, FirstObjectClass))
#define IS_FIRST_OBJECT_CLASS(klass) (G_TYPE_CHECK_CLASS_TYPE ((klass),
FIRST_OBJECT_TYPE))
#define IS_FIRST_OBJECT(obj) (first_object_check (obj))

struct _FirstObject
{
  /* Parent object's struct */
  GObject parent;

  /* Public data */
  gchar* username;

  /* Private data */
  FirstObjectPrivate* priv;
};

struct _FirstObjectClass
{
  GObjectClass parent;

  /* Public virtual methods */
  void (*do_decrypt) (FirstObject*);

  /* Signals signature */
  void (*auth_ok) ();
  void (*auth_err) (const gchar* error);
};

typedef struct _FirstObject FirstObject;
typedef struct _FirstObjectClass FirstObjectClass;

/* Public methods */

gboolean first_object_check (void *obj);

FirstObject*
first_object_new ();

void
first_object_do_authenticate (FirstObject* obj, const gchar* password);

first-object.c:

enum
{
  AUTH_OK,
  AUTH_ERR,
  N_SIGNALS
};
static gint signals[N_SIGNALS];

struct _FirstObjectPrivate
{
    /* private data */
    gchar password[256];
}

#include "first-object.h"
#include "first-iface.h"

static GObjectClass* parentClass = NULL;

/* Private methods */

void first_object_do_authenticate_private (FirstObject* obj, const gchar* password)
{
    GError *error = NULL;

    if (first_object_check_password (obj, password, &error))
    {
        g_signal_emit (obj, signals[AUTH_OK], 0);
        /* Copy the password to the private field */
        g_snprintf (obj->priv->password, 256, "%s", password);
    }
    else
    {
        g_signal_emit (obj, signals[AUTH_ERR], 0, error);
        g_error_free (error);
    }
}

/* Initialization, checks */

static void
first_object_init (GTypeInstance * instance, gpointer g_class)
{
    FirstObject* obj = FIRST_OBJECT (instance);

    obj->username = NULL;

    obj->priv = g_malloc0 (sizeof (FirstObjectPrivate));
}

static void
first_object_dispose (GObject * obj)
{
    /* Unref any referenced object */

    /* Call parent class' dispose method */
    G_OBJECT_CLASS (parentClass)->dispose (obj);
}

static void
first_object_finalize (GObject * obj)
{
    /* Free any memory used by our object */

    FirstObject* obj = FIRST_OBJECT (_obj);

    if (obj->username != NULL)
    {
        g_free (obj->username);
    }
Visualizador e organizador de canais RSS/RDF para o escritório Gnome

```c
obj->username = NULL;

if (obj->priv != NULL)
    g_free (obj->priv);
obj->priv = NULL;

/* Call parent class' finalize method */
G_OBJECT_CLASS (parentClass)->finalize (_obj);

static void
first_object_class_init (gpointer klass, gpointer class_data)
{
    GObjectClass *gobject_class = G_OBJECT_CLASS (klass);
    parentClass = g_type_class_peek_parent (klass);

    gobject_class->dispose = first_object_dispose;
    gobject_class->finalize = first_object_finalize;

    FIRST_OBJECT_CLASS (klass) -> do_authenticate =
        first_object_doAuthenticate_private;

    /* Signals registration */
    signals[AUTH_OK] = g_signal_new ("auth_ok",
        G_TYPE_FROM_CLASS (klass),
        G_SIGNAL_RUN_FIRST | G_SIGNAL_ACTION,
        G_STRUCT_OFFSET (FirstObjectClass, auth_ok),
        NULL,
        NULL,
        g_cclosure_marshal_VOID_VOID,
        G_TYPE_NONE,
        0);

    signals[AUTH_ERR] = g_signal_new ("auth_err",
        G_TYPE_FROM_CLASS (klass),
        G_SIGNAL_RUN_FIRST | G_SIGNAL_ACTION,
        G_STRUCT_OFFSET (FirstObjectClass, auth_err),
        NULL,
        NULL,
        g_cclosure_marshal_VOID_VOID_POINTER,
        G_TYPE_NONE,
        1,
        G_TYPE_POINTER);
}

gboolean
first_iface_set_info (FirstIface *self, UserInfo* info, GError** error)
{
    /* Implementation of FirstIface::set_info */
}

gboolean
first_iface_get_info (FirstIface *self, UserInfo** info, GError** error)
{
    /* Implementation of FirstIface::get_info */
}

static void
first_object_first_interface_init (gpointer g_iface, gpointer iface_data)
{
    FirstInterfaceClass *klass = (FirstInterfaceClass *)g_iface;

    klass->set_info = (gboolean (*)(*) (FirstInterface *self,
```
userInfo* info, 
GError** error)
first_object_set_info;

klass->get_info = (gboolean (*)(FirstInterface *self, 
userInfo** info, 
GError** error))
first_object_get_info;

}

GType
first_object_get_type (void)
{
    static GType type = 0;

    if (type == 0)
    {
        static const GTypeInfo info = {
            sizeof (FirstObjectClass),
            NULL, /* base_init */
            NULL, /* base_finalizer */
            first_object_class_init, /* class_init */
            NULL, /* class_finalizer */
            NULL, /* n_preallocs */
            sizeof (FirstObject),
            0, /* n_preallocs */
            first_object_init /* instance_init */
        };

        type = g_type_register_static (G_TYPEOBJECT, "FirstObject", &info, 0);

        static const GInterfaceInfo first_interface_info = {
            (GInterfaceInitFunc) first_object_first_interface_init, /* interface_init */
            NULL,
            NULL
        };

        g_type_add_interface_static (type, FIRST_IFACE_TYPE, &first_interface_info);

    }

    return type;
}

gboolean
first_object_check (void *obj)
{
    return (G_TYPECHECKINSTANCE_TYPE ((obj), FIRST_OBJECT_TYPE) 
&
FIRSTOBJECT (obj)->username != NULL);
}

FirstObject*
first_object_new()
{
    FirstObject* obj = g_object_new (FIRST_OBJECT_TYPE, NULL);

    obj->username = g_strdup (**);

    return obj;
}

void
first_object_do_authenticate (FirstObject* obj, const gchar* password)
{
    FIRSTOBJECT_GET_CLASS (G_OBJECT_GET_CLASS (obj))->do_authenticate
    (obj, password);
}
Bibliografia


Articles a la web


- **The Story of the GNOME project.** De Icaza, Miguel. [http://primates.ximian.com/~miguel/gnome-history.html](http://primates.ximian.com/~miguel/gnome-history.html)


Visualizador i organitzador de canals RSS/RDF per a l'escriptori Gnome


http://developer.gnome.org/projects/gup/hig/

