backimap: backup del correu IMAP

(Projecte de Final de Carrera: Enginyeria en Informàtica de la UOC)

Alex Muntada Duran*

(Consultor: Jordi Delgado Pin)

2011.06.13

** aquest treball no hauria estat possible sense la llum de l'Alba.*

El correu electrònic ha esdevingut una eina molt important per a la societat i crítica per a molts negocis. Amb la proliferació dels proveïdors de correu gratuït a la web, hom pot accedir fàcilment al correu des de qualsevol lloc, fins i tot seguir accedint-hi amb les eines de sempre. Així, moltes empreses i particulars han deixat de tenir serveis de correu propis i han passat a utilitzar el que ofereixen d'altres proveïdors. Però què passa amb les còpies de seguretat del correu? Quins proveïdors ofereixen recuperar correus esborrats? En quines condicions? Durant quant de temps guarden les còpies de seguretat? Què passa si el proveïdor deixa d'oferir el servei? Amb *backimap* hom pot recuperar el control de les còpies de seguretat del correu i romandre una mica més tranquil.

Índex

Capítol 1

Introducció

backimap és una paraula inventada que s'obté de la contracció de *backup* (còpia de seguretat, en anglès) i l'acrònim IMAP (*Internet Message Access Protocol*), que dóna nom a l'eina presentada en aquest projecte. Es tracta d'una eina per a realitzar còpies de seguretat de les bústies de correu emmagatzemat en servidors remots mitjançant el protocol IMAP i està disponible en aquests llocs:

- backimap al Github
- backimap al CPAN

1.1 Justificació

La tria d'aquest projecte sorgeix de la confluència de diversos interessos personals i professionals alhora: d'una banda, la necessitat de disposar d'una eina que permeti emmagatzemar còpies del correu hostatjat a Gmail (*Google Mail*) de forma eficient en espai, que a més es pugui usar en qualsevol altre servei IMAP; de l'altra, la voluntat d'encetar un projecte de programari lliure amb l'excusa del projecte, que a més a més em permeti posar-me al dia de les darreres metodologies i tècniques de desenvolupament.

L'ús de Gmail em facilita moltíssim la gestió del correu personal, però sempre hi ha present la incertesa del que pot passar en cas d'esborrar un missatge important sense voler o de perdre l'accés al compte de correu per qualsevol motiu i no poder llegir el correu acumulat al llarg de tants anys. Gmail només és un dels diversos proveïdors en què aquesta incertesa és present, tot i que sovint els usuaris d'aquests serveis no en són conscients fins que els passa alguna cosa. Però el cas de Gmail és significatiu pel volum de correu que permet emmagatzemar i perquè ofereix serveis de correu per a empreses i organitzacions que han decidit delegar-los. Atesa la possibilitat d'utilitzar el protocol IMAP per accedir als serveis de correu de Google, s'obre doncs l'oportunitat per a utilitzar *backimap* per a qualsevol altre servidor de correu amb suport d'IMAP i dur aquesta eina també al món de l'empresa.

D'altra banda, donant-li forma de projecte de programari lliure, no només puc contribuir a millorar el ventall d'eines disponibles per a la comunitat sinó que el projecte podrà enriquir-se de l'experiència d'altres professionals i les diverses aportacions que sorgeixin per part dels usuaris, ja sigui a través de suggeriments i propostes de canvi, ajuda per a resoldre errors, traduccions, extensions de funcionalitat, etc.

Finalment m'agradaria destacar que fa una pila d'anys vaig començar a participar activament a la comunitat del llenguatge de programació Perl com a membre de l'equip de provadors de codi, era una bona manera d'ajudar i aprendre alhora. Després d'un període llarg d'inactivitat, reprenc la meva participació en la comunitat però ara com a programador i amb la intenció de contribuir amb el meu granet de sorra a un dels més antics i més grans dipòsits de programari lliure que existeix, el *Comprehensive Perl Archive Network* (CPAN), on publicaré el projecte amb una llicència lliure.

1.2 Objectius

S'estableixen els objectius per al projecte tot emmarcant-los en dues categories: d'una banda, m'interessa formar-me i adquirir nous coneixements sobre metodologies i tècniques de desenvolupament àgils, poder aprofundir en els meus coneixements de diversos sistemes de control de versions i poder posar-me al dia de les novetats en programació orientada a objectes del llenguatge Perl; de l'altra, vull que aquest projecte obtingui uns resultats útils per poder-los oferir a la comunitat de programari lliure al mateix temps que em serveixin per a solucionar una preocupació personal.

1.2.1 Formació necessària

- Estudi de les metodologies de desenvolupament àgil.
- Pràctica del desenvolupament basat en tests.
- Estudi dels sistemes de control de versions.
- Perl modern.

1.2.2 Resultats desitjats

- Construir una eina que fa còpies de seguretat del correu amb IMAP.
- Publicar l'eina amb una llicència lliure.
- Anunciar el projecte i engegar una comunitat.

1.3 Enfocament i mètode

Amb una jornada laboral a temps complet, una agenda social com la meva i els compromisos que sorgeixen sovint de forma inesperada em resultaria impossible planificar i executar un projecte seguint les metodologies tradicionals de gestió de projectes. A més a més, aquestes metodologies acostumen a ser feixuges, sobretot si l'equip és petit com en aquest cas. La meva experiència indica que, en el marc d'un projecte de programari lliure, les comunitats agraeixen que hi hagi informació clara de com evoluciona el projecte i quins objectius de futur té, però les metodologies excessivament estrictes i la burocràcia sovint allunyen els col·laboradors. Així doncs, des del començament he tingut

molt clara la meva aposta per les metodologies àgils. Tot i que és un tema que coneixia superficialment de fa temps, fins ara no havia tingut la possibilitat de provar-ho de primera mà i aquest projecte em brinda l'oportunitat per fer-ho.

Parteixo d'uns coneixements suficients sobre aquestes metodologies per començar el projecte sense la necessitat de formar-me a fons, per tant iniciaré el desenvolupament alhora que reservo una part del temps a millorar aquests coneixements mitjançant la lectura diària de llibres sobre aquestes matèries.

Els punts clau per enfocar el projecte són el disseny incremental, els cicles de desenvolupament d'entre una a dues setmanes, una bona adaptació als canvis i comunicació constant amb els usuaris potencials.

1.4 Planificació

Aquest és un projecte a llarg termini, dividit en cinc fases i sense data de finalització ja que es tracta d'un projecte obert a la participació de la comunitat de programari lliure.

1.4.1 Fases del projecte

1. **La primera fase correspon a aquest projecte de final de carrera** (aquesta fase serà a la que faré referència en general quan parli del projecte).
2. Un cop finalitzada la primera fase, publicaré oficialment la primera versió estable de *backimap*, engegaré una comunitat d'usuaris i en faré publicitat.
3. Després de publicar la primera versió estable, miraré d'aconseguir que s'inclogui el `backimap` a les principals distribucions de GNU+Linux (Debian, Ubuntu, Fedora, Red Hat, etc.) i d'altres sistemes operatius.
4. Tot seguit, miraré com puc millorar el projecte perquè es pugui utilitzar com a solució de còpies de seguretat del correu en entorns professionals (aquí apareix una oportunitat de negoci a estudiar en el futur).
5. Amb l'ajuda de la comunitat espero que el projecte s'anirà enriquint amb el temps (de fet, a la llista de temes pendents ja hi ha algunes peticions d'usuaris potencials).

1.4.2 Abast de la primera fase

L'abast de la primera fase d'aquest projecte és el següent:

- Formació i aprenentatge de nous coneixements.
  - Metodologies àgils.
  - Desenvolupament basat en tests.
  - Sistemes de control de versions.
  - Perl modern.
  - Protocol IMAP.
- Desenvolupament del projecte.

  - Llistat de carpetes.
  - Descàrrega de correus.
  - Magatzem de dades.
  - Exclusió de carpetes.
  - Reprendre la còpia després d'una interrupció.
  - Còpies incrementals.
  - Descomposició dels correus en parts.
- Memòria i presentació.

1.5 Definició de rols

Es defineixen els següents rols per al projecte:

- **Consultor**: l'expert en la realització de projectes de final de carrera, supervisa l'elaboració de la memòria i dóna el vist-i-plau al seu lliurament.
- **Client**: la persona que fa portaveu dels usuaris potencials i que determina les prioritats i l'abast en el seu paper d'usuari tipus.
- **Desenvolupador**: el cap de projecte, dissenyador i programador del producte.

1.6 Anàlisi de riscos

1. **Abast del projecte massa gran pel temps disponible**
   - **Descripció**: Aquest és un risc important a considerar en qualsevol projecte i consisteix en no dimensionar correctament l'abast en funció del temps disponible per a dur-lo a terme.
   - **Impacte**: Lliurament del projecte fora de termini.
   - **Probabilitat**: Mitjana
   - **Acció de mitigació**: Amb l'ajuda de les metodologies àgils, realitzaré un disseny incremental, consensuat amb alguns usuaris potencials del producte, i publicaré periòdicament versions funcionals del producte, com a mínim una després de cada cicle de desenvolupament.
2. **Falta de comunicació**
   - **Descripció**: La falta de comunicació, abans i durant el projecte, produeix incertesa deguda a la inestabilitat o el desconeixement dels requeriments, als canvis de direcció i als malentesos.
   - **Impacte**: Canvis en l'abast i el termini del projecte, malestar i abandonament per part del client.
   - **Probabilitat**: Alta

- **Acció de mitigació**: La comunicació amb els usuaris potencials serà constant i faré una avaluació de les prioritats en l'abast després de finalitzar cada cicle de desenvolupament, que serà consensuada amb el client (l'usuari tipus).

3. **Desconeixement, poca experiència o motivació en les eines**

   - **Descripció**: La tria d'un llenguatge de programació amb el qual no tinc experiència, l'aplicació de metodologies de treball tedioses o poc engrescadores, etc.
   - **Impacte**: Dedicació excessiva de temps a formació i aprenentatge, moral baixa.
   - **Probabilitat**: Baixa
   - **Acció de mitigació**: Així com des del començament he tingut clar que el projecte m'havia de motivar suficientment, passa el mateix amb les eines i metodologies: triaré les eines que em semblin més interessants i engrescadores però que no impliquin un temps excessiu d'aprenentatge, de la mateixa manera que apostaré per incorporar aquells trucs i mecanismes que em semblin més interessants i que puguin donar bons resultats d'entre l'enorme conjunt de metodologies existents per a la gestió de projectes.

4. **No disponibilitat d'un entorn de treball adient i estable**

   - **Descripció**: M'he trobat anteriorment que després d'una actualització de versió del sistema operatiu, les eines que utilitzava per a fer un projecte han canviat el funcionament o han deixat de ser compatibles. També pot passar que s'espatlli l'ordinador de treball o, pitjor encara, el disc dur (amb probabilitat de pèrdua de codi i documentació del projecte), etc.
   - **Impacte**: Endarreriment de les tasques i pèrdua d'informació vital.
   - **Probabilitat**: Alta
   - **Acció de mitigació**: Mitjançant l'ús de sistemes de control de versions distribuïts (un dipòsit a GitHub), serveis d'emmagatzemament al núvol (Ubuntu One i Dropbox) i la publicació de versions noves a cada cicle de desenvolupament, podré disposar de diverses còpies i històrics de la informació del projecte. D'altra banda, he decidit no actualitzar la versió del sistema operatiu dels equips de casa i de la feina per a poder tenir el mateix entorn estable en tots tres fins que faci el lliurament del projecte, com a mínim. Finalment, per a poder treballar des de qualsevol lloc còmodament he adquirit un portàtil.

5. **Problemes amb els mòduls d'altres autors**

   - **Descripció**: Utilitzar codi d'altres autors permet agilitzar la feina i concentrar-se en realitzar les tasques previstes enlloc de reinventar la roda per a cada minúscula part del projecte. Però tothom comet errors i els mòduls d'altres autors poden tenir errors que afectin algunes parts crítiques del projecte.
   - **Impacte**: Retards en la realització de les tasques.
   - **Probabilitat**: Mitjana

- **Acció de mitigació**: Quan una cosa no funcioni el primer que faré serà mirar d'aïllar el problema per determinar si és per un error meu o d'algú altre. En el primer cas, el resoldré directament però en el segon buscaré si es pot trobar un camí alternatiu per evitar l'error mentre n'informo l'autor i l'ajudo a trobar una solució definitiva.

6. **Dependència de proveïdors de serveis externs**

   - **Descripció**: El proveïdor de l'ADSL, els servidors del correu IMAP, els servidors de noms (*Domain Name System*, DNS), etc. ofereixen la infraestructura necessària per a la realització d'aquest projecte.
   - **Impacte**: No poder realitzar proves reals.
   - **Probabilitat**: Baixa
   - **Acció de mitigació**: La compra d'un portàtil m'ofereix la flexibilitat necessària per evitar la dependència dels proveïdors de xarxa. Quant als servidors IMAP, disposo de diversos comptes de correu amb proveïdors diferents per a fer proves i, a més a més, existeixen d'altres proveïdors gratuïts als que podria tenir accés en cas necessari.

7. **Imponderables IRL**

   - **Descripció**: IRL és un acrònim de l'anglès *In Real Life*, una expressió habitual per als que dediquem tant o més temps al ciberespai. La vida real sempre té un munt de compromisos i imprevistos: la feina, la família, les amistats, l'activitat social en diferents col·lectius, l'esport, la salut, etc. En una persona activa com jo és un risc que no es pot menysprear.
   - **Impacte**: Retards en l'execució del projecte.
   - **Probabilitat**: Alta
   - **Acció de mitigació**: Prioritzaré la feina del projecte assegurant que es mantenen els cicles de desenvolupament amb la periodicitat prevista i delegaré o declinaré, sempre que sigui possible, explicant la meva dedicació prioritària al projecte. Avaluaré periòdicament el progrés del projecte per detectar si avança dins de termini.

1.7 Productes obtinguts

Com a resultat del projecte espero obtenir un nou projecte de programari lliure prou interessant per a la comunitat perquè vulgui utilitzar-lo i col·laborar-hi. El producte obtingut inicialment amb la finalització de la primera fase del projecte tindrà les característiques següents:

- Una eina que fa còpies de seguretat del correu de Gmail.
- Que també es pot utilitzar amb d'altres servidors mitjançant IMAP.
- Que guarda històrics del correu de forma eficient en espai.
- Que mostra una estimació del temps restant.

- Que permet excloure carpetes de correu de la còpia.
- Que permet fer còpies incrementals per anar més ràpid.
- Que permet extreure els adjunts.
- Que permet exportar el correu a formats coneguts.
- Que permet reproduir l'estat del correu en un dia determinat.
- Que permet veure els canvis entre dues dates.

1.8 Presentació dels capítols

Per tal d'aprofundir en les diverses metodologies i eines utilitzades en aquest projecte, dedicaré un capítol a la formació adquirida i als coneixements previs, que han estat una motivació important tant a l'hora de triar el projecte com per enfocar-lo: les metodologies àgils, el desenvolupament basat en tests, els sistemes de control de versions, les tendències modernes del llenguatge de programació Perl i el protocol IMAP. Seguiré amb una breu explicació dels orígens i el context del projecte, per continuar amb l'evolució del desenvolupament del projecte. Per acabar, faré un seguit de valoracions sobre la cronologia, els riscos i el cost del projecte, i finalment tancaré amb les conclusions.

Capítol 2

Formació i coneixements

Una part important per a mi d'aquest projecte és l'oportunitat de poder posar en pràctica alguns coneixements adquirits i aprofundir-hi més a partir de l'experiència per determinar fins a quin punt són tan interessants i pràctics com sembla. Sovint m'ha passat que quan llegia sobre alguna d'aquestes matèries immediatament pensava que el que m'estaven explicant era exactament el que necessitava per millorar algun aspecte de la meva vida. Com si es tractés d'un llibre d'autoajuda, de seguida pensava que tenien raó i que podria aplicar-ho en tal situació o en tal altra. En general, jo sóc partidari de fer canvis i provar noves idees, però sovint em passa que no tinc temps o qui tinc al voltant no pensa el mateix i aleshores em costa trobar bones oportunitats per experimentar amb els canvis (per experiència sé que nedar contra corrent pot ser engrescador i divertit durant un temps, però acaba sent esgotador).

Ara que trobo l'oportunitat ideal, la vull aprofitar i per això vull dedicar aquest capítol a la formació i els coneixements que he adquirit o millorat per a poder realitzar aquest projecte. Per a cada matèria, faré una introducció i després explicaré per què m'interessa i com l'he aplicat al projecte.

2.1 Metodologies àgils

Les metodologies àgils de desenvolupament fan referència a un grup de metodologies de desenvolupament de programari basades en el disseny incremental i cicles iteratius de desenvolupament on els requeriments i les solucions evolucionen a través de la col·laboració d'equips multidisciplinars i auto-organitzats [Agile Software Development]. Dues de les metodologies més conegudes sota el terme d'àgils són Scrum i la programació extrema (XP, de l'anglès *eXtreme Programming*).

Una de les premisses de la programació extrema que deixen més bocabadat és que es poden eliminar les fases de recollida de requeriments, disseny i test, així com també tots els documents formals que les acompanyen [The Art of Agile Development]. La raó és que se substitueix la recollida de requeriments per la comunicació constant amb el client, que fins i tot pot passar a formar part de l'equip. Enlloc de dedicar tota una fase prèvia a la predicció del disseny, es planifica un disseny de creixement incremental mitjançant la refactorització, molt més simple i que permet oferir resultats immediats al client, que pot provar des del primer moment el producte i decidir com vol que evolucioni. Es planifiquen

cicles curts de desenvolupament i de durada fixa per tal que el client pugui predir quan tindrà una nova versió del producte. No cal una fase dedicada exclusivament a fer tests perquè els tests formen part del procés de desenvolupament de cada cicle.

Un dels objectius principals de les metodologies àgils és que les persones estiguin motivades amb la feina, se sentin part de l'equip i dediquin tot el seu potencial al projecte. Les hores extres i la productivitat, en termes de volum de feina, no formen part dels objectius. En canvi, els productes de qualitat, sense errors i amb el vist-i-plau del client sí.

Per aquest projecte, he apostat per aquestes metodologies perquè crec que el disseny incremental i els cicles de desenvolupament, combinats amb la comunicació amb els usuaris i clients són la millor estratègia per mantenir la meva motivació amb el projecte i tirar-lo endavant amb l'ajuda de la comunitat de programari lliure.

A més a més, he combinat aquestes metodologies amb tècniques d'organització del temps [Getting Things Done] per fer el seguiment de les tasques.

2.2 Desenvolupament basat en tests

El desenvolupament basat en tests és una de les tècniques més important i alhora controvertides de la metodologia de programació extrema. La tècnica consisteix en escriure un test que descrigui un cas d'ús abans de fer el desenvolupament corresponent, de forma que el test fallarà. A continuació es desenvolupa la solució mínima necessària perquè el test passi. S'escriu un segon test, que tornarà a fallar, després es desenvolupa la solució mínima perquè el test passi, etc. La idea d'aquesta tècnica és que arriba un punt després de diverses iteracions en què cal refactoritzar les solucions per a generalitzar-les i adaptar-les als nous casos d'ús (els nous tests, en aquest cas) i d'aquesta manera s'aconsegueix construir una solució de forma incremental. Un cas concret d'aplicació d'aquesta tècnica és en la investigació i solució d'errors del programari (coneguts universalment amb el terme anglès *bugs*): una recomanació molt habitual abans de fer front a un error és crear un test que permeti reproduir-lo, per tal de tenir-lo ben identificat; a continuació es desenvolupa la solució i es comprova que el test de l'error ja no falla. D'aquesta manera, si l'error es tornés a produir (un fet que s'anomena regressió) ràpidament el test ens alertaria.

Els experts en la matèria expliquen que cal dedicar temps i ser constant per dominar la tècnica perquè costa aconseguir que esdevingui un hàbit i al principi no s'aprecien tots els beneficis que ofereix respecte a la dedicació que necessita [The Art of Agile Development]. Però els tests són una part essencial en qualsevol projecte de programari i, en aquest cas en concret, són part quotidiana de la comunitat de desenvolupadors del llenguatge de programació Perl, amb una gran diversitat de mòduls de test i documentació extensa sobre el tema [Perl Testing: A Developer's Notebook].

L'objectiu que em plantejo per aquest projecte és experimentar amb el desenvolupament basat en tests, especialment per al tractament dels errors. Però sóc conscient que aquesta tècnica imposa molta dedicació i que, per limitacions de temps, hauré de triar quines són les oportunitat més adients per a dedicar-hi prou temps.

2.3 Sistemes de control de versions

Una eina que gestiona i fa el seguiment de diferents versions del codi o d'altres tipus de contingut és el que s'anomena genèricament un sistema de control de versions [Version Control with Git]. Aquests sistemes poden ser centralitzats o distribuïts, segons si requereixen estar sempre connectats a un servidor central o no per a realitzar les operacions sobre els fitxers que gestionen. Segons si emmagatzemen el contingut sencer de cada versió (sovint anomenats *blobs*) o només les diferències (també conegudes com a *deltas*), podem distingir sistemes amb característiques diferents. La facilitat amb què els sistemes de control de versions distribuïts permeten treballar de forma coordinada i descentralitzada alhora, junt amb la flexibilitat que ofereixen per al desenvolupament de canvis privats ha fet que d'uns anys ençà hagin proliferat tot un ventall de serveis d'hostatgeria de projectes al voltant d'aquestes eines (GitHub, Gitorious, Google Code, Bitbucket, Launchpad Branches, etc.).

Les característiques més interessants d'aquestes eines són la traçabilitat dels canvis (mitjançant quaderns de bitàcola i càlcul de diferències entre versions) i la possibilitat de tenir magatzems de dades descentralitzats, en el cas dels sistemes distribuïts.

En aquest projecte utilitzaré el sistema de control de versions per a la gestió dels canvis en el codi i la documentació, com seria habitual en qualsevol projecte de desenvolupament de programari. Però també utilitzaré la característica que ofereixen els sistemes basats en *blobs* per a configurar un magatzem de dades per a les còpies del correu IMAP que sigui eficient en espai i prou ràpid per gestionar diversos milers de fitxers alhora (en aquest cas concret, he triat Git perquè s'ajusta perfectament a aquests requeriments).

2.4 Perl modern

El llenguatge de programació Perl té una pila d'anys i des del seus inicis l'any 1987 fins avui dia ha canviat moltíssim. Una de les majors virtuts d'aquest llenguatge és la seva extensibilitat i l'altra és que hi ha més d'una manera de fer les coses [Programming Perl, Third Edition]. Curiosament aquesta darrera virtud és, segons l'autor original (Larry Wall) un dels dos lemes del llenguatge. L'altre lema és que les coses fàcils han de ser fàcils i les difícils han de ser possibles.

En aquest sentit, i després d'un llarg període amb pocs canvis substancials al llenguatge, d'uns anys ençà s'ha revifat gràcies a un canvi de lideratge, la publicació periòdica de noves versions amb millores i solució a problemes coneguts durant anys i a la proliferació d'un seguit de tendències que s'engloben en el que s'anomena Perl modern [Modern Perl].

Tot i que no està clar en quin moment va sorgir el moviment del Perl modern, un dels seus principals impulsors ha estat el mòdul Moose i la seva visió fresca de la programació orientada a objectes. Aquest mòdul proporciona la base per a un seguit d'eines per a la programació de classes i objectes que d'altres llenguatges tenen de fa temps, que combinant diferents tècniques i mecanismes ofereix un ventall de funcionalitats considerable.

- **Protocol de meta-objectes**: també conegut com a MOP, és un intèrpret de la semàntica d'un programa que és obert i extensible. Així, permet crear nous atributs o mètodes en una classe, construir classes noves en temps d'execució del programa

i en general canviar el comportament de qualsevol objecte del programa a través de modificacions de la seva estructura interna. Aquesta característica fa que sigui relativament més senzill dotar al sistema d'objectes de Perl de noves millores, com ara els atributs derivats (anomenats *lazy*).

- **Subtipus i coerció de tipus**: `Moose` permet definir subtipus i paràmetres de coerció de tipus que permeten realitzar controls més robustos i automàtics dels tipus de dades dels objectes, sense la necessitat d'afegir a mà les comprovacions de tipus. L'avantatge és que Perl ens permet triar quan i com volem utilitzar aquests mecanismes.
- **Rols**: defineixen comportament sense necessitat d'establir una jerarquia, tal com passa amb l'herència. Els rols combinen les tècniques dels *traits* i els *mixins* alhora, permetent una composició horitzontal de la funcionalitat que vulguem atorgar a una classe determinada.
- **Modificadors de mètodes**: faciliten la creació d'embolcalls per als mètodes d'una classe. Existeixen tres tipus de modificadors, segons el moment en què s'apliquen respecte a l'execució del mètode afectat: `before` (abans), `around` (al voltant) i `after` (després). Mitjançant aquests modificadors és possible extendre la funcionalitat d'un mètode sense que calgui rescriure'l.

Però el Perl modern té, a més a més, noves eines que faciliten el desenvolupament de projectes amb aquest llenguatge: App::perlbrew permet instal·lar molt fàcilment diverses versions alhora de Perl a qualsevol directori i triar la que ens calgui a cada moment, App::cpanminus agilitza la instal·lació de mòduls del CPAN i Dist::Zilla simplifica enormement les tasques de construcció d'un projecte en Perl que segueixi les bones pràctiques [Perl Best Practices].

Tot plegat, són motius que m'engresquen a voler triar Perl per al projecte i buscar oportunitats per a experimentar-hi i modernitzar-me jo també.

2.5 Protocol IMAP

Els dos protocols més utilitzats en el servei de correu són POP (*Post Office Protocol*) i IMAP. El primer es basa en la descàrrega dels missatges per treballar-hi localment, el segon treballa directament sobre el servidor i per aquest mateix motiu ha esdevingut el protocol probablement més interessant per a la gestió del correu electrònic. La millora de la velocitat a les xarxes i la capacitat d'emmagatzemament dels servidors ha fet que el protocol IMAP hagi tingut i segueixi tenint una gran repercussió.

El [RFC 3501] descriu amb detall totes les característiques del protocol, però en destacaré principalment que, com és habitual en d'altres protocols de correu, és un protocol de transmissió de text (fet que fa sigui més fàcil per als humans rastrejar els errors i fer proves) i que és un estàndard conegut i provat sobradament (la darrera versió és del 2003) pel qual disposem d'un ventall molt gran d'eines. Per altra banda, es tracta d'un protocol extens i farcit de detalls (el document té més de 100 pàgines).

En aquest cas, es tracta d'una part essencial per al desenvolupament del projecte a la qual caldrà fer referència sovint, tot i que no caldrà estudiar-la minuciosament per implementar-la atès que existeixen diversos mòduls de Perl que ja ho han fet.

Capítol 3

Backup del correu IMAP

Un bon dia, un company em demanava ajuda per missatgeria instantània per a recuperar l'accés al seu correu de Gmail. Em deia que no recordava la contrasenya i que ja no disposava del correu original amb què va registrar-se, així que no podia utilitzar el mecanisme habitual per restablir la contrasenya. Va provar de contactar directament amb el suport tècnic de Google però li demanaven dades que el poguessin identificar com a propietari d'aquell compte i no les tenia. Fet i fet, no se'n va sortir i va perdre l'accés a tot el correu que tenia emmagatzemat en aquell compte des de feia anys.

Avui en dia, un parell d'anys més tard, Google recorda de tant en tant als usuaris que mantinguin al dia les seves dades de contacte i que facilitin algun mecanisme alternatiu de verificació d'identitat (per exemple, amb el telèfon) per evitar que casos com aquest es tornin a produir.

Però no fa pas massa es va produir un greu problema al servei de Gmail i un 0.02% dels usuaris van perdre el seu correu (vegeu el Gmail Fiasco). Els tècnics de Google van trigar diversos dies en poder recuperar el correu de tots els usuaris afectats de les seves còpies de seguretat en cinta i desconec si es va arribar a perdre correu, però és força possible que sí.

Situacions com les que acabo de descriure em van fer buscar quines eines de programari lliure existien per a poder tenir una còpia local del correu de Gmail. Tenint en compte que Gmail ofereix suport per al protocol IMAP des de fa una temporada, hi hauria d'haver solucions per a fer *backups* periòdicament.

En principi qualsevol eina client d'IMAP serviria per a fer les còpies ja que normalment permeten tenir connexions a diferents servidors d'IMAP alhora i copiar correu d'una carpeta a una altra és una operació habitual i fàcil de fer. El problema és que Gmail basa les seves carpetes IMAP en les etiquetes dels correus i n'hi poden haver moltíssimes, de forma que copiar tots els correus de totes les carpetes esdevindria una feina massa feixuga per fer-la manualment.

3.1 Etiquetes de Gmail

Des de bon començament, el servei de Gmail ha ofert la possibilitat de marcar els correus amb etiquetes com a mesura flexible i potent de classificar el correu. Amb gmail és possible crear filtres que assignin automàticament determinades etiquetes als correus

entrants per tal de classificar-los i fins i tot desar-los fora de la bústia d'entrada, lluny de la vista. Aquesta característica de Gmail, juntament amb una gran quota d'espai i unes cerques prou bones (per exemple, cerques per etiqueta) han estat la clau de l'èxit de Gmail per davant d'altres solucions de correu web gratuït existents fins aleshores.

Uns anys més tard, a petició popular, Google oferia accés per IMAP al correu de Gmail. A banda d'algunes particularitats sobre el que significa esborrar un correu de Gmail per IMAP (Google permet triar entre el comportament clàssic d'esborrar o marcar com a esborrat i el comportament equivalent al correu web de Gmail, que és el que s'utilitza per omissió), el comportament del servidor d'IMAP de Google és l'esperat (*Do What I Mean*, DWIM). En aquest sentit, cada etiqueta que l'usuari assigna a un correu correspon a una carpeta d'IMAP. Tot i això, algunes etiquetes que Gmail considera pròpies o especials (com ara el correu enviat, el destacat, l'important, tots els missatges, etc.) són accessibles com a subcarpetes de la carpeta `[Gmail]`.

Així, ens trobem amb què la carpeta de tot el correu dóna accés a tots els correus emmagatzemats a Gmail que no estan classificats com correu brossa. A efectes pràctics, vist des del servidor IMAP, és com si tots els correus tinguessin com a mínim una còpia en aquesta carpeta. Vegem-ne un exemple pràctic per il·lustrar millor les repercussions d'aquesta característica.

Imagineu un correu a la bústia d'entrada que hem classificat amb només 2 etiquetes (feina i lectura). Des del punt de vista d'un client IMAP, hi ha 4 còpies del mateix correu en carpetes diferents (tot i que no pot determinar que són el mateix correu):

1. `INBOX` (la bústia d'entrada)
2. `feina`
3. `lectura`
4. `[Gmail]/Tots els missatges` (suposant que hem triat la interfície en català)

3.2 Sincronització vs. històrics

Quan hom parla de fer còpies de seguretat de dades canviants pot triar entre dos enfocaments: el primer consisteix en tenir una única còpia sincronitzada amb l'estat en què es trobava l'original en el moment de fer l'operació; el segon realitza una nova còpia cada vegada enlloc de sincronitzar la còpia antiga amb l'orginal, de forma que existeix la possibilitat d'accedir a diferents versions de l'original al llarg del temps.

Habitualment el mètode de sincronització utilitza tècniques eficients per a copiar només les parts de l'original que han canviat (per exemple, l'rsync) ja que l'objectiu és que l'operació duri el menys possible i la còpia estigui sincronitzada durant el major temps possible. En canvi, existeixen aproximacions molt diverses per a emmagatzemar les còpies històriques, depenent del volum de dades, del tipus de magatzem (per exemple, en disc, cinta o DVD), de la política d'expiració de les còpies, etc.

El principal inconvenient de la sincronització és la nul·la possibilitat de recuperar una còpia antiga, sobretot si la finestra de temps entre sincronitzacions és molt petita (per exemple, és frustrant esborrar un fitxer i que el procés de sincronització sigui tan ràpid

que no ens doni temps a recuperar de la còpia). D'altra banda, l'inconvenient més important de conservar històrics és la gestió de grans volums de dades i dispositius d'emmagatzemament força més complexos i cars.

3.3 Eines existents

Després de fer recerca de les eines de programari lliure existents, en vaig fer una valoració per determinar si s'ajustaven als requeriments plantejats:

- Offline Gmail és un component (en angles, *plugin*) que permet a diversos navegadors moderns desar els correus i adjunts al disc local, fins i tot els esborranys i els correus pendents d'enviar. Com que es tracta d'un component dissenyat exclusivament per a Gmail, no es pot utilitzar en servidors IMAP. D'altra banda, no serveix per a guardar històrics, només permet sincronitzar el correu.

- Thunderbird Offline Mode és un mode de funcionament d'aquest client de correu, que permet desar al disc local una còpia total o parcial del correu emmagatzemat en un servidor IMAP. És una eina molt similar a l'Offline Gmail però de caire més genèric ja que funciona per a qualsevol servidor IMAP, Gmail inclòs. Però de la mateixa manera que el de Gmail, el mode *offline* del Thunderbird només permet sincronitzar el correu al mateix estat que el que tingui el servidor IMAP, per tant no serveix per a guardar històrics. A més a més, com que no distingeix entre Gmail i qualsevol altre servidor IMAP, si tenim correus amb moltes etiquetes a Gmail, l'espai ocupat per la còpia local d'un correu es multiplica innecessàriament.

- imapsync és una eina per a la interfície d'ordres que agilitza la migració de correu IMAP entre diferents servidors i permet definir una sèrie d'operacions prou interessants com cercar, filtrar i copiar el correu de diversos servidors alhora en un de sol. Malauradament, tampoc té suport per a guardar històrics i a més a més segueix calent un client d'IMAP per a accedir al correu copiat al servidor destí. Per tot plegat, es tracta d'una eina útil però poc interessant per als usuaris en general.

- OfflineIMAP també és una eina per a la interfície d'ordres que potser no ofereix tanta flexibilitat com imapsync a l'hora de migrar el correu IMAP, però és força més útil per als usuaris perquè permet desar les còpies de correu al disc local en un format conegut per a molts clients de correu (Maildir). També pateix el problema de l'espai ocupat pels correus amb moltes etiquetes i, com la resta d'eines, no permet desar un històric dels correus.

3.4 *Scratching your own itch*

Un cop vist que el panorama de les solucions disponibles no s'ajustava a la meva idea d'una eina per a fer còpies de seguretat, vaig decidir que el tema m'interessa i que podria fer-ne una jo que em servís a més a més com a projecte de final de carrera a la UOC. Allò que els anglesos anomenen «rascar allí on pica» (*Whats your problem?* [Getting Real]).

Capítol 4

Desenvolupament

Seguint les recomanacions de les metodologies àgils, he començat el desenvolupament definint un seguit de fites que es van assolint en cada nou cicle iteratiu. L'objectiu d'aquestes fites és proporcionar als usuaris alguna funcionalitat nova (anomenada història) en cada cicle i atendre les seves demandes a mida que es van realitzant, de forma que percebin una millora progressiva.

A continuació procediré a detallar l'evolució del desenvolupament del projecte en l'ordre en què s'han anat produint els canvis.

4.1 Llistat de les carpetes

La primera història és la que permet mostrar un resum de les carpetes existents al servidor, el número total de missatges de cada carpeta i quants d'ells són nous (és a dir, que no han estat vistos encara). Les passes necessàries són les següents:

1. **Obté la descripció del servidor IMAP i les credencials de l'usuari**. Cal descomposar l'identificador (URI) corresponent que proporciona l'usuari a l'ordre `backimap` (per exemple, `imaps://nom.cognom@gmail.com:contrasenya@imap.gmail.com`):
   - L'esquema `imaps:` identifica el protocol IMAP sobre una connexió segura mitjançant TLS/SSL.
   - El nom del servidor (en aquest cas, `imap.gmail.com`) queda a la dreta del darrer caràcter «@».
   - La contrasenya és opcional i s'indica entre els caràcters «:» i el darrer «@». Però si la contrasenya conté qualsevol d'aquests dos caràcters o d'altres de reservats com la «/», es codificaran convenientment amb el mètode del percentatge (URL encode).
   - Finalment el nom d'usuari està comprès entre els caràcters «//» i el «:» o el darrer «@», en cas que la contrasenya no s'hagi indicat.
2. **Estableix la connexió amb el servidor IMAP**. En cas que l'URI no indiqui la contrasenya, li pregunta a l'usuari per a poder autenticar-lo al servidor. Després de diverses proves entre els candidats disponibles al CPAN, el mòdul `Mail::IMAPClient` ha resultat el que s'ajusta millor a les nostres necessitats i ofereix tot el ventall d'opcions del protocol IMAP i més.

3. **Obté la llista de carpetes**. Obté del servidor una llista recursiva de totes les carpetes disponibles.
4. **Mostra els comptadors de cada carpeta**. Examina la carpeta i obté els comptadors corresponents, a menys que contingui d'altres subcarpetes però no missatges.
5. **Tanca la connexió**.

Amb aquesta història es desenvolupa tot el tractament dels URI i de la connexió al servidor IMAP, de forma que l'usuari pot provar si el `backimap` funciona correctament amb diversos servidors, obtenir llistats de les seves carpetes i veure els canvis que es produeixen quan arriba correu nou al seu compte.

4.2 La bústia d'entrada en un dipòsit Git

La història següent consisteix en obtenir els correus de la bústia d'entrada i desar-los en un dipòsit Git prèviament creat:

1. **Obté la localització del dipòsit Git**. L'usuari l'ha hagut de crear prèviament i cal que indiqui la seva localització mitjançant l'opció `--dir` de l'ordre `backimap`.
2. **Crea el directori corresponent a la bústia d'entrada**. Crea el directori `INBOX` dins del dipòsit, per a poder-hi desar els missatges del servidor IMAP.
3. **Obté la llista de correus**. Selecciona la carpeta `INBOX` al servidor i obté la llista dels identificadors de missatge (*Unique Identifier*, UID).
4. **Desa els correus al dipòsit**. Per a cada missatge, n'obté el contingut, crea un fitxer al directori `INBOX` utilitzant el seu identificador com a nom dins la carpeta i l'afegeix al dipòsit Git.
5. **Confirma els canvis introduïts al dipòsit Git**.

Un cop assolida aquesta fita, l'eina *backimap* ja és operativa i permet fer còpies de les bústies d'entrada dels servidors IMAP que s'indiquin, emprant un dipòsit Git per a cadascun. L'usuari ja pot començar a experimentar amb l'eina i obtenir els primers beneficis, encara que per a fer-ho hagi de tenir uns coneixements mínims sobre com construir un dipòsit Git:

```
$ mkdir /backups/gmail
$ cd /backups/gmail
$ git init
$ backimap --dir=/backups/gmail --uri=imaps://nom.cognom@gmail.com@imap.gmail.com/INBOX
```

4.3 Els missatges de totes les carpetes

Tot seguit, la història prevista consisteix en generalitzar el cas anterior per a totes les carpetes IMAP. En aquest sentit es fusionen les dues històries anteriors en un sol bucle:

1. **Obté la descripció del servidor IMAP i les credencials de l'usuari**.
2. **Estableix la connexió amb el servidor IMAP**.
3. **Obté la localització del dipòsit Git**.
4. **Obté la llista de carpetes**.
5. Per a cada carpeta…
   (a) **Mostra els comptadors**. Aquest pas ara és opcional, només es realitza en cas d'indicar l'opció `--verbose` del `backimap`.
   (b) **Crea el directori corresponent a la carpeta**.
   (c) **Obté la llista de correus**.
   (d) **Desa els correus al dipòsit**. En aquest punt, a més a més, decideix que si ja existeix el fitxer corresponent a un identificador, no cal desar aquest correu perquè ja el té i salta al següent.
   (e) **Confirma els canvis introduïts al dipòsit Git**.
6. **Mostra el temps d'execució**, també només en cas que s'hagi indicat l'opció `--verbose` al `backimap`.

Aquesta història acaba de lligar la funcionalitat essencial de l'eina `backimap`, amb la qual un usuari pot fer còpies del seu correu IMAP periòdicament i disposar d'uns històrics en cas que vulgui recuperar algun correu antic. L'usuari pot decidir si vol veure les operacions en detall amb l'opció `--verbose`:

```
$ mkdir /backups/domini
$ cd /backups/domini
$ git init
$ backimap --dir=/backups/domini --verbose --uri=imaps://usuari@imap.domini.tld
```

Però no només això, aquesta fita ens permet comprovar l'eficiència del magatzem de dades d'un dipòsit Git, en què un mateix correu es desa un sol cop al dipòsit independentment del número d'etiquetes que tingui a Gmail o del número de còpies que n'existeixin en un servidor IMAP qualsevol.

4.4 L'estat persistent

En la història anterior, després de recórrer tota la llista de carpetes IMAP recursivament, el `backimap` ha consultat els comptadors de cada carpeta. Doncs bé, ara ens interessa que aquesta informació quedi desada de forma persistent al dipòsit Git per tal de conservar un resum *offline* de la darrera operació realitzada. Així, el `backimap` realitza un seguit de canvis addicionals:

5. Per a cada carpeta…

(a) **Mostra els comptadors**. Ara, no només els mostra els comptadors en cas que l'usuari ho indiqui, també els desa a memòria.

(b) …segueix amb la resta de passes descrites prèviament.

6. **Desa l'estat al dipòsit Git**. Aquest estat persistent consisteix en un fitxer en format JSON que conté els valors següents:

   - una marca de temps del moment en què s'executa el `backimap`,
   - el nom del servidor IMAP,
   - el nom d'usuari,
   - els comptadors de cada carpeta.

7. **Mostra el temps d'execució**.

Per exemple, aquest seria l'estat inicial d'un compte de Gmail després de rebre el primer correu (el fitxer original no conté salts de línia ni espais en blanc, que s'han afegit per tal de fer-ne més fàcil la lectura):

```
{
    "timestamp": 1298661558,
    "user":      "nom.cognom@gmail.com",
    "server":    "imap.gmail.com",
    "counters":  {
        "INBOX":             { "count":"1","unseen":"1" },
        "[Gmail]/All Mail":  { "count":"1","unseen":"1" },
        "[Gmail]/Sent Mail": { "count":"0","unseen":"0" },
        "[Gmail]/Trash":     { "count":"0","unseen":"0" },
        "[Gmail]/Important": { "count":"0","unseen":"0" },
        "[Gmail]/Spam":      { "count":"0","unseen":"0" },
        "[Gmail]/Starred":   { "count":"0","unseen":"0" },
        "[Gmail]/Drafts":    { "count":"0","unseen":"0" }
    }
}
```

4.5 Inicialització del dipòsit Git

A fi d'evitar que sigui l'usuari qui hagi d'inicialitzar el dipòsit Git, afegim una nova opció `--init` al `backimap` per tal que se'n faci càrrec. Així, amb aquesta història afegim unes passes addicionals:

3. **Obté la localització del dipòsit Git**.

   (a) **Inicialitza el dipòsit**. En cas que l'usuari hagi indicat `--init`, el `backimap` primer comprova si el dipòsit ja existia prèviament o si existeix el fitxer d'estat persistent (en qualsevol dels dos casos donaria error). En cas negatiu, procedeix a inicialitzar el dipòsit Git. Tot seguit, com a part de la inicialització, procedeix a crear el fitxer d'estat persistent amb els valors que ja disposa de la marca temporal, l'usuari i el servidor. D'aquesta manera el dipòsit ja queda relacionat amb el compte de correu IMAP corresponent.

(b) **Carrega l'estat anterior**. Tant si ha calgut inicialitzar o no el dipòsit, el `backimap` carrega l'estat anterior i verifica que les dades de l'usuari i el servidor de correu corresponen als actuals. Amb aquesta comprovació, el `backimap` evita que un usuari faci còpies del correu IMAP d'un servidor en el dipòsit d'un altre per error.

La fita d'aquesta història ara permet a l'usuari realitzar la còpia inicial amb una sola ordre, ja que no li cal inicialitzar el dipòsit manualment:

```
$ backimap --dir=/backups/gmail --init --uri=imaps://nom.cognom@gmail.com@imap.gmail.com
```

4.6 Serialització de l'estat

Les metodologies àgils indiquen que no s'han d'optimitzar abans d'hora, ni realitzar canvis en previsió del futur si no estan associats a una història que tingui algun benefici per a l'usuari. Per altra banda, també acostumen a indicar que menys codi implica menys errors i un disseny més senzill. Arribats a aquest punt, es fa evident que l'estat ha de tenir entitat pròpia i que cal construir-lo adequadament. Així, dissenyem un estat molt senzill però modular (Figura 4.1), que permetrà eliminar codi que delegarem a les seves classes.

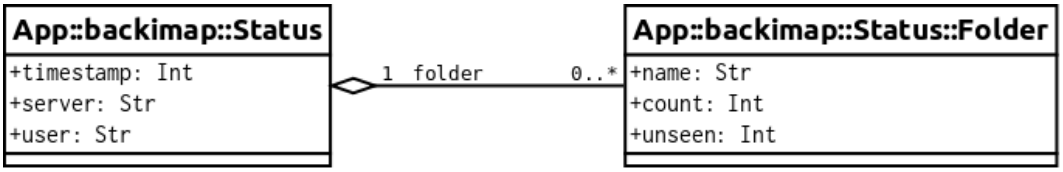


Figura 4.1: *Diagrama de classes de l'estat.*

Aquesta refactorització de l'estat ens permet introduir l'ús del mòdul `Moose` i aprofitar la seva potència en la descripció dels atributs. Però al mateix temps ens permet utilitzar els rols de serialització del mòdul `MooseX::Storage`, que serà l'encarregada de fer la transformació dels objectes de l'estat al format JSON i a l'inrevés, construir-los de nou durant el procés d'inicialització. La descripció d'aquestes classes en Perl seria aquesta:

```
package App::backimap::Status;
use Moose;
use MooseX::Storage;
with Storage( 'format' => 'JSON' );

has timestamp => ( is => 'ro', isa => 'Int', required => 1, default => 0         );
has server    => ( is => 'ro', isa => 'Str', required => 1, default => "server?" );
has user      => ( is => 'ro', isa => 'Str', required => 1, default => "user?"   );
has folder    => ( is => 'rw', isa => 'HashRef[App::backimap::Status::Folder]'   );

package App::backimap::Status::Folder;
use Moose;
use MooseX::Storage;
```

```
with Storage;

has name   => ( is => 'rw', isa => 'Str', required => 1 );
has count  => ( is => 'rw', isa => 'Int', required => 1 );
has unseen => ( is => 'rw', isa => 'Int', required => 1 );
```

4.7 El magatzem de dades

Fins ara havíem utilitzat un dipòsit Git com a magatzem de dades però, tot i que vulguem seguir gaudint de Git per les seves característiques, deslligar el tipus de magatzem de les operacions que ofereix no només aportarà flexibilitat al disseny sinó que permetrà fer tests de manera més senzilla i modular.

En el punt que estem, encara no s'han pogut aplicar les tècniques del desenvolupament basat en tests perquè la solució no és prou modular i no és possible elaborar tests unitaris, tret d'alguna part aïllada com la sintaxi de les URI d'IMAP. Els tests funcionals que s'han pogut realitzar fins aquest moment són manuals degut a què totes les parts de l'eina estan molt entrelligades (la xarxa, el protocol IMAP, el dipòsit Git, l'estat). Així doncs, la fita consisteix en substituir el dipòsit Git per un magatzem de dades amb característiques similars però que ens permeti deslligar les operacions del tipus de magatzem i provar-les més endavant de forma unitària.

Partim d'un disseny molt senzill que anirà evolucionant a mida que sorgeixin les noves necessitats, tal com recomana la metodologia de la programació extrema (*Do the Simplest Thing That Could Possibly Work*, [Extreme Programming Pocket Guide]). Per tant, d'entrada hem descartat definir una classe abstracta sobre la qual definir una subclasse per al magatzem de tipus Git. Tenint en compte els tests unitaris que volem realitzar, la solució més senzilla ara per ara és tenir una sola classe (Figura 4.2).

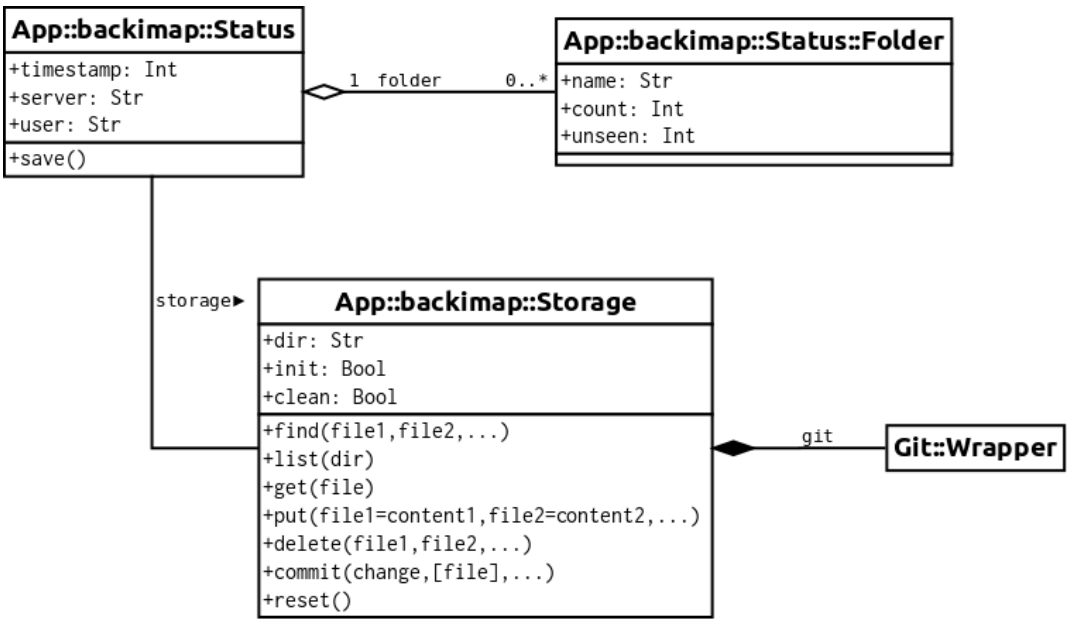


Figura 4.2: *Diagrama de classes del magatzem i l'estat.*

La classe té tres atributs públics: el directori on està situat el magatzem, un booleà que indica si cal inicialitzar-lo i un altre que indica si cal netejar-lo (aquest atribut serà cert si l'usuari ha indicat l'opció `--clean` al `backimap`). També té un atribut derivat per al dipòsit Git, que es construeix a partir dels altres tres. Finalment, la classe ofereix un seguit d'operacions bàsiques per a gestionar el magatzem:

- **Busca fitxers al magatzem**. Busca la llista dels fitxers indicats i retorna el subconjunt dels fitxers que ha trobat.
- **Llista els fitxers d'un directori**, en cap ordre determinat.
- **Obté el contingut d'un fitxer**. El `backimap` gestiona missatges de correu, la mida dels quals està limitada al voltant dels 10-20 MB en la majoria de servidors. Així doncs, no optimitza l'ús de la memòria.
- **Desa el contingut de diversos fitxers alhora**. En cas que un fitxer hagi d'estar en un directori concret i aquest directori no existeixi, el magatzem el crearà automàticament.
- **Elimina diversos fitxers**. Cal tenir en compte que els fitxers romandran al magatzem, tot i que no ja seran visibles després d'aquesta operació i passaran a formar part de l'històric.
- **Confirma els canvis pendents**, tot indicant el motiu del canvi. Enlloc de permetre que sigui l'operació de desar la que pugui confirmar els canvis pendents a cada crida, és més interessant oferir la possibilitat de confirmar-los explícitament en qualsevol altre moment (per exemple, confirmant cada operació de desar per a un número molt gran de fitxers el dipòsit Git creixeria excessivament).
- **Descarta els canvis pendents**. Aquesta operació és la que realitza el `backimap` quan s'indica explícitament l'opció `--clean` (per exemple, per netejar el magatzem després de qualsevol interrupció).

L'esquelet bàsic de la definició d'aquesta classe en Moose seria d'aquesta manera:

```
package App::backimap::Storage;
use Moose;
use Git::Wrapper;

has dir   => ( is => 'ro', isa => 'Str', required => 1, default => "/tmp/backimap"   );
has init  => ( is => 'ro', isa => 'Bool', default => 0                               );
has clean => ( is => 'ro', isa => 'Bool', default => 0                               );
has _git  => ( is => 'ro', isa => 'Git::Wrapper', lazy => 1, builder => '_build_git' );

sub _build_git { }

sub find   { }
sub list   { }
sub get    { }
sub put    { }
sub delete { }
sub commit { }
sub reset  { }
```

Però també cal actualitzar convenientment la definició de la classe de l'estat, tot afegint l'atribut `storage` i el mètode `save`:

```
package App::backimap::Status;
use Moose;
use MooseX::Storage;
with Storage( 'format' => 'JSON' );

has timestamp => ( is => 'ro', isa => 'Int', required => 1, default => 0         );
has server    => ( is => 'ro', isa => 'Str', required => 1, default => "server?" ;
has user      => ( is => 'ro', isa => 'Str', required => 1, default => "user?"   );
has folder    => ( is => 'rw', isa => 'HashRef[App::backimap::Status::Folder]'   );
has storage   => ( is => 'ro', isa => 'App::backimap::Storage'                   );

sub save { }
```

4.8 Purga de missatges

De cop m'adono que fins ara el `backimap` no elimina els correus de la còpia local si han estat esborrats al servidor IMAP. Però afortunadament el nou magatzem ja és plenament funcional i ens ofereix l'operació per eliminar fitxers, així que només resta per decidir com farà la purga el `backimap`. Com que es tracta d'una operació desitjable, en principi no afegim cap opció al backimap que permeti triar si es desitja fer-la o no (al cap i a la fi, seguim disposant dels històrics en cas que ens facin falta).

Amb tot plegat, refactoritzo les passes que realitza el `backimap` i la cosa queda així:

1. **Inicialitza el programa**.
   (a) **Prepara el magatzem**.
   (b) **Estableix la connexió IMAP**.
   (c) **Configura l'estat**.
2. **Copia el correu IMAP**.
   (a) **Obté la llista de carpetes**.
   (b) Per a cada carpeta…
      i. **Obté els comptadors**.
      ii. **Obté una llista dels correus que hi ha al magatzem per aquella carpeta**.
      iii. Per a cada missatge d'aquesta carpeta al servidor IMAP…
         A. **Mira si ja està al magatzem**. En cas afirmatiu salta al següent missatge, altrament segueix.
         B. **Desa el contingut del missatge al magatzem**. En aquest punt, segur que es tracta d'un missatge nou en aquesta carpeta per tant cal desar-lo.
      iv. **Purga els correus restants del magatzem**. A partir de la diferència entre la llista de correus que hi havia al magatzem prèviament i la que té actualment el servidor IMAP, el `backimap` determina quins correus li cal purgar

del magatzem. Si l'usuari ha indicat l'opció `--verbose`, es mostrarà la llista dels missatges purgats.

3. **Desa l'estat al magatzem**.

4. **Confirma els canvis pendents al magatzem**.

5. **Mostra el temps d'execució**.

4.9 UTF-7 i IMAP

El protocol IMAP [RFC 3501] defineix com cal codificar el nom de les carpetes que contenen caràcters especials utilitzant una variant no estàndard de l'Unicode, UTF-7. Amb aquesta codificació, s'obtenen cadenes de caràcters ASCII d'una mida més curta que si s'emprés UTF-8 amb la codificació QP (*quoted printable* en anglès, que consisteix en el caràcter «=» seguit del valor hexadecimal del caràcter a representar).

És important tenir-ho en compte ja que quan el `backimap` demana al servidor la llista de carpetes, si alguna d'elles conté caràcters no imprimibles en ASCII (per exemple, lletres amb accent, etc.), la rebra codificada en UTF-7 d'IMAP. Així doncs, cal trnasformar la resposta del servidor IMAP a una codificació acceptable per al client (he triat UTF-8 com a codificació per defecte ja que és la més extesa en els escriptoris moderns de programari lliure). Afortunadament, existeix el mòdul `Encode::IMAPUTF7` que ens facilita aquesta feina.

4.10 Indicació del progrés i una estimació

Quan l'usuari copia un gran volum de correu i desitja que el `backimap` li vagi mostrant per on va, pot utilitzar l'opció `--verbose` que li anirà indicant per quina carpeta passa i quants correus té (nous i en total). Però si una d'aquestes carpetes té molts missatges (de l'ordre de milers, per exemple), l'usuari segurament agrairà una pista del temps que trigarà en acabar la còpia i en quin punt es troba respecte del total a copiar.

Així doncs, en aquesta història oferim a l'usuari una barra de progrés amb una estimació del temps restant per tal que pugui anar a prendre un cafè tranquil·lament quan vegi les indicacions del `backimap`. Per a fer-ho, he decidit emprar el mòdul `Term::ProgressBar` que ofereix les dues coses alhora (barra de progrés i estimació). Un cop creada la barra de progrés, només cal que el `backimap` li indiqui quan acaba de tractar cada missatge.

```
$ backimap --dir=/backups/gmail --init --verbose --uri=imaps://nom.cognom@gmail.com@imap.gmail.com
Password: ************************
Examining folders...
 * INBOX (4/17):  11% [======                                                    ]0m07s Left
```

4.11 Canvi de nom de les carpetes IMAP

Si l'usuari ha mogut un missatge d'una carpeta a una altra al servidor IMAP, se n'adona el `backimap`? En principi, com que l'operació passa al servidor IMAP, fora del control del `backimap`, aquest no té forma de saber que s'ha produït aquest moviment. Des del seu punt de vista, el missatge haurà desaparegut de la carpeta origen i haurà aparegut a la destí, amb un identificador diferent. Però això no és problema per al magatzem, ja que el Git interpreta de la mateixa manera l'operació degut a què el contingut del missatge no varia. Així, el Git és capaç de veure que l'objecte que conté el missatge ha deixat d'estar en una banda i apareix en l'altra.

Aleshores què passa quan es produeix un canvi de nom en una carpeta IMAP? La situació és força similar: el Git veuria com un munt d'objectes han desparegut d'una banda i apareixerien en una altra. Però en aquest cas, el servidor ens ofereix una mica més d'ajuda que podem combinar amb l'estat anterior del `backimap` que havíem desat el darrer cop. Cada carpeta IMAP té un identificador únic al servidor que s'anomena *unique identifier validity* i que no canvia mai, encara que canviï el nom. Així doncs, el `backimap` només ha d'obtenir el nom i l'identificador de validesa d'una carpeta, comparar el nom amb el que tenia anteriorment i, si detecta que s'ha produït un canvi, actualitza l'estat i indica el canvi al magatzem amb el nou mètode `move`.

D'altra banda, què passa quan l'usuari elimina una carpeta IMAP? Actualment el `backimap` no realitzaria cap canvi al magatzem perquè la carpeta ja no sortiria a la llista de les que cal tractar segons el servidor IMAP. Com que eliminar definitivament una carpeta no és una operació gaire comú, l'he descartada per aquesta fase del projecte.

4.12 Exclusió de carpetes IMAP

Avui dia, pràcticament tots rebem spam per correu electrònic tot i que normalment els filtres fan que la majoria d'aquest correu quedi fora de la vista, desat en una carpeta determinada o s'esborri directament. En canvi, m'atreviria a dir que pocs usuaris voldrien tenir còpies de seguretat de les seves carpetes d'spam per tant sembla òbvia la necessitat de disposar d'una opció del `backimap` per indicar que l'usuari vol excloure una carpeta del *backup*.

Així doncs, afegim una nova opció `--exclude=NOM` que l'usuari pot utilitzar tants cops com desitgi per excloure diverses carpetes, per exemple excloent també la paperera:

```
$ backimap --dir=/backups/gmail --verbose --uri=imaps://nom.cognom@gmail.com@imap.gmail.com \
          --exclude='[Gmail]/Spam' --exclude='[Gmail]/Trash'
```

Què passaria si l'usuari indica que vol excloure una carpeta després d'haver-la copiat anteriorment al magatzem? L'exclusió no tindria cap efecte sobre les dades ja existents, atès que no es pot interpretar que l'usuari vulgui eliminar-les i per tant és millor triar l'opció més segura.

4.13 Reprendre la còpia després d'una interrupció

Quan es copien grans volums de dades per la xarxa es poden produir interrupcions, bé sigui per un temps excedit (*timeout* en anglès) en la connexió amb el servidor o per un tall de la connexió a nivell físic (avaria d'un equip de comunicacions, un cable tallat, etc.). En els casos que això passa, el magatzem queda brut (*dirty*, en la nomenclatura de Git) i l'usuari pot indicar que vol fer net dels canvis pendents amb l'opció `--clean`, tal com hem comentat anteriorment.

Però com que considerem una història molt interessant per als usuaris poder reprendre l'operació que s'havia quedat a mig fer, hem afegit una nova opció `--resume` al `backimap`:

```
$ backimap --dir=/backups/gmail --uri=imaps://nom.cognom@gmail.com@imap.gmail.com --init
^C
$ backimap --dir=/backups/gmail --uri=imaps://nom.cognom@gmail.com@imap.gmail.com --resume
```

Aquesta opció es pot utilitzar sense cap por encara que en l'anterior còpia no s'hagi produït una interrupció ja que el `backimap` sap perfectament si el magatzem està brut o no, i quan no és el cas s'estalvia de netejar inútilment.

4.14 Còpies incrementals

Una còpia incremental és aquella que comença al punt on va acabar l'anterior. Per exemple, una còpia incremental d'un sistema de fitxers inclourà tots els fitxers nous i aquells que tinguin la data de modificació posterior al moment de la darrera còpia. Per al cas que ens ocupa, disposem la marca de temps de l'estat persistent i el protocol IMAP permet fer cerques sobre els missatges que tinguin al servidor una data (*internal date* del protocol IMAP) posterior. Així doncs, afegim l'opció `--incremental` al `backimap`:

```
$ backimap --dir=/backups/gmail --uri=imaps://nom.cognom@gmail.com@imap.gmail.com \
           --verbose --incremental
```

Només cal que tinguem en compte que en aquest cas el `backimap` no podrà determinar si hi ha missatges pendents de purgar i per tant no n'eliminarà cap fins a la propera còpia total.

4.15 Descomposició dels missatges

Al març d'aquest any vaig fer una petita introducció del projecte a la comunitat dels Perl Mongers de Barcelona. Els vaig explicar les virtuds d'utilitzar Git per al magatzem de dades perquè estalvia molt d'espai quan es fan còpies del correu de Gmail. Els presents van fer preguntes i suggeriments, però n'hi ha una en concret que em va agradar força: si havia pensat en extreure els ajunts dels missatges de correu.

Després de rumiar-hi una mica més a fons i refinant aquella idea, vaig decidir que afegiria una opció `--explode` al `backimap` per tal de descomposar els missatges en les diferents parts de què estan composats:

```
$ backimap --dir=/backups/gmail --uri=imaps://nom.cognom@gmail.com@imap.gmail.com \
           --init --explode --verbose
```

El format dels missatges de correu està descrit actualment a l'estàndard [RFC 5322]. Un missatge de correu es composa de dues parts principals, la secció de les capçaleres i opcionalment el cos, que pot estar estructurat en d'altres parts que conformen un arbre. Cadascuna d'aquestes parts habitualment està identificada per un tipus MIME (*Multipurpose Internet Mail Extensions*, [RFC 2045], [RFC 2046], [RFC 2047], [RFC 2048] i [RFC 2049]), que permet una gran flexibilitat a l'hora de combinar dades de tipus molt diferents en un mateix missatge de text.

Vegeu a la Figura 4.3 l'estructura de l'exemple complex de l'apèndix A al [RFC 2049] per a il·lustrar-ho amb més detall.

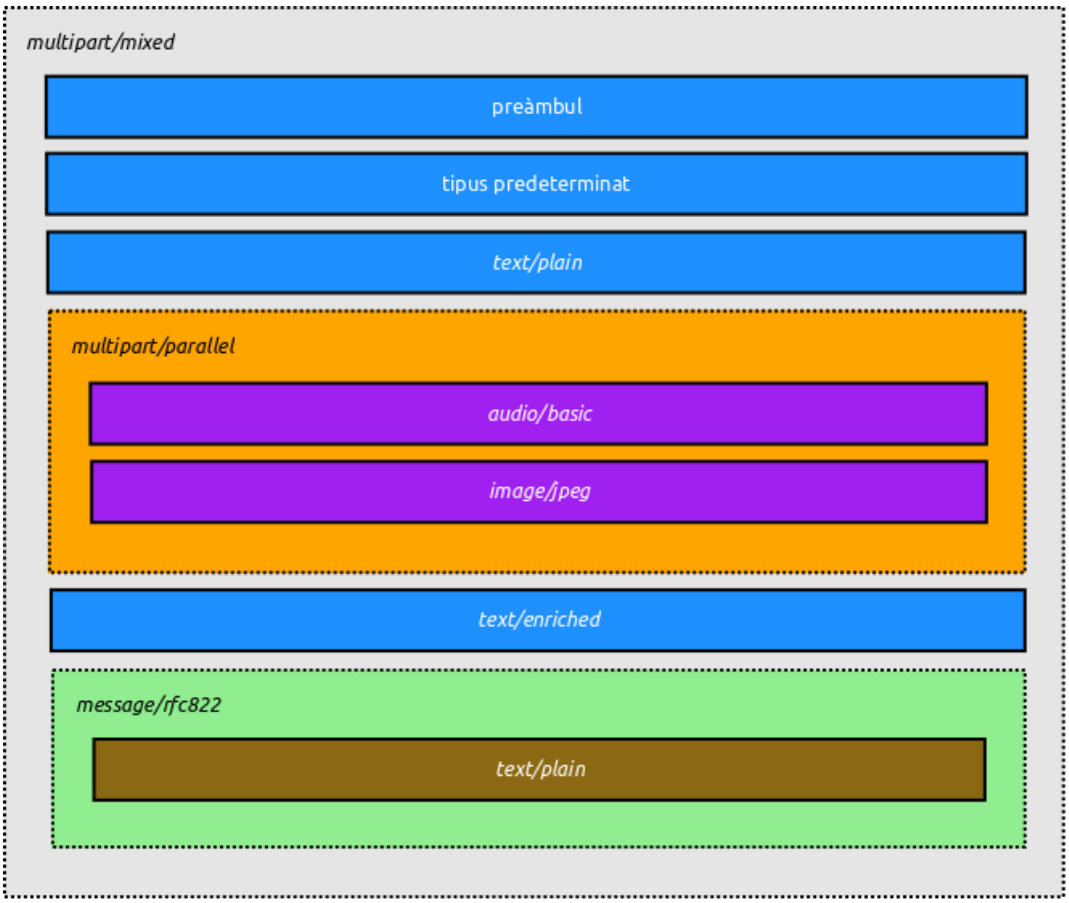


Figura 4.3: *Estructura de les parts d'un exemple complex de MIME.*

El mòdul `MIME::Parser` de Perl permet fer la descomposició d'aquestes estructures de dades i desar les diferents parts en fitxers (això és molt pràctic per als antivirus i d'altres eines que han d'analitzar el contingut dels correus a fons), tret de la secció de les capçaleres. Per a desar-les he triat un mòdul clàssic de serialització en Perl anomenat

`Storable` i he anomenat el fitxer corresponent `__MIME__` per tal de poder-lo identificar clarament d'entre la resta de parts.

Així doncs, quins avantatges tindria la descomposició dels missatges? El més evident és l'accés més ràpid i senzill als adjunts, que fins i tot permetria utilitzar eines de cerca i indexació típiques dels sistemes de fitxers tradicionals (per exemple, buscant paraules clau dins un document PDF, fent reconeixement de cares en les imatges JPEG, etc.). Per a situacions en què les còpies es realitzessin en un entorn amb molts usuaris sobre un mateix magatzem (actualment no es permet compartir un magatzem entre diferents usuaris) podria haver-hi un estalvi d'espai significatiu si aquests usuaris rebessin correus amb els mateixos adjunts (per exemple, degut a llistes de correu amb molts membres). Malauradament, l'experiència en el cas d'un sol usuari és totalment la contrària: hi ha un creixement significatiu de l'espai ocupat pel magatzem i a més la còpia triga més en finalitzar (en els casos de comptes IMAP amb molts correus, la diferència encara és més gran).

En qualsevol cas, aquesta és una àrea a seguir explorant en el futur. Sobretot per la possibilitat de treballar directament amb les dades en el seu format original, enlloc d'estar codificades en els missatges de text. Probablement en aquests casos, caldria descartar el dipòsit Git per al magatzem i substituir-lo per alguna de les solucions de magatzem que proporciona Hadoop (similars a les utilitza Google internament per als seus serveis però basades en programari lliure per a la computació distribuïda).

4.16 Tests

Per comprovar la flexibilitat que ofereix Perl a l'hora de fer tests unitaris, utilitzaré els de la classe de magatzem, que il·lustren com es proven els diferents casos d'ús.

4.16.1 Inicialització dels tests

Aquest bloc mostra com es prepara un fitxer de tests senzill per al magatzem:

```
use strict;
use warnings;
use Test::More tests => 1+9+4+3;
use Path::Class::Dir();

my $tmp_dir = Path::Class::Dir->new('t/tmp');
my $class   = 'App::backimap::Storage';
my %args    = (
    dir    => $tmp_dir,
    init   => 1,
    clean  => 0,
    resume => 0,
    author => 'me',
    email  => 'me@me.me',
);
use_ok($class);
```

4.16.2 Tests d'operacions bàsiques

Tot seguit, un cop ja inicialitzat l'entorn de test, el següent bloc efectua els tests de les operacions bàsiques que ofereix el magatzem mitjançant els mètodes de la classe:

```
{
    my $storage = $class->new(%args);
    isa_ok( $storage, $class );

    $storage->put( file1 => 'content1' );
    is( $storage->get('file1'), 'content1', 'get file after put' );
    is_deeply( [ $storage->list('/') ], [ 'file1' ], 'list files' );

    $storage->move( 'file1', 'file2' );
    is( $storage->get('file2'), 'content1', 'get file after move' );
    is_deeply( [ $storage->list('/') ], [ 'file2' ], 'list files after move' );
    is_deeply( [ $storage->find(qw( file1 file2 )) ], [ 'file2' ], 'find existing files' );
    is_deeply( [ $storage->find(qw( file1 file3 )) ], [], 'find non-existing files' );

    $storage->commit('test file2');
    $storage->delete('file2');
    is_deeply( [ $storage->list('/') ], [], 'list files after delete' );

    $storage->reset();
    is_deeply( [ $storage->list('/') ], [ 'file2' ], 'list after reset' );

    $tmp_dir->rmtree();
}
```

Per tal d'evitar conflictes de visibilitat entre variables d'altres tests, s'engloba cada bateria en un bloc separat i s'utilitzen variables lèxiques de Perl (les declarades amb `my`), que només són visibles dins del context lèxic actual. Finalment, en acabar els tests, fem dissabte i eliminem el magatzem sencer per evitar crear conflictes amb d'altres tests.

4.16.3 Test del magatzem brut

En aquest cas, per comprovar que el `backimap` efectua correctament la neteja del magatzem quan està brut, hem de preparar un seguit d'operacions que deixin el magatzem en aquest estat (només cal que afegim un fitxer al magatzem i no confirmem el canvi):

```
{
    my $storage = $class->new(%args);
    isa_ok( $storage, $class );

    $storage->put( file => 'file content' );
    $storage->commit('test file');
    $storage->put( failed => 'failed content' );

    # resume previous scheduled operation
    my $other_storage = $class->new( %args, init => 0, resume => 1 );
    isa_ok( $other_storage, $class );
```

```
    is_deeply( [ $other_storage->find('failed') ], [ 'failed' ], 'find after resume' );
    is( $other_storage->get('failed'), 'failed content', 'get after resume' );

    $tmp_dir->rmtree();
}
```

4.16.4 Test de descomposició MIME

Per acabar, us mostro un exemple trivial per comprovar que la descomposició d'un MIME funciona correctament:

```
{
    my $storage = $class->new(%args);
    isa_ok( $storage, $class );

    my $text    = "Hello, world!";
    my $content = qq{Content-Type: text/plain; charset=US-ASCII; name="hello.txt"\n\n}
                . qq{$text\n};
    $storage->explode( mime => $content );
    is_deeply(
        [ sort $storage->list("/mime") ],
        [ sort '__MIME__', 'hello.txt' ],
        'list after explode MIME',
    );
    is( $storage->get('/mime/hello.txt'), "$text\n", 'get exploded MIME content' );

    $tmp_dir->rmtree();
}
```

4.17 Opcions, ajuda i documentació

Mitjançant el rol `MooseX::Getopt` defineixo les diverses opcions de l'ordre `backimap` esmentades anteriorment. Aquest mòdul permet declarar les opcions de manera molt senzilla com a atributs de la classe principal `App::backimap`:

```
package App::backimap;
use Moose;
with 'MooseX::Getopt';
use Moose::Util::TypeConstraints;
use Encode();
use MooseX::Types::Path::Class;

subtype 'ArrayOfUtf8' => as   'ArrayRef';
coerce  'ArrayOfUtf8' => from 'ArrayRef' => via { Encode::encode( 'utf-8', $_ ) };
MooseX::Getopt::OptionTypeMap->add_option_type_to_map( 'ArrayOfUtf8' => '=s@' );

my @ExcludeOpt = ( is => 'ro', isa => 'ArrayOfUtf8',      default  => sub { [] } );
```

```
my @DirOpt     = ( is => 'ro', isa => 'Path::Class::Dir', coerce   => 1          );
my @UriOpt     = ( is => 'ro', isa => 'Str',              required => 1          );
my @BoolOpt    = ( is => 'ro', isa => 'Bool',             default  => 0          );

has exclude    => ( @ExcludeOpt, documentation => 'Folder name to exclude from backup (e.g. spam)' );
has dir        => ( @DirOpt,     documentation => 'Path to storage (default: ~/.backimap)'        );
has uri        => ( @UriOpt,     documentation => 'URI for the remote IMAP folder'                 );
has init       => ( @BoolOpt,    documentation => 'Initialize storage and setup backimap status'   );
has clean      => ( @BoolOpt,    documentation => 'Clean up storage if dirty'                      );
has resume     => ( @BoolOpt,    documentation => 'Resume previous failed backup'                  );
has incremental => ( @BoolOpt,   documentation => 'Perform an incremental backup since last time' );
has explode    => ( @BoolOpt,    documentation => 'Explode message MIME parts (e.g. attachments)'  );
has verbose    => ( @BoolOpt,    documentation => 'Enable verbose messages'                        );
```

Un aspecte interessant a destacar és com es defineix el subtipus `ArrayOfUtf8` per a l'opció `--exclude`. Com que volem que l'usuari pugui indicar aquestes opcions diversos cops per a carpetes diferents, cal mapar-les en un tipus conegut per `Moose` (en aquest cas `ArrayRef`). Després només cal indicar com es realitzarà la coerció automàtica del tipus `via` un iterador que s'aplicarà a cadascun dels elements del vector. Finalment, s'indica al rol `MooseX::Getopt` quin comportament volem que tinguin les opcions d'aquest nou subtipus `ArrayOfUtf8`.

Tot plegat, calia complicar-se tant la vida? Els avantatges d'aquestes tècniques són habituals als llenguatges de programació fortament tipats, que sovint eviten molts errors de programació i haver de posar controls sobre el tipus i forma de les dades. La diferència en aquest cas és que Perl no és un d'aquests llenguatges però ens ofereix la possibilitat de comportar-se com a tal en els casos que ho desitgem. A més de tots aquests avantatges, la tria d'aquest mòdul en té d'altres com ara la possibilitat de combinar-lo amb d'altres rols, com per exemple `MooseX::ConfigFromFile` que permet tenir les opcions preferides per l'usuari en un fitxer de configuració.

Així doncs, per obtenir un resum ràpid de les que disposa ja podem utilitzar l'opció `--help` (que ve inclosa de regal amb el rol `MooseX::Getopt`):

```
$ backimap --help
Required option missing: uri
usage: backimap [-?] [long options...]
    -? --usage --help  Prints this usage information.
    --uri              URI for the remote IMAP folder
    --exclude          Folder name to exclude from backup (e.g. spam)
    --dir              Path to storage (default: ~/.backimap)
    --init             Initialize storage and setup backimap status
    --clean            Clean up storage if dirty
    --resume           Resume previous failed backup
    --incremental      Perform an incremental backup since last time
    --explode          Explode message MIME parts (e.g. attachments)
    --verbose          Enable verbose messages
```

D'altra banda, cada mòdul del projecte disposa de documentació estàndard en format POD de Perl que serveix per tant per a desenvolupadors com per a usuaris. Aquest

format permet crear manuals d'usuari sencers i es pot barrejar amb el codi (documentant cada funció, mètode i atribut per separat). Però també es pot deslligar totalment posant-la al principi, al final o en un fitxer a banda amb l'extensió `.pod`.

Per acabar, cal destacar que la documentació en POD es publica automàticament al CPAN quan un autor hi publica el codi del seu projecte. Per exemple, per al cas del magatzem del `backimap` la documentació es pot llegir a App::backimap::Storage.

4.18 Instal·lació

La instal·lació del `backimap` es pot realitzar com la de qualsevol altre mòdul de Perl, només cap indicar que es vol instal·lar el mòdul `App::backimap`. D'altra banda, si no coneixeu cap de les eines típiques per instal·lar-ho, podeu seguir les instruccions següents en una distribució Debian o Ubuntu, que us preparan un directori `$HOME/backimap` on podreu instal·lar el `backimap` amb totes les dependències necessàries:

```
$ sudo apt-get install curl build-essential libssl-dev libxml2-dev git-core
$ mkdir -p $HOME/backimap/bin
$ curl --location http://cpanmin.us > $HOME/backimap/bin/cpanm
$ chmod +x $HOME/backimap/bin/cpanm
$ $HOME/backimap/bin/cpanm --local-lib=$HOME/backimap local::lib
$ echo 'eval $(perl -I$HOME/backimap/lib/perl5 -Mlocal::lib=$HOME/backimap)' >> ~/.bashrc
```

A partir d'aquest moment, en les noves sessions de terminal que obriu tindreu disponible l'entorn adient per fer la instal·lació (o l'actualització a noves versions) del `backimap`, només caldrà que executeu aquesta ordre:

```
$ cpanm App::backimap
```

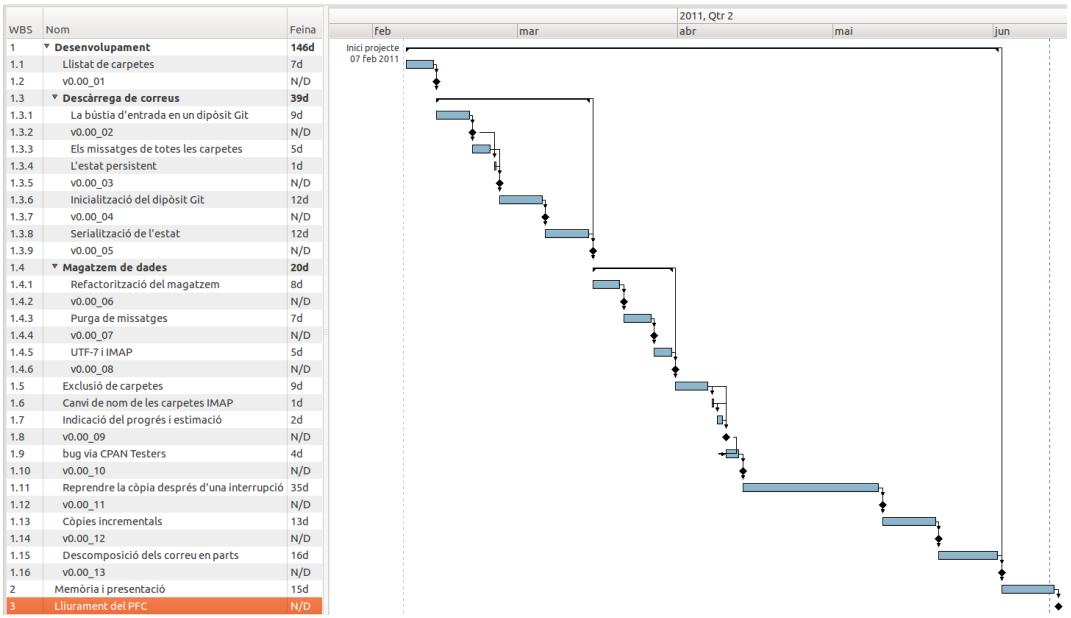
Capítol 5

Valoracions

Abans de tancar aquesta primera fase del projecte *backimap* voldria fer un seguit de valoracions que crec que són importants per poder avaluar la seva execució i fer-se una idea més clara sobre els resultats del projecte.

5.1 Cronologia

Tal com he explicat a la introducció, la planificació d'aquest projecte s'ha centrat en cicles de desenvolupament d'entre una i dues setmanes. Al final de cada cicle, he publicat una versió de desenvolupament del `backimap` al CPAN per tal que els CPAN Testers puguessin començar a fer els tests automàtics en diferents entorns de Perl.



WBS	Nom	Feina
1	**Desenvolupament**	**146d**
1.1	Llistat de carpetes	7d
1.2	v0.00_01	N/D
1.3	**Descàrrega de correus**	**39d**
1.3.1	La bústia d'entrada en un dipòsit Git	9d
1.3.2	v0.00_02	N/D
1.3.3	Els missatges de totes les carpetes	5d
1.3.4	L'estat persistent	1d
1.3.5	v0.00_03	N/D
1.3.6	Inicialització del dipòsit Git	12d
1.3.7	v0.00_04	N/D
1.3.8	Serialització de l'estat	12d
1.3.9	v0.00_05	N/D
1.4	**Magatzem de dades**	**20d**
1.4.1	Refactorització del magatzem	8d
1.4.2	v0.00_06	N/D
1.4.3	Purga de missatges	7d
1.4.4	v0.00_07	N/D
1.4.5	UTF-7 i IMAP	5d
1.4.6	v0.00_08	N/D
1.5	Exclusió de carpetes	9d
1.6	Canvi de nom de les carpetes IMAP	1d
1.7	Indicació del progrés i estimació	2d
1.8	v0.00_09	N/D
1.9	bug via CPAN Testers	4d
1.10	v0.00_10	N/D
1.11	Reprendre la còpia després d'una interrupció	35d
1.12	v0.00_11	N/D
1.13	Còpies incrementals	13d
1.14	v0.00_12	N/D
1.15	Descomposició dels correu en parts	16d
1.16	v0.00_13	N/D
2	Memòria i presentació	15d
3	Lliurament del PFC	N/D

Figura 5.1: *Cronologia del projecte.*

Com es pot veure a la Figura 5.1, en general he pogut respectar el termini dels cicles de desenvolupament tret del cicle de la versió v0.00_11, que ha durat el doble. Curiosament, la Pasqua d'enguany va coincidir en aquest mateix període però els motius principals d'aquest desviament han estat uns altres.

D'una banda, vaig malinterpretar el requeriment de disposar d'un mecanisme per gestionar el format de les dates del protocol IMAP i vaig crear un nou mini-projecte per a resoldre-ho (el mòdul DateTime::Format::RFC3501 del CPAN), però finalment no va resultar necessari per al `backimap`. De l'altra, vaig estar una setmana barallant-me amb un problema dels mètodes constructors de les classes de l'estat i del magatzem, que va resultar ser un *bug* el mòdul Test::TestCoverage, però vaig poder ajudar l'autor a resoldre'l amb un test de demostració i a corregir la solució que va publicar inicialment amb una nova versió corregida del mòdul de test.

5.2 Avaluació dels riscos

Tot seguit faré un repàs als riscos que s'han produït durant aquesta primera fase del projecte i explicaré quin impacte han tingut i com he mirat de posar-hi remei.

3. **Desconeixement, poca experiència o motivació en les eines**

   Durant el desenvolupament de l'opció per a descomposar els missatges en parts m'he trobat amb una situació associada a aquest risc: vaig valorar la serialització de les capçaleres dels missatges amb el mateix mecanisme que utilitzo per a l'estat, el rol `MooseX::Storage`. Després de barallar-me i estudiar com s'implementa la serialització de dades en aquest rol, vaig realitzar algunes proves sense èxit amb el mòdul `MooseX::NonMoose` (permet crear subclasses `Moose` des d'una classe que no ho és). Malauradament vaig acabar descobrint que no es pot aplicar un rol de Moose a objectes que no ofereixen el protocol de meta-objectes (MOP), com és el cas dels objectes `MIME::Head` que construeix el mòdul `MIME::Parser`. Caldria crear dinàmicament una subclasse de `MIME::Head` amb `MooseX::NonMoose` que indiqués amb el MOP quins són els atributs, els mètodes, etc. En tot cas, seria una opció a considerar en el futur però a curt termini vaig preferir serialitzar les capçaleres amb un altre mòdul que no depèn del MOP, `Storable`.

5. **Problemes amb els mòduls d'altres autors**

   Al llarg del projecte he trobat tres errors en diversos mòduls que utilitza el `backimap` i sobre els quals he informat als autor corresponents:

   - Un error al mètode `secure` del mòdul `URI::imaps`, CPAN #65679. L'impacte d'aquest error ha estat molt petit ja que, gràcies a la sobrecàrrega de mètodes, he pogut rescriure el mètode `secure` perquè funcioni correctament mentre l'autor no decideixi si accepta la meva proposta de solució.
   - Errors en l'ús dels modificadors de mètode per als objectes `Moose` dins el mòdul `Test::TesCoverage`, CPAN #67862. En aquest cas, l'impacte sobre la durada del cicle ha estat considerable i ha arribat a duplicar el temps necessari per tancar-lo. L'esforç més important va ser trobar el motiu dels errors, un cop determinat que era aquest mòdul vaig informar-ne l'autor i vaig seguir avançant sense utilitzar-lo.
   - Un avís sobre les versions durant la construcció d'objectes `Moose` que han estat serialitzats amb el rol `MooseX::Storage`, CPAN #68358. Aquest problema no

oferia cap conflicte en el comportament del `backimap` atès que es tractava d'un avís i prou. Però tractant-se d'un avís que podria confondre als usuaris, vaig decidir dedicar el temps suficient per esbrinar com desactivar l'avís després d'informar del problema a l'autor.

6. **Dependència de proveïdors de serveis externs**

   - Allò que era poc probable que passés va passar: al mes de febrer vaig patir talls intermitents de la connexió ADSL. Encara no havia comprat el portàtil però aquest va ser un factor determinant per decidir-me a fer-ho. Afortunadament, també tinc un mòdem GSM que permet connectar qualsevol ordinador per USB a la xarxa 3G i, tot i la incomoditat de la xarxa més lenta, no va ser un impediment important per al projecte.
   - Després, al maig els mantenidors del servidor *alioth* del projecte Debian, on hi ha hostatjat l'equip de desenvolupament de Perl de Debian, van decidir actualitzar i modernitzar el servidor. Me'n vaig adonar després de comprovar que els tests de qualitat del `backimap` es quedaven congelats sempre al mateix punt (buscaven informació al servidor *alioth*). Un cop trobat el problema i confirmat a través del xat, vaig descartar seguir fent els tests de qualitat fins que acabessin l'actualització i vaig suggerir l'autor del mòdul de test de qualitat que distribueixin la informació des del propi CPAN (disposa de miralls per tot el món), CPAN #68398.

7. **Imponderables IRL**

   Finalment, era previsible que apareguessin imponderables que no he pogut declinar o delegar durant aquest període: tres casaments en quatre setmames entre els mesos de maig i juny, la col·laboració i diverses entrevistes per a realitzar el reportatge Ciberactivistes de TV3 entre els mesos de març i maig i finalment, aquest mateix mes de juny, la recuperació de la web del Grup Excursionista d'Oliana, que havia deixat de funcionar i de la que m'he hagut de fer càrrec sense cap transferència de coneixements. Afortunadament, tot i l'impacte d'aquests imponderables en la meva dedicació al projecte, crec que he sabut trobar la mesura justa i imprescindible.

5.3 Valoració econòmica

En aquesta valoració econòmica, no tinc en compte el rol del consultor ni el del client, ja que l'esforç majoritari d'aquest projecte ha recaigut en el rol del desenvolupador. Tampoc s'ha realitzat un seguiment de l'esforç al detall però es pot resumir d'aquesta manera:

- 146 dies de projecte equivalen aproximadament a 21 setmanes,
- 3 dies de treball a la setmana suposen 63 dies,
- 4 hores de treball al dia sumen un total de 252 hores.

Com a desenvolupador he dividit el meu temps de la forma següent (tots els preus són sense l'IVA):

- 25% de gestió del projecte (70 €/h): 4.410,00 €
- 15% al disseny (55 €/h): 2.079,00 €
- 60% a la programació (35 €/h): 5.292,00 €

Així doncs, tenint en compte a més a més els 1.256,59 € del cost del portàtil, el cost final d'aquesta primera fase del projecte seria de **13.037,59 €**, sense l'IVA.

Capítol 6

Conclusions

Després de les valoracions generals sobre el projecte és el moment d'extreure algunes conclusions. He començat aquest projecte amb la il·lusió que era l'encertat, tant per la tria del tema com per les metodologies i eines que vaig decidir utilitzar de bon començament. De fet, estic convençut que han estat la clau del que considero és un projecte molt satisfactori. La valoració personal que en faig és molt positiva ja que he tingut la possibilitat d'aprendre més a fons les metodologies que m'interessen i aplicar-les en la mesura que m'ha estat possible al projecte amb bons resultats, encara que tinc ben present que em falta més experiència per dominar-les. Aquest projecte representava un repte perquè barreja molts temes alhora i he tingut la sort de poder dedicar el temps que ha calgut a tots ells, tot i alguns entrebancs. Però el més important és que he aconseguit construir una eina que ha estat útil des del començament, que amb el temps ha anat millorant gràcies al disseny incremental i que encara li queda camí per recórrer. Això fa que estigui molt engrescat amb aquest projecte i la meva motivació per a seguir-lo tirant endavant amb la resta de fases fa que sorgexin noves idees cada dia, garantint la seva continuïtat. En resum, em considero molt afortunat i agraeixo aquesta oportunitat… :)

Capítol 7

Glossari

- ***backup***: [anglès] còpia de seguretat.
- ***blob***: [de l'anglès *binary large object*] un objecte binari gran (una fotografia en format JPEG, una manual d'usuari en PDF, etc.).
- **bústia d'entrada**: magatzem on es desa el correu electrònic entrant.
- **carpeta**: en un servidor IMAP, el nom d'un contenidor de missatges.
- **cicle de desenvolupament**: període curt i repetitiu en què s'estableixen uns objectius a assolir en la finalització del cicle, donant lloc a una nova versió funcional del producte a cada iteració del cicle.
- **correu web**: eina que funciona en un servidor web per a llegir i enviar correu mitjançant un navegador.
- **correu brossa**: correu no desitjat o que conté publicitat enganyosa, intents de robatori de dades, etc. Els correus amb virus sovint també es poden considerar correu brossa.
- ***delta***: [terme matemàtic] diferència entre dos objectes.
- **dipòsit**: [en anglès *repository*] magatzem de dades que utilitzen els sistemes de control de versions per a cada projecte. Per extensió, també s'anomena dipòsit a les còpies de treball dels dipòsits.
- **disseny incremental**: procés de disseny dividit en cicles iteratius, que evita fer prediccions a llarg termini.
- **Git**: un sistema de control de versions distribuït, modern i molt ràpid, que és capaç de gestionar molts fitxers i desa els objectes al dipòsit en forma de *blobs*.
- ***Gmail***: [contracció de *Google Mail*] serveis de correu de Google.
- **història**: la descripció d'un canvi que fa un usuari per interès propi o un desenvolupador que el negocia amb ell.
- **identificador únic**: en un servidor IMAP, un número que identifica un missatge de forma única dins una carpeta en aquell servidor.

- **identificador de validesa**: en un servidor IMAP, un número que identifica una carpeta del servidor.
- **IMAP**: [*Internet Mail Application Protocol*] protocol d'accés remot a un servidor de correu.
- ***inbox***: [anglès] bústia d'entrada. En el protocol IMAP, el nom `INBOX` identifica sempre la bústia d'entrada.
- **JSON**: [*Javascript Object Notation*] sintaxi de descripció d'objectes en el llenguatge Javascript.
- **metodologies àgils**: metodologies de desenvolupament que fomenten el disseny incremental i els cicles de desenvolupament.
- ***mixins***: en programació orientada a objectes, una classe que defineix una determinada funcionalitat que d'altres classes poden utilitzar mitjançant l'herència.
- **MOP**: protocol de meta-objectes.
- **protocol de meta-objectes**: intèrpret obert i extensible de la semàntica d'un programa, que permet crear i modificar les classes i els objectes en qualsevol moment.
- **rol**: unitat de codi que defineix un comportament reutilitzable per d'altres unitats, sense establir cap relació jeràrquica entre elles.
- **serialització**: procés de convertir una estructura de dades o un objecte a un format que es pot desar o transmetre per la xarxa, que pot ser utilitzat més tard per reconstruir un clon de l'objecte original.
- **sistema de control de versions**: mecanisme de gestió dels canvis dels documents d'un projecte (codi, gràfics, manuals, etc.) amb històric de versions i bitàcola de canvis.
- ***spam***: [anglès] correu brossa.
- **quota**: límit màxim d'un recurs imposat a un usuari per evitar que exhaureixi tots els recursos d'un sistema.
- **TLS/SSL**: [*Transport Layer Security/Secure Socket Layer*] protocol de xifrat de connexions per transmissió segura de dades entre dos punts.
- ***timeout***: temps excedit.
- ***traits***: model conceptual de programació per a definir conjunts de mètodes en programació orientada a objectes.
- **URI**: [*Uniform Resoure Identifier*] cadena de caràcters que permet identificar un nom o un recurs a Internet.
- **URL**: [*Uniform Resource Locator*] un tipus d'URI que permet localitzar un element.
- ***URL encode***: codificació de les URL per als caràcters de més de 7 bits, no imprimibles o considerats especials.

Capítol 8

Bibliografia

8.1 Llibres

- Beautiful Testing

  Tim Riley, Adam Goucher. © 2010 O'Reilly Media, Inc. (978-0596159818)

- Extreme Programming Pocket Guide

  chromatic. © 2003 O'Reilly Media, Inc. (0596004850)

- Getting Things Done

  David Allen. © 2001 David Allen. (978-0142000281)

- Getting Real

  Jason Fried, David Heinemeier Hansson, Matthew Linderman. © 1999-2006 37signals, LLC. (978-0578012810)

- Modern Perl

  chromatic. © 2010 chromatic. (978-0977920150)

- Perl Best Practices

  Damian Conway. © 2005 O'Reilly Media, Inc. (0596001738)

- Perl Testing: A Developer's Notebook

  Ian Langworth, chromatic. © 2005 O'Reilly Media, Inc. (0596100922)

- Pro Git

  Scott Chacon. © 2009 Scott Chacon. (978-1430218333)

- Programming Perl, Third Edition

  Larry Wall, Tom Christiansen, Jon Orwant. © 1991, 1996, 2000 O'Reilly Media, Inc. (978-0596000271)

- The Art of Agile Development

  James Shore, Shane Warden. © 2008 James Shore, chromatic. (978-0596527679)

- The Art of Community

  Jono Bacon. © 2009 Jono Bacon. (978-0596156718)

- Version Control with Git

  Jon Loeliger. © 2009 Jon Loeliger. (978-0596520120)

8.2 Protocols i estàndards

- RFC 3501 — *Internet Message Access Protocol, Version 4rev1*
- RFC 5322 — *Internet Message Format*
- RFC 2045 — *MIME Part One: Format of Internet Message Bodies*
- RFC 2046 — *MIME Part Two: Media Types*
- RFC 2047 — *MIME Part Three: Message Header Extensions for Non-ASCII Text*
- RFC 2048 — *MIME Part Four: Registration Procedures*
- RFC 2049 — *MIME Part Five: Conformance Criteria and Examples*

8.3 Eines

- App::perlbrew — *Manage perl installations in your $HOME*
- App::cpanminus — *get, unpack, build and install modules from CPAN*
- Dist::Zilla — *distribution builder; installer not included!*
- Tracks — *a web-based application to help you implement Getting Things Done*
- Dia — *a GTK+ based diagram creation program*
- Planner — *the GNOME project management tool*