Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων Και Μηχανικων Υπολογιστων

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Αναζήτηση και υπηρεσίες εξατομίκευσης σε ένα κατανεμημένο δίκτυο συνεργαζόμενων ψηφιακών βιβλιοθηκών Φυσικής Υψηλής Ενέργειας: αναβάθμιση του λογισμικού του CERN Document Server

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Νικολάου Κ. Κασιούμη

Επιβλέπων: Παναγιώτης Δ. Τσανάκας
Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2010

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Αναζήτηση και υπηρεσίες εξατομίκευσης σε ένα κατανεμημένο δίκτυο συνεργαζόμενων ψηφιακών βιβλιοθηκών Φυσικής Υψηλής Ενέργειας: αναβάθμιση του λογισμικού του CERN Document Server

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Νικολάου Κ. Κασιούμη
Α.Μ. 03101155

Επιβλέπων: Παναγιώτης Δ. Τσανάκας
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 26 Φεβρουαρίου 2010

....	....	....
Παναγιώτης Δ. Τσανάκας	Νεκτάριος Κοζύρης	Διονύσιος-Δημήτριος Κουτσούρης
Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2010

....

Νικόλαος Κ. Κασιούμης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Το CERN, ο ευρωπαϊκός οργανισμός πυρηνικών ερευνών, δραστηριοποιείται εδώ και χρόνια στη διάδοση των ερευνητικών του αποτελεσμάτων. Για το σκοπό αυτό, αναπτύσσει και χρησιμοποιεί το λογισμικό CDS Invenio, ένα πλήρες σύστημα ψηφιακής βιβλιοθήκης για την αποθήκευση, συντήρηση, διαχείριση και διάθεση του έργου του. Το λογισμικό βελτιώνεται και εμπλουτίζεται συνεχώς, ώστε να ανταποκρίνεται στις βιβλιογραφικές ανάγκες των ερευνητών και των επιστημόνων. Ένας από τους σύγχρονους στόχους του είναι η δυνατότητα συνεργασίας μεταξύ αντίστοιχων ψηφιακών βιβλιοθηκών, οργανισμών κοινών ερευνητικών ενδιαφερόντων. Στα πλαίσια μιας τέτοιας συνεργασίας προβλέπεται η δυνατότητα αναζήτησης και υπηρεσιών εξατομίκευσης σε ένα κατανεμημένο δίκτυο συνεργαζόμενων ψηφιακών βιβλιοθηκών.

Στην παρούσα εργασία εξετάσαμε καταρχήν εκτενώς τις αρχές λειτουργίας και τα χαρακτηριστικά του λογισμικού CDS Invenio. Με βάση αυτά, προτείναμε και υλοποιήσαμε μια σχεδίαση η οποία επιτρέπει την αναζήτηση και τη χρήση υπηρεσιών εξατομίκευσης σε ένα κατανεμημένο δίκτυο συνεργαζόμενων ψηφιακών βιβλιοθηκών. Περιγράψαμε αναλυτικά όλες τις απαραίτητες αλλαγές και προσθήκες για την επίτευξη του σκοπού μας. Στον πυρήνα της σχεδίασής μας βρίσκονται οι φιλοξενούμενες συλλογές, οι οποίες παραπέμπουν σε διαφορετικές συλλογές εγγραφών των μελών του κατανεμημένου δικτύου. Μια σειρά συναρτήσεων αναλαμβάνουν την αναζήτηση στις συλλογές αυτές μέσω διαδικτυακών πρωτοκόλλων επικοινωνίας, προσκομίζοντας και ακολούθως αναλύοντας τα δεδομένα. Με παρόμοιο τρόπο πραγματοποιείται και η υλοποίηση βασικών υπηρεσιών εξατομίκευσης για τις φιλοξενούμενες συλλογές, όπως είναι η δημιουργία προσωπικών συλλογών και οι προσωπικές ειδοποιήσεις αναζήτησης.

Στην πορεία της εργασίας αντιμετωπίσαμε κυρίως το αναμενόμενο πρόβλημα της αβεβαιότητας στη διαθεσιμότητα πόρων του δικτύου, καθώς και των εμμέσων καθυστερήσεων που εισάγει, διατηρώντας την ταχύτητα απόκρισης του λογισμικού σε ιδιαίτερα αποδεκτά επίπεδα. Αντιμετωπίσαμε επίσης το έμφυτο μειονέκτημα της κατανεμημένης αναζήτησης, την έλλειψη δηλαδή πλήρης διαχείρισης των πόρων κάθε μέλους του κατανεμημένου δικτύου.

Καταλήξαμε στο συμπέρασμα ότι η προταθείσα αρχιτεκτονική αποτελεί μια ολοκληρωμένη λύση αναζήτησης και υπηρεσιών εξατομίκευσης σε ένα κατανεμημένο δίκτυο ψηφιακών βιβλιοθηκών, προσφέροντας σε μεγάλο βαθμό ποιότητα εφάμιλλη αυτής μιας αυτόνομης ψηφιακής βιβλιοθήκης. Δέχεται επίσης επεκτάσεις και βελτιώσεις, που μπορούν να αποτελέσουν αντικείμενο μελλοντικών εργασιών.

Λέξεις κλειδιά: ψηφιακή βιβλιοθήκη, κατανεμημένη αναζήτηση, υπηρεσίες εξατομίκευσης, συνεργαζόμενες ψηφιακές βιβλιοθήκες, κατανεμημένο δίκτυο, CERN, CDS Invenio

Abstract

CERN, the European Organization for Nuclear research, has been involved for years with the open dissemination of scientific research results. To this end, the CDS Invenio software is being developped and used as a complete digital library system for the storage, preservation, management and distribution of CERN's scientific work. The software is being constantly refined and improved in order to meet the bibliographic needs of researchers and scientists alike. One of the its current objectives is the collaboration of respective digital libraries, maintained by organizations of similar research interests. Within such a collaboration, searching and personalization services on a distributed network of co-operative digital libraries are being planned.

In this thesis, we first examined in depth the basic principles and features of CDS Invenio. On this basis, we made a proposal and its respective implementation that allows for searching and personalization services on a distributed network of co-operative digital libraries. We thoroughly described all the necessary adjustments and additions in achieving our goal. Hosted collections are the core of our design, which refer to different record collections belonging to the members of the distributed network. A range of classes and functions handle the searching of these collections fetching the data, using the hypertext transfer protocol, and then analysing it. The personalization services, such as personal record collections and personalized notifications about newlly added records, are implemented in a similar way for hosted collections.

In the course of our work we addressed the presumed uncertainty of network resources' availability and respective network delays, by maintaining a highly acceptable response time. We also dealt with the inherent drawback of distributed searching, namely the lack of control over the different indexing and ranking capabilities of each member of the distributed network.

We concluded that the proposed architecture is an integrated solution for searching and personalization services on a distributed network fo co-operative digital libraries, offering commensurate quality with the one of an autonomous digital library. It is also open to extensions and improvements, which may be the subject of future work.

Keywords: digital library, distributed searching, personalization services, co-operative digital libraries, distributed network, CERN, CDS Invenio

Ευχαριστίες

Η διπλωματική αυτή εργασία πραγματοποιήθηκε σχεδόν εξ ολοκλήρου στο ερευνητικό κέντρο του CERN, στη Γενεύη της Ελβετίας, κατά το έτος 2009 υπό την επίβλεψη των Tibor Simko και Jean-Yves LeMeur. Τους ευχαριστώ θερμά για τη βοήθειά τους, την εμπιστοσύνη τους, την καθοδήγηση και υποστήριξή τους καθ' όλο το διάστημα της εκεί εργασίας και παραμονής μου, σε ένα κλίμα φιλικό και ευχάριστο που παράλληλα σε εμπνέει και σου δίνει πρωτοβουλία.

Ολόθερμες ευχαριστίες θα ήθελα να εκφράσω στον επιβλέποντα καθηγητή μου Παναγιώτη Τσανάκα, για την ανάθεση του θέματος, την ενθάρρυνση και τη στήριξή του να εργαστώ στο CERN, την αμέριστη εμπιστοσύνη του και όλη τη βοήθεια για την ολοκλήρωση της διπλωματικής μου εργασίας. Ένα μεγάλο ευχαριστώ και στον καθηγητή Νεκτάριο Κοζύρη για τη σημαντική βοήθεια και συμπαράστασή του όποτε τη χρειάστηκα.

Ένα ευχαριστώ στον υποψήφιο διδάκτορα Κωνσταντίνο Κωτσοκάλη, ο οποίος με στήριξε απλόχερα όταν χρειάστηκα τις συμβουλές του.

Ευχαριστώ πολύ όλους τους φίλους και συναδέλφους μου στο CERN για την εξαιρετική συνεργασία μας και το ευχάριστο κλίμα στη δουλειά μας. Ιδιαίτερα δε ευχαριστώ τους Samuele Kaplun και Jerome Caffaro για την πολύτιμη και συνεχή βοήθειά τους.

Ευχαριστώ ακόμη όλους τους φίλους και συνεργάτες από τον Εθνικό Μετσόβιο Πολυτεχνείο για την πολύπλευρη στήριξή τους, ιδιαίτερα την Αλεξάνδρα Τσανάκα.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου για την αγάπη και την υπομονή της.

Acknowledgements

This thesis was almost entirely carried out at CERN, in Geneva, Switzerland, in 2009, under the supervision of Tibor Simko and Jean-Yves Le Meur. I thank them both warmly for their help, trust, guidance and support throughout my entire stay and work there, in a very pleasant and friendly environment, that inspires you and gives you initiative.

I would like to give a very warm thank you to my supervisor Panayiotis Tsanakas, for the assignment of the thesis, his encouragement and support for my work at CERN and his full trust and assistance for the completion of this thesis. A big thank you to professor Nektarios Kozyris as well for the significant help and support he gave me, whenever I needed it.

A thank you to PhD Candidate Constantinos Kotsokalis for his generous advice when I asked for it.

I would like to thank all my friends and colleagues at CERN for our excellent collaboration and the pleasant atmosphere at work. Particularly, Samuele Kaplun and Jerome Caffaro for their valuable and continuous assistance.

I would also like to thank all my friends and colleagues at the NTU of Athens for their multilateral support, particularly Alexandra Tsanaka.

Finally, I would like to thank my family for their love and patience.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή

1.1 Αντικείμενο της εργασίας

Βρισκόμαστε σε μια εποχή έντονης αναζήτησης και έρευνας σε διάφορα επιστημονικά πεδία από πολλούς ακαδημαϊκούς και επιστημονικούς οργανισμούς. Έντονη είναι και η ανάγκη αποθήκευσης των αποτελεσμάτων αυτών των ερευνών και του έργου των διαφόρων οργανισμών καθώς και διατήρηση και διάθεση τους. Στην εποχή μας, οι περιορισμοί της φυσικής αποθήκευσης και διαχείρισης όλου αυτού του υλικού έχουν ξεπεραστεί και έχουν δώσει τη σκυτάλη σε σύγχρονες ψηφιακές λύσεις. Έτσι έχουν γεννηθεί και αναπτύσσονται οι ψηφιακές βιβλιοθήκες, συστήματα λογισμικού δηλαδή, τα οποία ως σκοπό έχουν να προσφέρουν όλες τις υπηρεσίες που προσέφερε μια κλασική βιβλιοθήκη, καθώς και επιπλέον υπηρεσίες που μια κλασική βιβλιοθήκη αδυνατούσε να προσφέρει.

Ένα παράδειγμα επιστημονικού οργανισμού, που παράγει πλήθος σημαντικών επιστημονικών εγγράφων ετησίως, αποτελεί το CERN, το οποίο εστιάζει τις έρευνές του στην φυσική υψηλών ενεργειών. Στα πλαίσια των αναγκών αποθήκευσης, συντήρησης, διαχείρισης και διάθεσης των εγγράφων αυτών το CERN έχει αναλάβει την ανάπτυξη ενός τέτοιου συστήματος ψηφιακής βιβλιοθήκης, του CDS Invenio.

Παράλληλα υπάρχουν διάφοροι ακόμη αντίστοιχοι οργανισμοί και ινστιτούτα ανά τον κόσμο, που πραγματοποιούν παρόμοια έρευνα και έχουν παρόμοιες απαιτήσεις ψηφιακής οργάνωσης του έργου τους. Στα πλαίσια της συνεργασίας μεταξύ αυτών των οργανισμών γεννιέται η ιδέα μιας από κοινού κεντρικής πρόσβασης στο έργο τους. Στόχος μια τέτοιας συνεργασίας είναι η ευκολία πρόσβασης στην πληροφορία ενώ συγχρόνως διατηρείται η ανεξαρτησία μεταξύ των διαφόρων μελών της.

Για τη συνεχή επέκταση και βελτίωση του παραπάνω λογισμικού, CDS Invenio, απασχολούνται στο CERN διάφοροι έμπειροι και νέοι επιστήμονες και μηχανικοί. Στις προσπάθειες αυτές συμπεριλαμβάνεται και η παρούσα εργασία, στην οποία αναλύεται το λογισμικό CDS Invenio και περιγράφεται μια προταθείσα σχεδίαση και η υλοποίησή της, για την αναζήτηση και τις υπηρεσίες εξατομίκευσης σε ένα κατανεμημένο δίκτυο συνεργαζόμενων ψηφιακών βιβλιοθηκών, βασισμένες σε αυτό το λογισμικό.

Στην επόμενη ενότητα περιγράφεται η οργάνωση του συνολικού κειμένου.

1.2 Οργάνωση του κειμένου

Η παρούσα εργασία αποτελείται από επτά κεφάλαια. Σε αυτά μελετώνται και αναλύονται καταρχήν οι υπάρχουσες τεχνολογίες με βάση τις οποίες πραγματοποιήθηκε η εργασία, στη συνέχεια περιγράφονται η σχεδίαση και η υλοποίηση των προτεινόμενων αλλαγών, και τέλος γίνεται μια αναφορά στην τρέχουσα αξιοποίησή τους, καθώς και στις μελλοντικές τους επεκτάσεις. Ειδικότερα:

Στο πρώτο κεφάλαιο γίνεται μια εισαγωγή του αναγνώστη στο αντικείμενο της εργασίας.

Στο δεύτερο κεφάλαιο παρουσιάζονται κάποιες εισαγωγικές έννοιες σχετικές με τις ψηφιακές βιβλιοθήκες καθώς και κάποια παραδείγματα συστημάτων λογισμικού ψηφιακών βιβλιοθηκών.

Στο τρίτο κεφάλαιο περιγράφεται αναλυτικά το λογισμικού του εξυπηρετητή αρχείων του CERN, τα χαρακτηριστιά του και οι σχετικές με αυτόν τεχνολογίες.

Στο τέταρτο και πέμπτο κεφάλαιο παρουσιάζονται η μεθοδολογία και η υλοποίηση της εργασίας σχετικά με την αναζήτηση και τις υπηρεσίες εξατομίκευσης στο λογισμικό της ψηφιακής βιβλιοθήκης CDS Invenio. Ειδικότερα, στην ενότητα της μεθοδολογίας αναλύεται ο υπάρχων τρόπος λειτουργίας και η προταθείσα σχεδίαση για την επίτευξη των στόχων της εργασίας, ενώ στην ενότητα της υλοποίησης περιγράφονται τα κυριότερα σημεία εφαρμογής της προτεινόμενης σχεδίασης.

Τέλος, στο έκτο κεφάλαιο δίνονται τα συμπεράσματα της εργασίας, γίνεται αναφορά στην αξιοποίησή της στην πράξη και προτείνονται κάποιες μελλοντικές επεκτάσεις.

Κεφάλαιο 2

Οι Ψηφιακές Βιβλιοθήκες στην Έρευνα της Σωματιδιακής Φυσικής

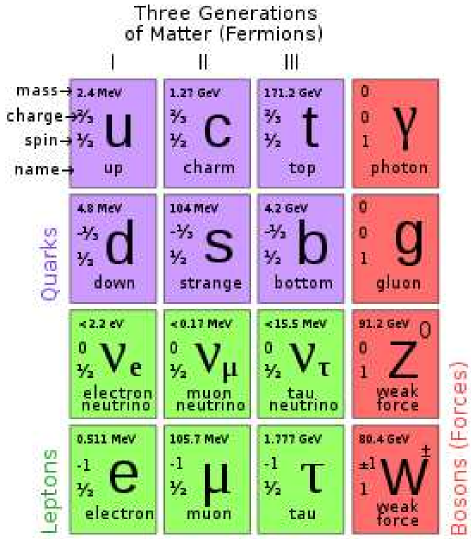
2.1 Σωματιδιακή φυσική ή Φυσική των στοιχειωδών σωματιδίων

Η Σωματιδιακή φυσική ή Φυσική των στοιχειωδών σωματιδίων είναι ένας κλάδος της Φυσικής που μελετά τα στοιχειώδη σωματίδια της ύλης και την ακτινοβολία τους, καθώς και τις αλληλεπιδράσεις τους. Είναι αλλιώς γνωστή και ως Φυσική Υψηλών Ενεργειών (High Energy Physics - HEP) διότι πολλά από τα στοιχειώδη αυτά σωματίδια δεν συναντώνται υπό κανονικές συνθήκες στη φύση αλλά μπορούν να δημιουργηθούν και να παρατηρηθούν μόνο κατά τη διάρκεια ενεργειακών συγκρούσεων με άλλα σωματίδια στο περιβάλλον ενός σωματιδιακού επιταχυντή.

Η ιδέα ότι η ύλη αποτελείται από στοιχειώδη σωματίδια πρωτοεμφανίστηκε τουλάχιστον τον 6ο αιώνα π.Χ. Απασχόλησε αρχαίους Έλληνες φιλοσόφους, Ινδούς φιλοσόφους, επιστήμονες κατά τη διάρκεια του μεσσαίωνα και νεότερους ευρωπαίους επιστήμονες όπως τους Robert Boyle και Isaac Newton. Τον 20ο αιώνα οι πρόοδος της πυρηνικής καθώς και κβαντικής φυσικής άνοιξαν νέους ορίζοντες στη σωματιδιακή φυσική και οδήγησαν στην εδραίωση του Καθιερωμένου Μοντέλου (βλ. σχήμα 2.1, σελίδα 18) τη δεκαετία του 1970.

Η έρευνα πάνω στη σωματιδιακή φυσική σήμερα εστιάζεται στα υποατομικά σωματίδια, όπως ηλεκτρόνια, πρωτόνια, νετρόνια, και, πέραν αυτών, σε πολλά άλλα σωματίδια, παράγωγα των υποατομικών σωματιδίων, σχηματιζόμενα μέσω συγκρούσεων μεγάλων ταχυτήτων (υψηλών ενεργειών) μεταξύ τους.

Επεξηγηματικά, στις αρχές της δεκαετίας του '30 οι θεωρητικοί φυσικοί νόμιζαν ότι είχαν ανακαλύψει όλο τον ατομικό κόσμο, τα πρωτόνια, νετρόνια, ηλεκτρόνια καθώς και μερικά ακόμη σωματίδια που είχαν βρεί. Αλλά 30 χρόνια αργότερα, στις



Σχήμα 2.1: Το Καθιερωμένο Μοντέλο της Σωματιδιακής Φυσικής

αρχές της δεκαετίας του ΄60, ανακάλυψαν ένα τεράστιο πλήθος σωματιδίων, περίπου διακόσια σωματίδια, είτε στην κοσμική ακτινοβολία είτε στα πειράματα που έκαναν με τους επιταχυντές της εποχής εκείνης. Προσπάθησαν να τα κατατάξουν σε ομάδες για την καλύτερη εξήγησή τους, συγχρόνως όμως προσπαθούσαν να βρουν μοντέλα που θα εξηγούσαν πως αλληλεπιδρούν μεταξύ τους τα θεμελιώδη δομικά υλικά του σύμπαντος.

Έτσι, τα είχαν χωρίσει: Πρώτον στα αδρόνια, που είναι όλα τα σωμάτια που αναπτύσσουν τις ισχυρές αλληλεπιδράσεις. Τα αδρόνια χωρίζονται στα μεσόνια (που όλα είναι μποζόνια) και στα βαρυόνια (που όλα είναι φερμιόνια). Δεύτερον στα λεπτόνια που είναι όλα φερμιόνια και που συμμετέχουν σε ασθενείς και ηλεκτρομαγνητικές αλληλεπιδράσεις κι όχι σε ισχυρές. Τρίτον στα μποζόνια βαθμίδας, που είναι οι φορείς της ηλεκτρασθενούς δύναμης (+W, -W, Z) και τέταρτον στα γκλουόνια (από την ελληνική λέξη γλοιόνια-κολλώδη), που είναι οι φορείς της ισχυρής πυρηνικής δύναμης.

Σήμερα, ένας ικανός αριθμός εργαστηρίων ανά τον κόσμο ασχολείται με την εν λόγω έρευνα παράγοντας καθημερινά νέες εργασίες και άρθρα. Μερικά από αυτά είναι το CERN (European Organization for Nuclear Research) κοντά στη Γενεύη της Ελβετίας, το DESY (Deutsches Elektronen Synchrotron) κοντά στο Αμβούργο της Γερμανίας, το Fermilab (Fermi National Accelerator Laboratory) κοντά στο Σικάγο των Η.Π.Α. και το SLAC (Stanford Linear Accelerator Center) κοντά στο Στάνφορντ των Η.Π.Α.

Οι ψηφιακές βιβλιοθήκες, που αποτελούν αντικείμενο διερεύνησης στην παρούσα εργασία, δημιουργήθηκαν ακριβώς για να εξυπηρετήσουν την καλύτερη επικοινωνία, την ανταλλαγή εμπειριών και τη συνεργασία των διαφόρων επιστημόνων που ασχολούνται με τα θέματα αυτά σε παγκόσμιο επίπεδο. Την αναγκαιότητα αυτή των ψηφιακών βιβλιοθηκών κατανόησε και το CERN, το οποίο άρχισε να ασχολείται συστηματικά με το θέμα αυτό ήδη από τις αρχές της δεκαετίας του 1990, όπως θα δούμε παρακάτω.

2.2 Ευρωπαϊκός Οργανισμός Πυρηνικών Ερευνών

Ο Ευρωπαϊκός Οργανισμός Πυρηνικών Ερευνών (European Organization for Nuclear Research - CERN) είναι το μεγαλύτερο σε έκταση (πειραματικό) κέντρο πυρηνικών ερευνών και σωματιδιακής φυσικής στον κόσμο. Βρίσκεται δυτικά της Γενεύης, στα σύνορα Ελβετίας και Γαλλίας. Ιδρύθηκε το 1954 από δώδεκα ευρωπαϊκές χώρες, μεταξύ των οποίων και η Ελλάδα, και σήμερα αριθμεί 20 κράτη-μέλη. Η κύρια λειτουργία του αφορά στην παροχή επιταχυντών σωματιδίων και άλλων υλικοτεχνικών υποδομών που χρειάζονται για την πειραματική έρευνα στο πεδίο της φυσικής υψηλών ενεργειών. Στο CERN βρίσκεται ο Μεγάλος Επιταχυντής Αδρονίων (Large Hadron Collider - LHC), ένας επιταχυντής στοιχειωδών σωματιδίων, ο οποίος φιλοξενεί διάφορα από τα τρέχοντα πειράματα του οργανισμού μεταξύ των οποίων τα ALICE, ATLAS, CMS, LHCb.

Σκοπός των πειραμάτων αυτών είναι, μεταξύ άλλων, ο σχηματισμός του – προς το παρόν μη παρατηρήσιμου πλην όμως αναζητούμενου – σωματιδίου Χιγκς (Higgs Boson). Ελπίζεται ότι από τις αναμενόμενες συγκρούσεις των πρωτονίων, στη δεδομένη ενέργεια, θα παρατηρηθεί το εν λόγω σωματίδιο, το οποίο αποτελεί και έναν από τους χαμένους κρίκους του Καθιερωμένου Μοντέλου της σωματιδιακής φυσικής. Η πειραματική παρατήρηση του μποζονίου Χιγκς θα ερμηνεύσει πώς τα υπόλοιπα στοιχειώδη σωματίδια αποκτούν μάζα κατά το μοντέλο Χιγκς.

Τα πειράματα που λαμβάνουν χώρα στο CERN, καθώς και τους υπόλοιπους παρόμοιους οργανισμούς ανά τον κόσμο, παράγουν καθημερινά νέα συμπεράσματα καθώς και ενδιάμεσα συμπεράσματα τα οποία τεκμηριώνονται εγγράφως από τους σχετικούς επιστήμονες. Τα έγγραφα αυτά, όπως και όλα τα υπόλοιπα έγγραφα που παράγονται, άμεσα και έμμεσα συσχετιζόμενα με τις ερευνητικές διαδικασίες και τη λειτουργία του εκάστοτε οργανισμού, πρέπει να αποθηκεύονται ψηφιακά και να είναι διαθέσιμα για αναζήτηση ανά πάσα στιγμή από τους ενδιαφερόμους χρήστες.

2.3 Ψηφιακές βιβλιοθήκες

Μια ψηφιακή βιλβιοθήκη είναι μια βιβλιοθήκη η οποία παρέχει το σύνολο του υλικού της σε ψηφιακή μορφή, το οποίο είναι προσβάσιμο μέσω ηλεκτρονικών υπολογιστών. Οι σύγχρονες ψηφιακές βιβλιοθήκες μπορούν να υφίστανται μόνο ηλεκτρονικά, χωρίς δηλαδή κάποια φυσική υποδομή, και η ιστορία τους ξεκινά την τελευταία δεκαετία του εικοστού αιώνα. Τα περιεχομένα μιας ψηφιακής βιβλιοθήκης μπορεί να είναι φυσικά έγγραφα τα οποία έχουν ψηφιοποιηθεί ή και έγγραφα που είναι εκ της δημιουργίας τους ψηφιακά. Μια ψηφιακή βιβλιοθήκη τυπικά επιτρέπει την αναζήτηση στα περιεχόμενά της, επιτρέποντας έτσι την ευκολότερη εύρεση συγκεκριμένων πόρων της. Συχνές είναι και οι συνεργασίες μεταξύ ψηφιακών βιβλιοθηκών για συγκεντρωτικές αναζητήσεις στα περιεχόμενά τους όπως θα δούμε στην ενότητα 2.4, σελίδα 20.

Σήμερα οι περισσότεροι ακαδημαϊκοί και επιστημονικοί οργανισμοί φιλοξενούν μια ψηφιακή βιλβιοθήκη του έργου τους, όπως βιβλία, άρθρα, δημοσιεύσεις, περιοδικά και άλλα. Πολλές από αυτές τις βιβλιοθήκες είναι ελεύθερα προσβάσιμες, προσφέροντας έτσι άμεσα ελεύθερη πρόσβαση στην επιστήμη.

Τα πλεονεκτήματα των ψηφιακών βιβλιοθηκών είναι πολλά. Η φυσική υπόσταση της βιβλιοθήκης δεν είναι πλέον απαραίτητη και καθένας μπορεί να την επισκεφτεί ψηφιακά, ανεξαρτήτως ώρας και μέρους. Οι πόροι της βιβλιοθήκης δεν είναι περιορισμένοι, είναι δηλαδή διαθέσιμοι για όλους ανά πάσα στιγμή. Πολύπλοκες αναζητήσεις είναι δυνατές στα περιεχόμενα της βιβλιοθήκης για την εύρεση συγκεκριμένων πόρων. Δεν υπάρχει το πρόβλημα της φθοράς από τη χρήση, και η διατήρηση είναι ευκολότερη. Οι περιορισμοί στο χώρο είναι μηδαμινοί. Τέλος, η ποιότητα αποθήκευσης πληροφοριών είναι πολύ καλύτερη.

Δε λείπουν όμως και τα μειονεκτήματα από τις ψηφιακές βιβλιοθήκες. Οι διαφορετικές τεχνολογίες που χρησιμοποιούνται από διάφορες ψηφιακές βιβλιοθήκες, καθώς και η συνεχής ανάπτυξη και βελτίωσή τους δημιουργούν το πρόβλημα της συνεπούς διατήρησης των πόρων μιας ψηφιακής βιβλιοθήκης στην πάροδο του χρόνου. Προβλήματα επίσης μπορεί να δημιουργήσει η κυριότητα και τα δικαιώματα κάποιου έργου που φιλοξενείται από μια ψηφιακή βιβλιοθήκη λόγω της φύσης της ψηφιακής πληροφορίας και της διάδοσής της. Τέλος, η πολυπλοκότητα της καταλογράφησης και της αποθήκευσης των μεταδεδομένων για κάποια έργα μπορεί να αποτελέσει πρόβλημα στην καταγραφή και αναζήτησή τους.

2.4 Συνεργαζόμενες Ψηφιακές Βιβλιοθήκες

Η συνεργασία μεταξύ ψηφιακών βιβλιοθηκών βασίζεται κυρίως στη δυνατότητα πρόσβασης στα περιεχόμενά τους, που παρέχουν σε άλλες μηχανές αναζήτησης. Συχνά γίνεται από ψηφιακές βιβλιοθήκες η χρήση του καθιερωμένου πρωτοκόλλου συγκομιδής μεταδεδομένων του Open Archives Initiative (OAI-PMH), για την παροχή και εξαγωγή των μεταδεδομένων τους.

Υπάρχουν δύο κυρίως τεχνικές αναζήτησης σε μία συνεργασία ψηφιακών βιβλιοθηκών: η κατανεμημένη αναζήτηση και η αναζήτηση σε μεταδεδομένα που έχουν ήδη συλλεχθεί. Στην περίπτωση της κατανεμημένης αναζήτησης ένας εξυπηρετητής εκτελεί παράλληλες αναζητήσεις σε διάφορους εξυπηρετητές-μέλη της συνεργασίας. Τα αποτελέσματα, αφού περάσουν μια επεξεργασία, επιστρέφονται στον τελικό χρήστη. Το πλεονέκτημα αυτής της προσέγγισης είναι ότι αποφεύγεται η τοπική ευρετηρίαση και αποθήκευση του περιεχομένου των διάφορων ψηφιακών βιβλιοθηκών, και συνεπώς και το κόστος σε υπολογιστικούς πόρους που επιφέρουν.

Συγχρόνως όμως αυτό αποτελεί και μειονέκτημα εφόσον δεν υπάρχει τοπικός έλεγχος για την ευρετηρίαση και κατάταξη του περιεχομένου των διάφορων ψηφιακών βιβλιοθηκών, και κατά συνέπεια και της συλλογής των απολύτως επιθυμητών αποτελεσμάτων. Στην περίπτωση της αναζήτησης σε μεταδεδομένα τα οποία έχουν ήδη συλλεχθεί, ο εξυπηρετητής φροντίζει να επικοινωνεί περιοδικά με τις διάφορες ψηφιακές βιβλιοθήκες της συνεργασίας, συλλέγοντας μεταδεδομένα και δημιουργώντας ένα τοπικό ευρετήριο πληροφοριών. Το πρωτόκολλο (OAI-PMH), για το οποίο έγινε λόγος παραπάνω, χρησιμοποιείται συχνά σε αυτές τις περιπτώσεις. Το πλεονέκτημα αυτής της προσέγγισης είναι ότι ο τελικός μηχανισμός αναζήτησης έχει πλήρη έλεγχο στην ευρετηρίαση και κατάταξη των αποτελεσμάτων, παράγοντας πιο συνεπή αποτελέσματα. Στον αντίποδα, οι διαδικασίες συγκομιδής και ευρετηρίασης είναι ιδιαίτερα απαιτητικές σε υπολογιστικούς πόρους.

2.5 Παραδείγματα Ψηφιακών βιβλιοθηκών

Παρακάτω παρουσιάζονται μερικές από τις πιο διαδεδομένες πλατφόρμες ψηφιακών βιβλιοθηκών. Αναλυτικά το λογισμικό που αναπτύσσεται και χρησιμοποιείται στο CERN περιγράφεται στο κεφάλαιο 3, σελίδα 25.

2.5.1 EPrints

Το EPrints είναι ένα πακέτο ελεύθερου λογισμικού που χρησιμοποιείται για την κατασκευή αποθετηρίων εγγράφων ελεύθερης πρόσβασης. Είναι συμβατό με το πρωτόκολλο συγκομιδής μεταδεδομένων του Open Archives Initiative, και χρησιμεύει κυρίως ως ένα σύστημα διαχείρισης εγγράφων για ιδρύματα όπως ακαδημαϊκοί και επιστημονικοί οργανισμοί. Η ανάπτυξη του ξεκίνησε το 2000 από το School of Electronics and Computer Science του πανεπιστημίου του Southampton, και διατίθεται βάσει της άδειας GPL.

Το EPrints χρησιμποιείται ως μια διαδικτυακή εφαρμογή βασισμένη στην αρχιτεκτονική LAMP (Linux - Apache - MySQL - PHP/Python/Perl), όπου στην προκειμένη περίπτωση το λογισμικό είναι γραμμένο σε Perl. Το λογισμικού χρησιμοποιεί plugins για την εισαγωγή και εξαγωγή δεδομένων και την μετατροπή αντικειμένων ενώ widgets χρησιμοποιούνται για τη διεπαφή του χρήστη. Η ρύθμιση και προσαρμογή του λογισμικού μπορεί να γίνει μέσω μιας σειράς από XML αρχεία, ενώ η εμφάνιση του ελέγχεται από προσαρμόσιμα πρότυπα HTML.

Περισσότερες πληροφορίες μπορούν να βρεθούν στην ιστοσελίδα του λογισμικού [16].

2.5.2 DSpace

Το DSpace είναι ένα πακέτο ελεύθερου λογισμικού που παρέχει εργαλεία για τη διαχείριση ψηφιακών πόρων. Προορίζεται για χρήση από ιδρύματα όπως ακαδημαϊκοί και επιστημονικοί οργανισμοί ως εργαλείο διαχείρισης ψηφιακών πόρων, αλλά και ως πλατφόρμα ψηφιακής διατήρησης. Υποστηρίζει ένα πλήθος τύπων δεδομένων, όπως βιβλία, εργασίες, διατριβές, ψηφιακές σαρώσεις αντικειμένων τριών διαστάσεων, αρχεία πολυμέσων και άλλα, τα οποία είναι οργανωμένα σε συλλογές αντικειμένων. Το λογισμικό αναπτύσσεται από το 2002 ως συνεργασία δύο φορέων, του πανεπιστημίου MIT της Βοστώνης και των εργαστηρίων έρευνας της εταιρίας Hewlett-Packard, και διατίθεται βάσει της άδειας BSD.

Η κοινότητα του DSpace χρησιμοποιεί ένα μοντέλο ανάπτυξης σύμφωνα με το οποίο μέσα από ένα σύνολο χρηστών, διαφορετικές ομάδες φροντίζουν να αναπτύσουν το λογισμικό και να ελέγχουν την ορθότητα και συμφωνία του με τους καθορισμένους κανόνες ανάπτυξης. Πρόκειται κυρίως για μια διαδικτυακή εφαρμογή. Το λογισμικό είναι γραμμένο στη γλώσσα προγραμματισμού Java, με χρήση της διεπαφής προγραμματισμού εφαρμογών Java Servlet, ενώ χρησιμοποιείται και μια τυπική σχεσιακή βάση δεδομένων, όπως η PostgreSQL ή η Oracle. Υποστηρίζεται και η συγκομιδή μεταδεδομένων μέσω του αντίστοιχου πρωτοκόλλου OAI-PMH του Open Archives Initiative.

Περισσότερες πληροφορίες μπορούν να βρεθούν στην ιστοσελίδα του λογισμικού [17].

2.5.3 Fedora

Το Fedora είναι μια πλατφόρμα αρχιτεκτονικής διαχείρισης ψηφιακών πόρων, βασισμένη στη λογική των αυτόνομων λογισμικών μονάδων, και μπορεί να αποτελέσει τη βάση για την κατασκευή ψηφιακών βιβλιοθηκών, αποθετηρίων ακαδημαϊακών και επιστημονικών οργανισμών καθώς και συστημάτων ψηφιακών αρχείων και διατήρησης. Το λογισμικό προορίζεται ως η αρχιτεκτονική βάση, αλλά δεν αποτελεί ολοκληρωμένη λύση διαχείρισης, ευρετηρίασης, εύρεσης και διάθεσης, για τη δημιουργία ενός ψηφιακού αποθετηρίου. Η ανάπτυξή του ξεκίνησε το 1997 από το πανεπιστήμιο Cornell, ενώ σήμερα αναπτύσσεται από κοινού από το ίδιο πανεπιστήμιο καθώς και από τη βιβλιοθήκη του πανεπιστημίου της Βιρτζίνια, και διατίθεται βάσει της άδειας Apache.

Το βασικό χαρακτηριστικό του λογισμικού είναι ένα στρώμα διαχείρισης γενικής χρήσης για ψηφιακά αντικείμενα. Η διαχείριση των αντικειμένων αυτών γίνεται μέσω μοντέλων περιεχομένου τα οποία αναπαριστούν αντικείμενα δεδομένων ή και συλλογές τους. Τα αντικείμενα αυτά μπορούν να περιέχουν συνδέσμους με εγγραφές, μεταδεδομένα, δεδομένα διαχείρισης ή ακόμη και εντολές/πράξεις. Η πρόσβαση στο στρώμα διαχείρισης μπορεί να γίνει είτε μέσω μιας εφαρμογής πελάτη, είτε μέσω μιας διεπαφής προγραμματισμού εφαρμογών για διαδικτυακή πρόσβαση. Η εισαγωγή και εξαγωγή ψηφιακών αντικειμένων μπορεί να πραγματοποιηθεί με αρχεία XML διαφόρων τύπων.

Περισσότερες πληροφορίες μπορούν να βρεθούν στην ιστοσελίδα του λογισμικού [18].

2.5.4 Greenstone

Το Greenstone αποτελεί μια σειρά εργαλείων λογισμικού για την κατασκευή και διάθεση ψηφιακών βιβλιοθηκών. Χρησιμοποιείται για τη δημιουργία εκτενών συλλογών ψηφιακών εγγράφων με δυνατότητα αναζήτησης και προσφέρει μια διεπαφή διαχείρισης τους για βιβλιοθηκονόμους. Χειρίζεται ένα πλήθος μορφών αρχείων, όπως κείμενο και πολυμέσα, ενώ ειδικά για τα αρχεία κειμένου χρησιμοποιεί μια εσωτερική δομή βασισμένη σε XML. Είναι ελεύθερο λογισμικό, αναπτύσσεται από το πρόγραμμα ψηφιακών βιβλιοθηκών της Νέας Ζηλανδίας στο πανεπιστήμιο του Waikato, και διατίθεται βάσεις της άδειας GPL.

Περισσότερες πληροφορίες μπορούν να βρεθούν στην ιστοσελίδα του λογισμικού [19].

Κεφάλαιο 3

Ο Εξυπηρετητής Αρχείων του CERN: CDS Invenio

3.1 Ιστορικό

Το 1993 πρωτοεμφανίστηκε στο διαδίκτυο με τη μορφή του CERN Preprint Server το CDS (CERN Document Server), με σκοπό τη συλλογή για διάδοση όλων των ερευνητικών εγγράφων σχετικών με φυσική υψηλών ενεργειών. Χρησιμοποιήθηκε κυρίως ως θεσμικό αποθετήριο/πηγή πληροφοριών (institutional repository) με αρχικά δύο ξεχωριστές συλλογές εγγράφων: άρθρα από το ίδιο το CERN και επιστημονικά έγγραφα και άρθρα από όλο τον κόσμο τα οποία αναλάμβανε να σαρώνει (σκαννάρει / scan) η βιβλιοθήκη του CERN.

Το 1996 μετονομάστηκε σε CERN Library server (weblib) χρησιμοποιώντας το ίδιο λογισμικό για να παρέχει πρόσβαση σε επιπλέον υλικό όπως περιοδικά βιβλία και το περισσότερο υλικό που ήταν διαθέσιμο στη βιβλιοθήκη.

Το 2000 η νέα έκδοση του λογισμικού ήταν διαθέσιμη, με όνομα αυτή τη φορά: CERN Document Server Software: CDSware. Η νέα αυτή έκδοση επέτρεψε την εισαγωγή αρχείων πολυμέσων στην ψηφιακή βιβλιοθήκη, όπως φωτογραφίες, βίντεο, φυλλάδια, αφίσες και άλλα. Συγχρόνως το λογισμικό έγινε σύμμορφο προς τους κανόνες του OAI (Open Archives Initiative) και άρχισε να διανέμεται και να χρησιμιποιείται σε περισσότερα μέρη όπως το San Diego Supercomputer Center στο Σαν Ντιέγκο των Η.Π.Α., το HBZ NRW στην Κολωνία της Γερμανίας και το Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης.

Υπό αυτή τη μορφή και από το 2004 και μετά το CDSware επιλέχτηκε και ως το επίσημο σύστημα διαχείρισης αρχείων της διεύθυνσης του CERN για όλα τα εισερχόμενα και εξερχόμενα έγγραφα. Την ίδια χρονιά ξεκίνησε η συνεργασία μεταξύ του CERN και του EPFL (École Polytechnique Fédérale de Lausanne) για την συμμετοχική ανάπτυξη του λογισμικού.

Το 2006 το CDSware μετονομάστηκε σε CDS Invenio, το οποίο είναι και το τρέχον όνομα του λογισμικού, και έγινε πιο ξεκάθαρος ο ρόλος του CDS Software Consortium ως ομάδας που αναπτύσσει και συντηρεί λογισμικό διαχείρισης εγγράφων και συνεδρίων με κυρίως προϊόντα το CDS Invenio και το CDS Indico.

Το 2007 ξεκίνησε η συνεργασία μεταξύ του CDS Invenio και του SPIRES[1] με σκοπό τη δημιουργία του INSPIRE, μιας συγκεντρωτικής ψηφιακής βιβλθιοθήκης για όλα τα έγγραφα σχετικά με φυσική υψηλών ενεργειών. Το σχέδιο, που είναι μια συνεργασία μεταξύ τεσσάρων οργανισμών ανά τον κόσμο (CERN, DESY, Fermilab, SLAC) προβλέπεται να είναι σε χρήση μέσα στο 2010. Η εκκίνηση αυτής της συνεργασίας έχει οδηγήση στην ανάπτυξη νέων εργαλίων για το CDS Invenio όπως εξελιγμένα εργαλεία συντήρησης και εμπλουτισμού μεταδεδομένων (metadata), οντολογικά εργαλεία ανάλυσης παραθέσεων και λέξεων-κλειδιών καθώς και εργαλείων αποσαφήνισης συγγραφέων.

3.2 Παρούσα Κατάσταση

Επί του παρόντος το CDS Invenio βρίσκεται προ των πυλών της έκδοσης 1.0 του λογισμικού του, η οποία αναμένεται εντός των πρώτων μηνών του 2010. Η έκδοση 1.0 του CDS Invenio θα περιέχει πολλά από τα εργαλεία και χαρακτηριστικά της επερχόμενης διάθεσης του προαναφερθέντος INSPIRE. Η κυρίως ανάπτυξη του CDS Invenio γίνεται από το CDS Software Consortium, μια ομάδα μηχανικών λογισμικού βασισμένη στο CERN αλλά συγχρόνως συνεισφορές παρέχονται και από τις διάφορες συνεργασίες με άλλες ομάδες, ακόμη από μεμονωμένους προγραμματιστές και μηχανικούς λογισμικού. Το ίδιο το CDS Software Consortium συγκροτείται από μερικά μόνιμα μέλη-εργαζομένους στο CERN και από διάφορους σπουδαστές, τελειόφοιτους και μεταπτυχιακούς — συμπεραλαμβονεμένου και του γράφοντα, που κατά καιρούς αποτελούν μέρος της ομάδας. Η σύσταση αυτή της ομάδας αλλά και οι συνεισφορές από τους διάφορους συνεργάτες επιτρέπουν στο CDS Invenio να ανανεώνεται συνεχώς με νέες ιδέες, λύσεις και τεχνικές ανάπτυξης καθιστώντας το ιδιαίτερα δυναμικό.

3.3 Γενική Επισκόπηση

Το CDS Invenio είναι ένα σύνολο από εφαρμογές που αποτελούν το πλαίσιο και τα εργαλεία για τη δημιουργία και τη διαχείριση ενός αυτόνομου εξυπηρετητή ψηφιακής βιλβιοθήκης. Το λογισμικό του αποτελεί ελεύθερο λογισμικό και λογισμικό ανοιχτού κώδικα και διατίθεται υπό την άδεια GNU General Public Licence. Η τεχνολογία που προσφέρεται από το λογισμικό αυτό καλύπτει όλες τις πλευρές της διαχείρισης μια ψηφιακής βιβλιοθήκης. Η υψηλή απόδοση και η προσαρμόστικοτητά του το καθιστούν μια ολοκληρωμένη λύση για την πλήρη διαχείριση ενός

[1] Η ψηφιακή βιβλιοθήκη SPIRES ξεκίνηση από το Stanford Linear Accelerator Center (SLAC) τη δεκατία του 1960, ως μια βάση δεδομένων βιβλιογραφίας σωματιδιακής φυσικής. Σήμερα, η βιβλιοθήκη διαχειρίζεται και συντηρείται από το SLAC σε συνεργασία με άλλα ινστιτούτα παγκοσμίως, όπως τα DESY, FNAL, Kyoto, Durham U, IHEP, KEK, LIPI.

αποθετηρίου εγγράφων μεσαίου και μεγάλου μεγέθους. Στο CERN η υπάρχουσα υλοποίηση του CDS Invenio φιλοξενεί πάνω από 1.000.000 βιβλιογραφικά αρχεία διάφορων κατηγοριών, όπως άρθρα, βιβλία, περιοδικά, φωτογραφίες, βίντεο και άλλα. Εκτός από την τρέχουσα τοπική υλοποίηση, το CDS Invenio χρησιμοποιείται σήμερα από τουλάχιστον ακόμη 15 επιστημονικούς οργανισμούς παγκοσμίως.

Το λογισμικό έχει υποστεί μια συνεχή σταδιακή ανάπτυξη ώστε να καταλήξει από την αρχική βασική του έκδοση ως ψηφιακού εξυπηρετητή στο τρέχον σύστημα αποθήκευσης αρχείων υψηλής απόδοσης. Παρά την πολύπλοκότητα που έχει επιφέρει αυτή η πολυετής ανάπτυξη, το CDS Invenio έχει καταφέρει να διατηρήσει την υψηλή του απόδοση, τη φιλικότητα προς το χρήστη και τη δυνατότητα παραμετροποίησής του σε μεγάλο βαθμό με το να παραμένει σύμμορφο σε μια καθιερωμένη αρχιτεκτονική αυτόνομων λογισμικών μονάδων (modular architecture). Από καθαρά τεχνικής άποψης, το CDS Invenio τρέχει πάνω σε συστήματα GNU/Unix , χρησιμοποιώντας τον εξυπηρετητή βάσεων δεδομένων MySQL και τον εξυπηρετητή εφαρμογών διαδικτύου Apache/Python. Η διαμόρφωση κατά τη διάρκεια της μετάφρασης (compile-time configuration) επιτυγχάνεται μέσω του εργαλείου GNU Autoconf ενώ η διαμόρφωση κατά τη διάρκεια της εκτέλεσης (run-time configuration) μέσω διάφορων πινάκων δεδομένων διαμόρφωσης της βάσης δεδομένων MySQL (MySQL configuration tables). Ο κώδικας που αποτελεί το λογισμικό είναι σχεδόν εξ' ολοκλήρου γραμμένος στη γλώσσα προγραμματισμού Python ενώ κάποιες ειδικές λογισμικές μονάδες και λειτουργίες είναι γραμμένες στις γλώσσες προγραμματισμού PHP και Common Lisp.

Το χαρακτηριστικό-κλειδί της αρχιτεκτονικής του CDS Invenio βρίσκεται στη λογική των αυτόνομων λογισμικών μονάδων. Κάθε μία από αυτές τις μονάδες περιλαμβάνει μια συγκεκριμένη και ορισμένη λειτουργία του συνολικού συστήματος της ψηφιακής βιβλιοθήκης. Οι μονάδες αλληλεπιδρούν μεταξύ τους καθώς και με τη βάση δεδομένων και το στρώμα διεπαφής. Η λογική κάθε μονάδας, η λειτουργία και διαλειτουργικότητά της είναι επεκτάσιμες και προσαρμόσιμες / επιδεκτικές προσαρμογής. Μια σχηματική επισκόπηση της αρχιτεκτονικής του λογισμικού φαίνεται στο διάγραμμα του σχήματος 3.1, σελίδα 29.

Το διάγραμμα αναπαριστά σχηματικά τη ροή εργασίας από πάνω προς τα κάτω ενός εγγράφου στο CDS Invenio. Στην κορυφή του διαγράμματος, η απόκτηση των δεδομένων πραγματοποιείται από 3 διαφορετικές πηγές: άμεση υποβολή από τον ίδιο το συγγραφέα (χρησιμοποιώντας ηλεκτρονικό ταχυδρομείο ή τη διαδικτυακή διεπαφή) και συγκομιδή από πηγές δεδομένων σύμμορφες με το OAI (Open Archives Initiative) ή και όχι. Τα μεταδεδομένα (metadata) που συγκεντρώνονται μετατρέπονται κατευθείαν σε μια πρότυπη εσωτερική δομή απεικόνισης μεταδεδομένων (MARCXML) ενώ τα πλήρη κείμενα υποβάλλονται και αποθηκεύονται κατευθείαν στον εξυπηρετητή αρχείων αφού μετατραπούν πρώτα σε PDF (Portable Document Format). Μόλις τα μεταδεδομένα φορτωθούν στον εξυπηρετητή μπορεί να υποβληθούν σε διαδικασίες αξιολόγησης της ποιότητάς τους από τους καταλογογράφους της βιβλιοθήκης. Επίσης, τα μεταδεδομένα μπορούν να εμπλουτιστούν με παραπομπές που εξάγονται από αντίστοιχα πλήρη κείμενα. Στη συνέχεια ο εξυπηρετητής μπορεί, εάν του ζητηθεί, να υπολογίσει και να παράγει περιεχόμενα, τάξεις, ομάδες και βιβλιογραφικές διατάξεις, κατάλληλα για ταχεία ανάκτηση. Η πληροφορία εν τέλει διανέμεται, όπως φαίνεται στο κάτω μέρος του διαγράμματος, στον τελικό χρήστη και σε παρόχους υπηρεσιών OAI μέσω αιτημάτων OAI-PMH (The Open Archives Initiative Protocol for Metadata Harvesting), ειδοποιήσεων ηλεκτρονικού ταχυδρομίου και της διαδικτυακής μηχανής αναζήτησης. Επιπλέον, η διαδικτυακή διεπαφή προσφέρει πρόσβαση σε εξατομικευμένες συλλογές εγγράφων (καλάθια - baskets), έγγραφη τεκμηρίωση και στατιστικά στοιχεία. Οι περισσότερες από τις αλληλεπιδράσεις μεταξύ των διαφόρων αυτόνομων λογισμικών μονάδων-κλειδιών και της βάσης δεδομένων συγχρονίζονται μέσω ενός προγραμματιστή εργασιών (task scheduler - BibSched), ο οποίος δύναται να χρησιμοποιηθεί και για την εκτέλεση περιοδικών εργασιών (daemon mode).

3.4 Η τεχνολογία πίσω από το CDS Invenio

Σε αυτήν την ενότητα δίνεται έμφαση σε κάποια από τα εσωτερικά τεχνολογικά χαρακτηριστικά του CDS Invenio , κυρίως αυτά που το διαφοροποιούν από άλλα ελεύθερα λογισμικά και πλατφόρμες ψηφιακών βιβλιοθηκών και αποθετηρίων εγγράφων όπως το DSPace ή το EPrints. Επίσης περιγράφονται συνοπτικά οι διάφορες αυτόνομες λογισμικές μονάδες που συγκροτούν το λογισμικό.

3.4.1 Αναπαράσταση των μεταδεδομένων

Όλα τα βιβλιογραφικά δεδομένα στο CDS Invenio αναπαριστώνται εσωτερικά με τη δομή MARC 21. Αυτή η δομή εσωτερικής αναπαράστασης μεταδεδομένων επιλέχτηκε για διάφορους λόγους:

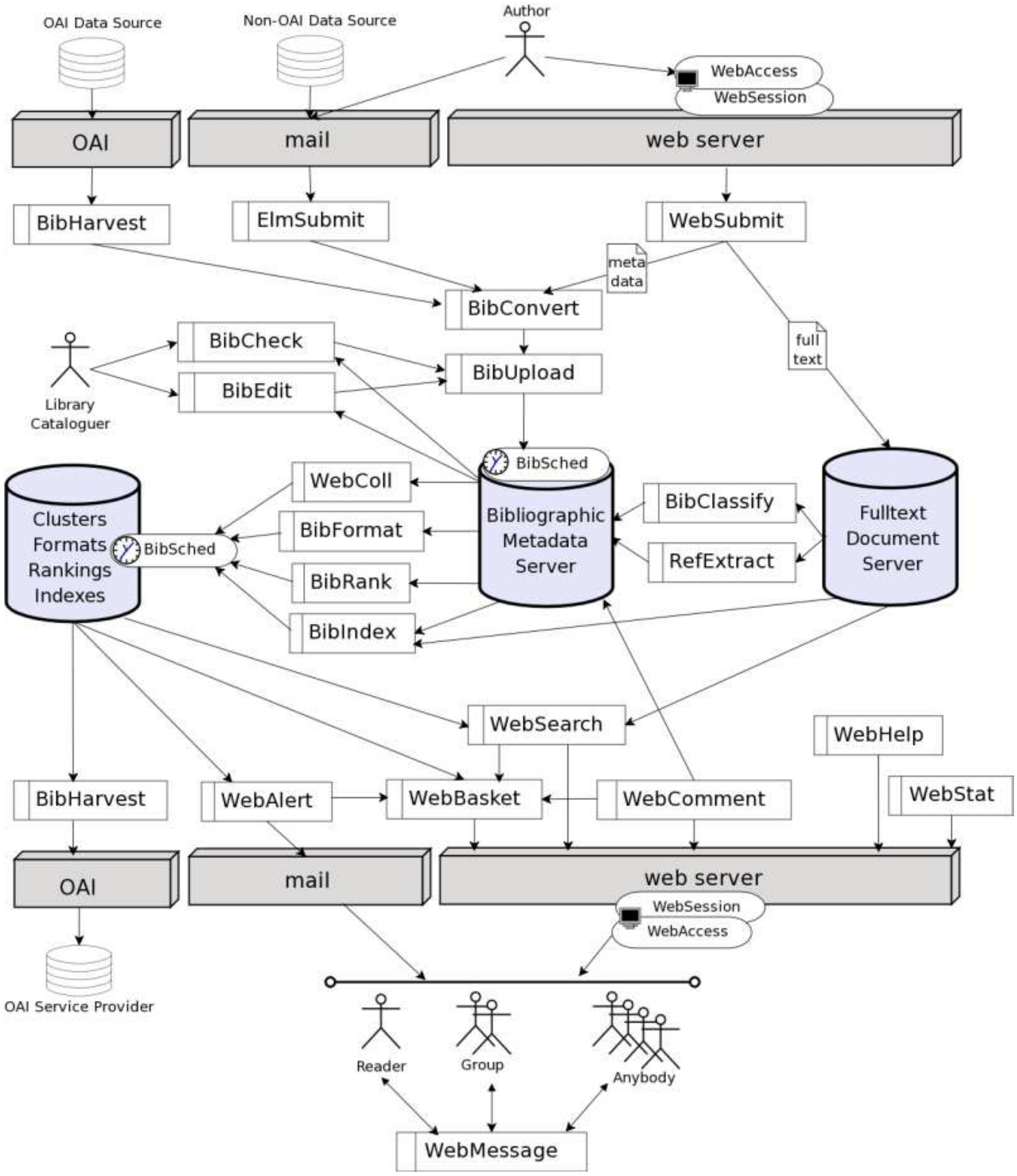
* Είναι ένα εδραιωμένο πρότυπο για τον κόσμο των βιβλιοθηκών, χρησιμοποιούμενο από το 1960.

* Συνδυάζεται καλά με σύγχρονες περιγραφικές τεχνολογίες όπως η γλώσσα XML. (το CDS Invenio χρησιμοποιεί την προσφάτως τυποποιημένη διάταξη MARCXML που παρέχεται από τη Βιβλιοθήκη του Κογκρέσου των Ηνωμένων Πολιτειών)

* Είναι ευπροσάρμοστο και συνεπώς μακροπρόθεσμα αξιόπιστο.

* Είναι απόλυτα επεκτάσιμο ώστε να προσαρμόζεται σε οποιαδήποτε δομή μεταδεδομένων (η τρέχουσα διάταξη MARCXML στο CDS Invenio περιλαμβάνει περισσότερα από 150 πεδία μεταδεδομένων)

Τα αποθετήρια που φιλοξενούν ομοιογενή έγγραφα συνήθως δε χρειάζεται να χρησιμοποιήσουν ένα πλήρες σύστημα καταλογογράφησης MARC. Σε αυτή την περίπτωση μπορούν να υιοθετήσουν την προεπιλεγμένη περιγραφική δομή που προσφέρει το



Σχήμα 3.1: Διάγραμμα των αυτοτελών λογισμικών μονάδων του CDS Invenio

CDS Invenio και η οποία καλύπτει τα πιο κοινά πεδία μεταδεδομένων. Σε διαφορετική περίπτωση, ο χρήστης μπορεί να υλοποιήσει, επιπλέον της προεπιλεγμένης δομής, επιπρόσθετα περιγραφικά πεδία για ειδικούς και τοπικούς τύπους μεταδεδομένων. Σε μια ακόμη πιο προχωρημένη περίπτωση ο χρήστης μπορεί να θέλει να ορίσει ιδιαίτερα συγκεκριμένα και ξεχωριστά αντικείμενα δεδομένων, υλοποιώντας έτσι μια καινούργια περιγραφική δομή εξ αρχής. Αυτή η ακραία περιγραφική επεκτασιμότητα μαζί με την παραμετροποίηση του CDS Invenio επιτρέπουν πρακτικά στο σύστημα να χειρίζεται οποιοδήποτε τύπο μεταδεδομένων (από εκθέματα ενός μουσείου ως παρουσιάσεις πολυμέσων).

3.4.2 Απόκτηση των μεταδεδομένων

Η απόκτηση των μεταδεδομένων πραγματοποιείται από αυτόματες και ημιαυτόματες διαδικασίες συγκομιδής εγγράφων (μονάδα BibHarvest) – εφαρμόζοντας είτε τυποποιημένες προσεγγίσεις όπως η χρήση της συγκομιδής δεδομένων σύμμορφης με το OAI-PMH είτε ειδικές διαδικασίες όπως τη ρηχή διαδικτυακή συγκομιδή. Η υποβολή εγγράφων γίνεται κατευθείαν από τους συγγραφείς τους μέσω του διαδικτύου ή ηλεκτρονικού ταχυδρομείου μέσω των μονάδων WebSubmit και ElmSubmit. Και στις δύο περιπτώσεις τα μεταδεδομένα συγκεντρώνονται πρώτα ακατέργαστα, στη συνέχεια μετατρέπονται στην εσωτερική δομή αναπαράστασης του CDS Invenio και τέλος καταχωρούνται στον εξυπηρετητή βιβλιογραφικών μεταδεδομένων.

Η μετατροπή λαμβάνει χώρα στη μονάδα BibConvert, η οποία επιτρέπει μετατροπές μεταξύ διαφόρων διατάξεων ακολουθίας (π.χ. ISO2709) και ημι-δομημένων διατάξεων (π.χ. XML) αλλά και μεταξύ διαφόρων διατάξεων μεταδεδομένων (π.χ. MARCXML, DublinCore, RFC1807). Η εν λόγω μονάδα παρέχει επίσης τη δυνατότητα λεπτομερούς μορφοποιήσης κειμένου, συμπεριλαμβανομένων και των κανονικών εκφράσεων που προσδιορίζονται στη γλώσσα διαμόρφωσης του BibConvert. Η γλώσσα διαμόρφωσης αυτή παρέχει έναν ορισμό της σύνταξης και σημασιολογίας της περιγραφής της μετατροπής των μεταδεδομένων, ο οποίος κωδικοποιείται σε ένα σύνολο προτύπων μετατροπής. Τα πρότυπα αυτά αναλαμβάνουν την εξόρυξη της περιγραφής του κάθε εγγράφου, παρέχουν την περιγραφή κάθε πεδίου μεταδεδομένων για κάθε έγγραφο καθώς και τη διαρρύθμίση τους.

Επιπλέον, το BibConvert παρέχει ένα μηχανισμό ελέγχου αντιστοιχίας των νέων εγγράφων με τα υπάρχοντα βιβλιογραφικά μεταδεδομένα στον εξυπηρετητή με σκοπό την αποφυγή διπλοεγγραφών (π.χ. σε περίπτωση πολλαπλής καταχώρησης). Τα έγγραφα τα οποία σημειώνονται ως διφορούμενα σε αυτό το στάδιο απορρίπτονται προσωρινά και προωθούνται για έλεγχο από το διαχειριστή απόκτησης των μεταδεδομένων. Εν συνεχεία τα επιβεβαιωμένα μεταδεδομένα καταχωρούνται στον εξυπηρετητή υπό τη δομή MARC 21 όπως περιγράφηκε στην υποενότητα 3.4.1, σελίδα 28.

Η μετατροπή των μεταδεδομέμων με το BibConvert επιτρέπει σε μεγάλο βαθμό την αυτοματοποίηση της όλης διαδικασίας: έγγραφα από διαφορετικές πηγές μπορούν εύκολα να καταχωρηθούν με τη διάταξη MARCXML και άμεσα να εισαχθούν στο σύστημα με χρήση απλά μερικών τυποποιημένων προτύπων διαμόρφωσης. Προκειμένου να διαπιστωθεί η γνησιότητα των μεταδεδομένων που εισάγονται στο σύστημα, οι καταλογογράφοι της βιβλιοθήκης μπορούν να πραγματοποιήσουν ελέγχους αξιολόγησης της ποιότητάς τους μέσω της μονάδας BibCheck.

3.4.3 Ευρετηρίαση και Κατάταξη

Οι μονάδες ευρετηρίασης και κατάταξης αποτελούν τον πυρήνα του CDS Invenio. Καθιερώθηκαν ειδικά σχεδιασμένοι δείκτες ευρετηρίου ώστε να παρέχουν εξαιρετική ταχύτητα ανταπόκρισης σε αποθετήρια μεγέθους ενός εκατομμυρίου εγγράφων. Εκτός από την αναζήτηση στα μεταδεδομένα, το CDS Invenio παρέχει επίσης τη δυνατότητα ευρετηρίου και αναζήτησης σε πλήρη κείμενα και τις αναφορές τους έτσι ώστε η συνδυασμένη αναζήτηση μεταδεδομένων / πλήρους κειμένου / αναφορών να είναι δυνατή. Για παράδειγμα μπορεί κάποιος να κάνει την παρακάτω αναζήτηση: *find all documents written by Ellis in 2002 that mention the term Higgs boson in the fulltext and that refer to Physical Review D 1997 paper.*

Τα αποτελέσματα που ανακτώνται από τη μηχανή αναζήτησης μπορούν να καταταχθούν επιπλέον σύμφωνα με διάφορα κριτήρια. Η προεπιλεγμένη εγκατάσταση του CDS Invenio περιλαμβάνει το κλασικό μοντέλο διαστήματος δεικτών συχνότητας λέξεων που επιτρέπει την ανάκτηση παρόμοιων εγγραφών. Επιπροσθέτως, περιλαμβάνει ένα μηχανισμό μεθόδων κατάταξης που βασίζεται σε συγκεκριμένες τιμές μεταδεδομένων. Για παράδειγμα, η μέθοδος κατάταξης σύμφωνα με τον συντελεστή απήχησης (impact factor) των περιοδικών η οποία κατατάσσει τα έγγραφα χρησιμοποιώντας μια παραμετροποιήσιμη γνωσιακή βάση περιοδικών και των αντίστοιχων συντελεστών απήχησης. Τέλος, πρόσφατες μέθοδοι παρέχουν τη δυνατότητα κατάταξης σύμφωνα με τον αριθμό των αναφορών και τον αριθμό των μεταφορτώσεων.

Τα αποτελέσματα της αναζήτησης ομαδοποιούνται σε συλλογές (collections). Ο διαχειριστής του συστήματος έχει τη δυνατότητα να ορίσει δέντρα κανονικών αλλά και εικονικών συλλογών διευκολύνοντας έτσι την πλοήγηση του χρήστη στο σώμα των εγγράφων που φιλοξενεί η ψηφιακή βιβλιοθήκη. Παράλληλα, μελετώνται διάφορες τεχνικές αυτοματοποιημένης ομαδοποίησης που θα παρέχουν έναν έξυπνο και αυτόματο τρόπο ομαδοποίησης των αποτελεσμάτων και πλοήγησης σε αυτά.

3.4.4 Διεπαφή και εξατομίκευση

Το CDS Invenio μπορεί να χειριστεί σχεδόν οποιοδήποτε είδος ηλεκτρονικού υλικού, χάρη στην ευέλικτη δομή MARC, όπως περιγράφεται ανωτέρω. Για να απεικονίζει σωστά όμως όλες αυτές τις διάφορες μορφές εγγράφων σύμφωνα με τα χαρακτηριστικά που ορίζει η κάθε μια, χρησιμοποιεί έναν ευέλικτο μηχανισμό εξόδου που δίνει τη δυνατότητα της αυτόματης δημιουργίας συνδέσμων προς εξωτερικούς πόρους με βάση το περιεχόμενο του εγγράφου.

Το γραφικό περιβάλλον του λογισμικού παρέχει στο χρήστη μια σειρά από επιλογές εξατομίκευσης και συλλογικότητας. Ο τελικός χρήστης μπορεί να ορίσει προσωπικές συλλογές εγγραφών (καλάθια - baskets) και περιοδικές ειδοποιήσεις (alerts) σχετικές με την προσθήκη νέων εγγράφων στον εξυπηρετητή με βάση τα πεδία του ενδιαφέροντός του. Ως μέρος των συλλογικών χαρακτηριστικών που παρέχει το λογισμικό ο χρήστης μπορεί να ορίσει το περιεχόμενο ενός καλαθιού ως μοιραζόμενο στα μέλη μιας ομάδας, καθώς επίσης και να σχολιάσει ή να αξιολογήσει έγγραφα.

Ένα ακόμη αξιοσημείωτο χαρακτηριστικό του CDS Invenio είναι η διεθνοποίησή του. Η διεπαφή του λογισμικού έχει μεταφραστεί σε 20 γλώσσες (Βουλγάρικα, Καταλανικά, Τσέχικα, Γερμανικά, Ελληνικά, Αγγλικά, Ισπανικά, Γαλλικά, Κροάτικα, Ουγγρικά, Ιταλικά, Ιαπωνικά, Νορβηγικά, Πολωνικά, Πορτογαλλικά, Ρωσικά, Σλοβάκικα, Σουηδικά, Ουκρανικά και Κινέζικα) που επιτρέπουν στον τελικό χρήστη να επιλέξει δυναμικά τη γλώσσα της επιλογής του. Οι περισσότερες από αυτές τις μεταφράσεις συντηρούνται από τα μέλη της ομάδας ανάπτυξης του CDS Invenio και χρήστες του CERN αλλά και οι συνεισφορές από τους διάφορους διαχειριστές εγκαταστάσεων του λογισμικού ανά τον κόσμο είναι άξιες αναφοράς.

3.4.5 Περιγραφή των αυτοτελών λογισμικών μονάδων

Όπως αναφέρθηκε στην ενότητα 3.3, σελίδα 26, το χαρακτηριστικό-κλειδί της αρχιτεκτονικής του CDS Invenio βρίσκεται στη λογική των αυτόνομων λογισμικών μονάδων. Κάθε μία από αυτές τις μονάδες αναλαμβάνει μια συγκεκριμένη λειτουργία του όλου συστήματος. Κατά αυτό τον τρόπο το σύστημα παραμένει ευέλικτο και παραμετροποιήσιμο και η συντήρησή του διευκολύνεται.

Ο σκοπός της κάθε μονάδας είναι ορισμένος και διαφορετικός από τις υπόλοιπες μονάδες, η λειτουργία τους όμως δεν είναι πάντα ανεξάρτητη. Αν χρειάζεται, μια μονάδα μπορεί να εισάγει και να χρησιμοποιεί λειτουργίες κάποιας άλλης.

Αυτοτελείς λογισμικές μονάδες σχετικές με την απόκτηση δεδομένων

Η εισαγωγή δεδομένων στο CDS Invenio γίνεται ή μπορεί να γίνει με δύο τρόπους: είτε με μια αυτοματοποιημένη διαδικασία την οποία ορίζει ο διαχειριστής, είτε με άμεση εισαγωγή από το χρήστη. Στην πρώτη περίπτωση η μονάδα BibHarvest αναλαμβάνει τη συγκομιδή των δεδομένων από αποθετήρια OAI , η μονάδα BibConvert τη μετατροπή των δεδομένων αυτών στην προεπιλεγμένη δομή XML MARC και τέλος η μονάδα BibUpload την καταχώρηση των τελικών δεδομένων στον εξυπηρετητή. Στη δεύτερη περίπτωση ο χρήστης αρχικά καλείται να εισάγει και να υποβάλει έγγραφα χρησιμοποιώντας τη μονάδα WebSubmit και στη συνέχεια τη μονάδα BibEdit για να επεξεργαστεί τα αντίστοιχα μεταδεδομένα αν το επιθυμεί. Οι σχετικές λογισμικές μονάδες περιγράφονται παρακάτω.

BibHarvest Η μονάδα αυτή ουσιαστικά αναλαμβάνει τη συγκομιδή και διάθεση μεταδεδομένων, σύμμορφη με τους κανόνες και προδιαγραφές του Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH). Σύμφωνα με το εν λόγω πρωτόκολλο, το αποθετήριο των μεταδεδομένων αυτών είναι χτισμένο απευθείας πάνω από τη βάση δεδομένων και διαθέτει ένα διαχειριστή αποθετηρίου OAI που επιτρέπει την εκτέλεση των διοικητικών καθηκόντων στον αποθηκευτικό χώρο πέραν των κυρίως διαχειριστικών λειτουργιών που προσφέρονται. Η βάση δεδομένων μπορεί να είναι μερικώς ή εξ ολοκλήρου ανοιχτή για συγκομιδή στο πεδίο εφαρμογής του OAI-PMH πρωτοκόλλου. Στην περίπτωση αυτή τα μεταδεδομένα παρέχονται σε ακατέργαστη μορφή και η σημασιολογία των διαφόρων πεδίων τους ορίζεται στο πλαίσιο της δομής MARC 21.

BibConvert Η μονάδα αυτή επιτρέπει την μετατροπή μεταδεδομένων από οποιαδήποτε δομημένη ή ημιδομημένη μορφή σε οποιαδήποτε άλλη. Στο CDS Invenio η μετατροπή γίνεται κατά βάση στη δομή XML MARC, η οποία είναι και η προεπιλεγμένη. Παρ΄ όλα αυτά, οι μορφές εισόδου και εξόδου για τη μετατροπή είναι απόλυτα παραμετροποιήσιμες. Η δύναμη της εν λόγω μονάδας έγκειται στο γεγονός ότι δεν θεωρεί δεδομένη κάποιου είδους διάρθρωση για τα δεδομένα εισόδου, αλλά δίνει τη δυνατότητα στο διαχειριστή να προσδιορίσει μια γλώσσα διαμόρφωσης ορίζοντας τη σύνταξη και τη σημασιολογία της κάθε μετατροπής. Αναπόφευκτα, η γλώσσα διαμόρφωσης είναι ιδιαίτερα πολύπλοκη. Το CDS Invenio παρέχει τις διαμορφώσεις για τις πιο συχνές μετατροπές μορφών μεταξύ τους.

Γενικά μια διαμόρφωση θα περιέχει την περιγραφή της δομής των δεδομένων εισαγωγής καθώς και αυτών της εξαγωγής. Ο επεξεργαστής της μονάδας αναλαμβάνει να αναλύσει τα δεδομένα εισόδου δημιουργώντας εν τέλει τη δομή εξόδου με τα δεδομένα που προκύπτουν. Στις περισσότερες περιπτώσεις η χρήση του BibConvert στοχεύει σε δεδομένα που στερούνται XML αναπαράστασης και η επεξεργασία τους συνεπάγεται διάφορα βήματα περιλαμβανομένου του διαχωρισμού εγγραφών, της εξόρυξης πεδίων, της μετατροπής των τιμών τους και τη μορφοποίησής τους. Προϋπόθεση είναι τα δεδομένα εισόδου να βρίσκονται σε οποιαδήποτε δομημένη ή ημιδομημένη μορφή (για παράδειγμα να μην είναι εκφρασμένα σε κάποια φυσική γλώσσα[2] που αποτελεί αντικείμενο εξαγωγής δεδομένων).

BibMatch Η μονάδα αυτή ελέγχει τα νεοεισαχθέντα δεδομένα ταιριάζοντας τα με τα ήδη υπάρχοντα στη βάση δεδομένων προς αποφυγή διπλών εγγραφών.

BibUpload Η μονάδα αυτή επιτρέπει την καταχώρηση νέων δεδομένων και εγγράφων στη βάση δεδομένων. Είθισται τα μεταδεδομένα που καταχωρούνται να βρίσκονται σε ένα καλώς δομημένο XML αρχείο, σύμφωνα με την τρέχουσα προεπιλεγμένη δομή, και συνήθως παρέχονται απευθείας από τη μονάδα BibConvert.

WebSubmit Η μονάδα αυτή είναι μια διεπαφή χρήστη και αποτελεί ουσιαστικά ένα ολοκληρωμένο σύστημα υποβολής που επιτρέπει σε εξουσιοδοτημένα άτομα (συντάκτες, γραμματείς, προσωπικό συντήρησης του αποθετηρίου) την υποβολή μεμονωμένων εγγράφων στον εξυπηρετητή. Το σύστημα αυτό διαθέτει ένα μηχανισμό ροής ελέγχου που εξασφαλίζει την έγκριση των δεδομένων από εξουσιοδοτημένες μονάδες. Συνολικά, υπάρχουν διάφορα διαθέσιμα σχήματα αξιοποίησης της υποβολής, περιλαμβανομένου ενός αυτοματοποιημένου μετατροπέα από διάφορες μορφές

[2] Μια μη προμελετημένη γλώσσα αποτέλεσμα της έμφυτης ανάγκης για διευκόλυνση της επικοινωνίας. Διακρίνεται από κατασκευασμένες και τυπικές γλώσσες, όπως γλώσσες προγραμματισμού και γλώσσες που χρησιμοποιούνται στη μελέτη της τυπικής λογικής, ειδικά της μαθηματικής λογικής.

κειμένου και εικόνων σε πλήρη έγγραφα. Η μονάδα παρέχει επίσης τη δυνατότητα εξόρυξης πληροφοριών με έμφαση σε βιβλιογραφικές οντότητες, όπως αναφορές, συγγραφείς, λέξεις-κλειδιά και άλλα.

ElmSubmit Η μονάδα αυτή δίνει τη δυνατότητα αυτοματοποιημένης υποβολής εγγράφων μέσω ηλεκτρονικού ταχυδρομείου από αξιόπιστες πηγές.

BibEdit Η μονάδα αυτή είναι μια διεπαφή χρήστη που αποτελεί ένα πλήρες γραφικό περιβάλλον για την επεξεργασία μεταδεδομένων. Λειτουργίες όπως η προβολή του ιστορικού των αλλαγών, η προσθήκη και αφαίρεση πεδίων, το κλείδωμα της επεξεργασίας του εγγράφου από τρίτους κατά τη διάρκεια της τρέχουσας επεργασίας του, είναι επίσης διαθέσιμες μεταξύ άλλων.

Αυτοτελείς λογισμικές μονάδες σχετικές με την παροχή δεδομένων

Η παρουσίαση και παροχή των εγγράφων και των μετεδεδομένων στο χρήστη πραγματοποιείται από διάφορες μονάδες: Η BibIndex ευρετηριάζει τα δεδομένα, η BibRank τα κατατάσσει, η BibFormat τα μορφοποιεί για την τελική προβολή στο χρήστη ενώ η WebSearch παρέχει τη δυνατότητα της εκτενούς αναζήτησης τους μέσω μιας διεπαφής χρήστη. Στη συνέχεια αναφερόμαστε ξεχωριστά σε κάθε μία από τις μονάδες αυτές.

BibIndex Η μονάδα αυτή αναλαμβάνει την ευρετηρίαση των μεταδεδομένων, των αναφορών και των εγγράφων. Υπάρχουν δύο είδη ευρετηρίασης: λέξεων και φράσεων. Ο χρήστης μπορεί να ορίσει διάφορους λογικούς δείκτες ευρετηρίασης (για παράδειγμα συγγραφέων, τίτλων) και την αντιστοιχία τους στα πεδία μεταδεδομένων του MARC21. Ένας δείκτης ευρετηρίασης αποτελείται από δύο μέρη. Πρώτον, ο δείκτης που δίνει τη λέξη (ή φράση) που βρέθηκε σε ένα συγκεκριένο πεδίο μεταδεδομένων μαζί με μια λίστα αναγνωριστικών των εγγράφων στα οποία βρέθηκε η συγκεκριμένη λέξη (ή φράση). Δεύτερον, ένας αντίστροφος δείκτης αναγνωριστικών εγγράφων με το σύνολο των λέξεων (ή φράσεων) προς τον πρώτο δείκτη. Η εν λόγω τεχνική επιτρέπει την αποδοτική ανανέωση μόνο των λέξεων που έχουν αλλάξει στα μεταδεδομένα εισόδου ενός εγγράφου. Οι δείκτες ευρετηρίασης σχεδιάστηκαν με σκοπό να προσφέρουν ιδιαίτερα γρήγορη απόκριση στις αναζητήσεις των χρηστών, ταχύτερη από αυτή των δεικτών της βάσης δεδομένων MySQL.

BibRank Η μονάδα αυτή επιτρέπει την δημιουργία διαφόρων κριτηρίων κατάταξης που χρησιμοιπούνται στο στάδιο της αναζήτησης από την αντίστοιχη μηχανή. Για παράδειγμα, κατάταξη σύμφωνα με τη συχνότητα εμφάνισης μια λέξης, ή σύμφωνα με την τιμή ενός πεδίου μεταδεδομένων όπως το βαθμό απήχησης ενός περιοδικού, ή ακόμη και σύμφωνα με τον αριθμό μεταφορτώσεων ενός εγγράφου.

BibClassify Η μονάδα αυτή δίνει τη δυνατότητα αυτόματης εξόρυξης λέξεων κλειδιών από έγγραφα, με βάση τη συχνότητα εμφάνισης ειδικών όρων που λαμβάνονται από ένα ελεγχόμενο λεξιλόγιο. Ένα ελεγχόμενο λεξιλόγιο μπορεί να εκφραστεί ως ένα απλό λεξικό αποθησαύρισης ή ως μια δομημένη ταξονομία σύμμορφη με το Resource Description Framework (RDF) ώστε να καταστεί δυνατή η σημασιολογική ταξινόμηση.

BibFormat Η μονάδα αυτή είναι υπεύθυνη για τη μορφοποίηση των βιβλιογραφικών μεταδεδομένων με τους διάφορους διαθέσιμους τρόπους. Με άλλα λόγια επιτρέπει τον πλήρη διαχωρισμό της διαχείρισης των δεδομένων αυτών καθ' αυτών και της διαρρύθμισης της μορφοποίησης τους. Η λειτουργία της BibFormat μπορεί να εκτελεσθεί όταν ζητηθεί για την μορφοποίηση ενός εγγράφου ή ακόμη και να προεκτελεστεί και να αποθηκεύσει ένα συχνά ζητούμενο είδος μορφοποίησης για ένα ή και περισσότερα έγγραφα (όπως για παράδειγμα την προεπιλεγμένη διάταξη μορφοποίησης που χρησιμοποιείται για την παρουσίαση αποτελεσμάτων αναζήτησης).

WebSearch Η μονάδα αυτή χειρίζεται πρακτικά τα αιτήματα των χρηστών για την αναζήτηση συγκεκριμένων λέξεων ή φράσεων στη βάση δεδομένων. Υπάρχουν δύο είδη αναζήτησης: λέξεις και φράσεις. Επιτρέπεται η αναζήτηση πολύπλοκων boolean ερωτημάτων, αναζήτηση κανονικών εκφράσεων, ακόμη και η συνδυασμένη αναζήτηση μεταδεδομένων, αναφορών και εγγράφων. Αν δεν είναι δυνατή η εύρεση αποτελεσμάτων για το ακριβές αίτημα του χρήστη, το σύστημα προτείνει εναλλακτικά αποτελέσματα ως οδηγό αναζήτησης για το χρήστη. Οι δείκτες ευρετηρίασης της αναζήτησης σχεδιάστηκαν έτσι ώστε να παρέχουν ιδιαίτερα γρήγορες αποκρίσεις για συλλογές δεδομένων μεσαίου μεγέθους (έως 106 εγγραφών).

Το σύνολο των μεταδεδομένων είναι οργανωμένο σε συλλογές μεταδεδομένων οι οποίες είναι άμεσα προσβάσιμες μέσω της λειτουργίας περιήγησης, ακολουθώντας τη λογική των δημοφιλών καταλόγων του διαδικτύου. Μια διαφορετική οργάνωση των συλλογών των εγγράφων μπορεί να είναι διαθέσιμη στο χρήστη μέσω των εικονικών συλλογών. Για παράδειγμα ένα έγγραφο μπορεί να ταξινομηθεί σύμφωνα με το είδος τους (βιβλίο, άρθρο) αλλά και σύμφωνα με την ημερομηνία έκδοσης του. Αυτή η ευέλικτη οργάνωση των εγγράφων σε συλλογές επιτρέπουν την εύκολη πλοήγηση του χρήστη και τη δυνατότητα αναζήτησης περιεχομένου στο σύστημα.

WebStat Η μονάδα αυτή δίνει τη δυνατότητα συγκέντρωσης στατιστικών στοιχείων για την κατάσταση του εξυπηρετητή, τη χρήση του συστήματος, καθώς και για άλλα χαρακτηριστικά του συστήματος.

Αυτοτελείς λογισμικές μονάδες σχετικές με την εξατομίκευση

Το CDS Invenio μπορεί να εξατομικευθεί ώστε να ανταποκρίνεται στις ανάγκες κάθε τελικού χρήστη. Η μονάδα WebSession αναγνωρίζει τους χρήστες και τις προσωπικές τους ρυθμίσεις, η WebBasket τους δίνει τη δυνατότητα να ορίσουν και να μοιραστούν προσωπικές συλλογές εγγράφων (καλάθια), ενώ η WebAlert επιτρέπει τη δημιουργία και ρύθμιση προσωπικών ειδοποιήσεων ηλεκτρονικού ταχυδρομείου.

WebSession Η μονάδα αυτή αναλαμβάνει τη διαχείριση των χρηστών και των προσωπικών τους ρυθμίσεων κατά την περίοδο που βρίσκονται συνδεδεμένοι στο σύστημα. Φροντίζει τη διαφοροποίηση των χρηστών μεταξύ τους και χρησιμεύει στην εξατομίκευση της διεπαφής του κάθε χρήστη και στη χρήση των υπολοίπων υπηρεσιών εξατομίκευσης.

WebBasket Η μονάδα αυτή δίνει τη δυνατότητα στο χρήστη να δημιουργεί και να διαχειρίζεται προσωπικές συλλογές εγγράφων που ονομάζονται καλάθια. Ο κάθε χρήστης μπορεί να δημιουργήσει και να κατέχει πολλά καλάθια στα οποία μπορεί να αποθηκεύσει έγγραφα του ενδιαφέροντος του. Το κάθε καλάθι μπορεί να είναι προσωπικό και περιορισμένο, ή να μοιράζεται με άλλους χρήστες - μέλη κάποιας ομάδας, ή και με όλους τους χρήστες του συστήματος.

WebAlert Η μονάδα αυτή επιτρέπει στο χρήστη να λαμβάνει ειδοποιήσεις ηλεκτρονικού ταχυδρομίου κάθε φορά που καταχωρείται στο σύστημα ένα έγγραφο που ταιριάζει με τα κριτήρια τα οποία έχει ορίσει. Τα κριτήρια αυτά ανταποκρίνονται σε ένα τυπικό αίτημα αναζήτησης του χρήστη. Για παράδειγμα ο χρήστη μπορεί να ειδοποιείται κάθε φορά που ένα έγγραφο που περιέχει κάποιες συγκεκριμένες λέξεις εισάγεται στο σύστημα. Κάθε χρήστης μπορεί να ορίσει διάφορες ειδοποιήσεις, των οποίων η συχνότητα μπορεί να είναι είτε ημερήσια, είτε εβδομαδιαία, είτε μηνιαία. Ο χρήστης ενημερώνεται μέσω ηλεκτρονικού ταχυδρομείου για τα αποτελέσματα της αναζήτησης που πραγματοποίησε μια ειδοποίηση, μπορεί όμως επίσης να ορίσει κατ' επιλογήν να αποθηκεύονται αυτόματα αυτά τα δεδομένα σε ένα από τα καλάθια που του ανήκουν.

WebComment Η μονάδα αυτή αποτελεί ένα εργαλείο της κοινότητας των χρηστών του συστήματος για την ταξινόμηση των εγγράφων και την ανταλλαγή σχολίων πάνω σε αυτά.

WebMessage Η μονάδα αυτή δίνει τη δυνατότητα στους χρήστες να επικοινωνούν μεταξύ τους ανταλλάσσοντας μηνύματα καθώς και προσκλήσεις για τη συμμετοχή σε ομάδες, κ.α.

Αυτοτελείς λογισμικές μονάδες σχετικές με την ενοποίηση του συστήματος

Κάποιες μονάδες εξυπηρετούν την ενοποίηση όλων των προαναφερθεισών μονάδων με σκοπό την ολοκληρωμένη λειτουργία του συστήματος. Η μονάδα BibSched επιτρέπει τον προγραμματισμό και τη διαχείριση βιβλιογραφικών εργασιών, η WebAccess τον προσδιορισμό ρόλων, επίσης τη λειτουργία ενός συστήματος πρόσβασης στις υπηρεσίες που προσφέρει το CDS Invenio. Η WebStyle επιτρέπει τον ορισμό μια κοινής αισθητικής για το σύστημα.

BibSched Η μονάδα αυτή αποτελεί τον προγραμματιστή και διαχειριστή βιβλιογραφικών εργασιών του συστήματος. Είναι κεντρική μονάδα και επιτρέπει σε όλες τις υπόλοιπες μονάδες να έχουν πρόσβαση στα βιβλιογραφικά δεδομένα του εξυπηρετητή με προγραμματισμένο τρόπο, εμποδίζοντας την από κοινού πρόσβαση σε δεδομένα όταν αυτό δεν επιτρέπεται και φροντίζοντας τη συνεπή εκτέλεση των εργασιών ενημέρωσης της βάσης δεδομένων. Η διεπαφή διαχείρισης του προγραμματιστή εργασιών δίνει τη δυνατότητα ελέγχου και παρακολούθησης της ουράς/ακολουθίας εργασιών και παρέχει διάφορες επιλογές ανθρώπινης παρέμβασης, όπως για παράδειγμα την παύση της ουράς, την αναδιάταξή της, τον επαναπρογραμματισμό κάποιων εργασιών κ.α.

WebAccess Η μονάδα αυτή είναι υπεύθυνη για τη παροχή πρόσβασης στους χρήστες στις διάφορες λειτουργίες του συστήματος. Χρησιμοποιείται μια τεχνική ελέγχου πρόσβασης ρόλων (Role-Based Access Control - RBAC) στην οποία κάθε χρήστης ανήκει σε διάφορες ομάδες ανάλογα με το ρόλο του στο σύστημα. Σε κάθε ομάδα μπορεί να χορηγηθεί το δικαίωμα να εκτελέσει μια λειτουργία ανάλογα με την εκάστοτε χρήση της λειτουργίας αυτής. Επι του παρόντος η μονάδα WebAccess χρησιμοποιείται κυρίως για τη διεπαφή διαχείρισης του συστήματος ελέγχοντας τη διαμόρφωση των διαχειριστικών μονάδων και την εκτέλεση των σχετικών εργασιών.

WebStyle Η μονάδα αυτή ελέγχει το σύνολο των ρυθμίσεων που καθορίζουν την αισθητική του συστήματος.

3.5 Τεχνολογίες σχετικές με το CDS Invenio

Παρακάτω περιγράφονται οι κύριες τεχνολογίες στις οποίες βασίζεται και τις οποίες χρησιμοποιεί το CDS Invenio.

3.5.1 GNU/Linux

Μια εγκατάσταση του CDS Invenio βασίζεται στο λειτουργικό σύστημα GNU/Linux.Πρόκειται για ένα λειτουργικό σύστημα σύμφωνο με τις αρχές του συστήματος Unix, βασισμένο στο πυρήνα Linux, που χρησιμοποιεί λογισμικό GNU. Κατά συνέπεια, είναι ένα λειτουργικό σύστημα εξ ολοκλήρου βασισμένο στο ελεύθερο λογισμικό και λογισμικό ανοιχτού κώδικα. Ιστορικά χρησιμοποιείται κυρίως σε εξυπηρετητές λόγω της σταθερότητας και ασφάλειας που προσφέρει, ενώ η φιλοσοφία του, που επιτρέπει σε κάθε χρήστη να συμμετέχει στην ανάπτυξή του, έχει συμβάλλει ιδιαίτερα στην κατεύθυνση αυτή.

Στην περίπτωση του CDS Invenio το GNU/Linux προσφέρει το περιβάλλον και όλα εκείνα τα απαραίτητα εργαλεία για την ανάπτυξη και προγραμματισμό του λογισμικού καθώς και την λειρουργία του ως εξηπηρετητή μιας ψηφιακής βιβλιοθήκης.

3.5.2 Python

Η Python είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου γενικής χρήσης. Η βασική της φιλοσοφία είναι η αναγνωσιμότητα σε συνδυασμό με τη δυναμικότητα, και η βασική της βιβλιοθήκη είναι εκτενής και συνδυάζει αυτά τα χαρακτηριστικά. Η Python υποστηρίζει διάφορα προγραμματιστικά παραδείγματα, όπως το αντικειμενοστραφές, το προστακτικό και το συναρτησιακό. Χαρακτηριστικό της είναι η αυτόματη διαχείριση μνήμης και ως δυναμική γλώσσα προγραμματισμού μπορεί να χρησιμοποιηθεί και ως γλώσσα δέσμης ενεργειών. Λόγω της φύσης της αυτής, είναι κατάλληλη μεταξύ άλλων και για διαδικτυακές εφαρμογές.

Το σύνολο αυτών των χαρακτηριστικών κάνουν τη γλώσσα προγραμματισμού Python να αποτελεί μια εξαιρετική επιλογή για την ανάπτυξη και προγραμματισμό του CDS Invenio.

3.5.3 MySQL

Το MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων. Μέσα από αυτό ο χρήστης μπορεί να ορίσει και να διαχειριστεί ένα πλήθος βάσεων δεδομένων, και να προσφέρει πρόσβαση σε αυτές. Πρόκειται για ελεύθερο λογισμικό, το οποίο διατίθεται βάσει της άδειας GPL. Είναι ένα ιδιατερα δημοφιλές σύστημα, και έχει μεγάλη απήχηση στις διαδικτυακές εφαρμογές, συνήθως ως κομμάτι του μοντέλου ανάπτυξης LAMP (Linux - Apache - MySQL - PHP/Python/Perl). Προσφέρει διεπαφές προγραμματισμού εφαρμογών για διάφορες γλώσσες προγραμματισμού και μερικά από τα χαρακτηριστικά του είναι τα παρακάτω: εκτενής υποστήριξη της γλώσσας SQL για την υποβολή ερωτημάτων στη βάση δεδομένων, υποστήριξη διάφορων υπολογιστικών πλατφορμών, αποθηκευμένες διαδικασίες, εναύσματα, δρομείς, ενημερώσιμες όψεις, μερική υποστήριξη του διεθνούς προτύπου κωδικοποίησης Unicode, και άλλα.

Το CDS Invenio χρησιμοποιεί το σύστημα MySQL ήδη εδώ και δέκα χρόνια με επιτυχία, καθώς αποτελεί ένα παράγοντα σταθερότητας και ταχύτητας στην αποθήκευση και διάθεση των δεδομένων.

3.5.4 XML / MARCXML

Η XML είναι ουσιαστικά μια γλώσσα / σύνολο κανόνων κωδικοποίησης κειμένων ψηφιακά. Τα κύρια χαρακτηριστικά της είναι η απλότητα, η γενικότητα και η ευκολία χρήσης στο διαδίκτυο. Υποστηρίζει πλήρως το διεθνές πρότυπο κωδικοποίησης Unicode, και προορίζεται κυρίως για κείμενα, αλλά χρησιμοιείται ευρέως και για την απεικόνιση άλλων δομών δεδομένων. Στο διαδίκτυο συχνά χρησιμοποιείται για την ανταλλαγή δεδομένων. Υπάρχουν πολλές προγραμματιστικές διεπαφές για την επεξεργασία και ανάλυση της ως το μέσο επεξεργασίας κειμένων.

Μια χρήση της είναι στην επιστήμη της πληροφορίας και των βιβλιοθηκών. Η δομή MARCXML, βασισμένη στο πρότυπο MARC21, υπογορεύει τη σύνταξη βιβλιογραφικών μεταδεδομένων στη γλώσσα XML.

Το CDS Invenio χρησιμοποιεί τη δομή MARCXML για τη διαχείριση, ανταλλαγή και μετατροπή μεταδεδομένων, εκμεταλλευόμενο τις δυνατότητες που προσφέρει η XML.

3.5.5 OAI / OAI-PMH

Το Open Archives Initiative (OAI) αποτελεί μια προσπάθεια προώθησης προτύπων διαλειτουργικότητας που αποσκοπούν στη διευκόλυνση της αποτελεσματικής διάδοσης του περιεχομένου. Έχει τις ρίζες του στο κίνημα της ελεύθερης πρόσβασης και των ακαδημαϊκών και επιστημονικών αποθετηρίων. Η συνέχιση της υποστήριξης αυτής της προσπάθειας εξακολουθεί να αποτελεί τον ακρογωνιαίο λίθο του Open Archives Initiative.Σήμερα, ωστόσο, το έργο του OAI έχει επεκταθεί για να προωθήσει την ευρεία πρόσβαση σε ψηφιακούς πόρους.

Ένα από τα κύρια έργα της πρωτοβουλίας είναι το πρωτόκολλο για τη συγκομιδή μεταδεδομένων, Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), που αποτελεί ένα μηχανισμό διαλειτουργικότητας των αποθετηρίων. Οι πάροχοι μεταδεδομένων μπορούν αρχικά να εκθέσουν και να διαθέσουν μεταδεδομένα μέσω των κανόνων που ορίζει το πρωτόκολλο, ενώ οι πάροχοι υπηρεσιών στη συνέχεια μπορούν να πραγματοποιήσουν αιτήσεις συγκομιδής τους, ώστε να διαθέσουν τελικά ένα σύνολο μεταδεδομένων προερχόμενα από διάφορους παρόχους. Οι αιτήσεις αυτές γίνονται στη γλώσσα XML με βάση το πρωτόκολλο επικοινωνίας HTTP.

Το CDS Invenio δύναται να χρησιμοποιήσει το πρωτόκολλο ώστε να εκθέσει το σύνολο των μεταδεδομένων των εγγραφών μιας εγκατάστασής του.

Κεφάλαιο 4

Αναζήτηση

4.1 Μεθοδολογία

Σε αυτήν την ενότητα αναλύεται ο υπάρχων τρόπος λειτουργίας της μονάδας WebSearch (4.1.1, σελίδα 41) και περιγράφονται οι αλλαγές που προτάθηκαν (4.1.2, σελίδα 47) ώστε να είναι δυνατή η κατανεμημένη αναζήτηση σε εξυπηρετητές άλλων ψηφιακών βιβλιοθηκών.

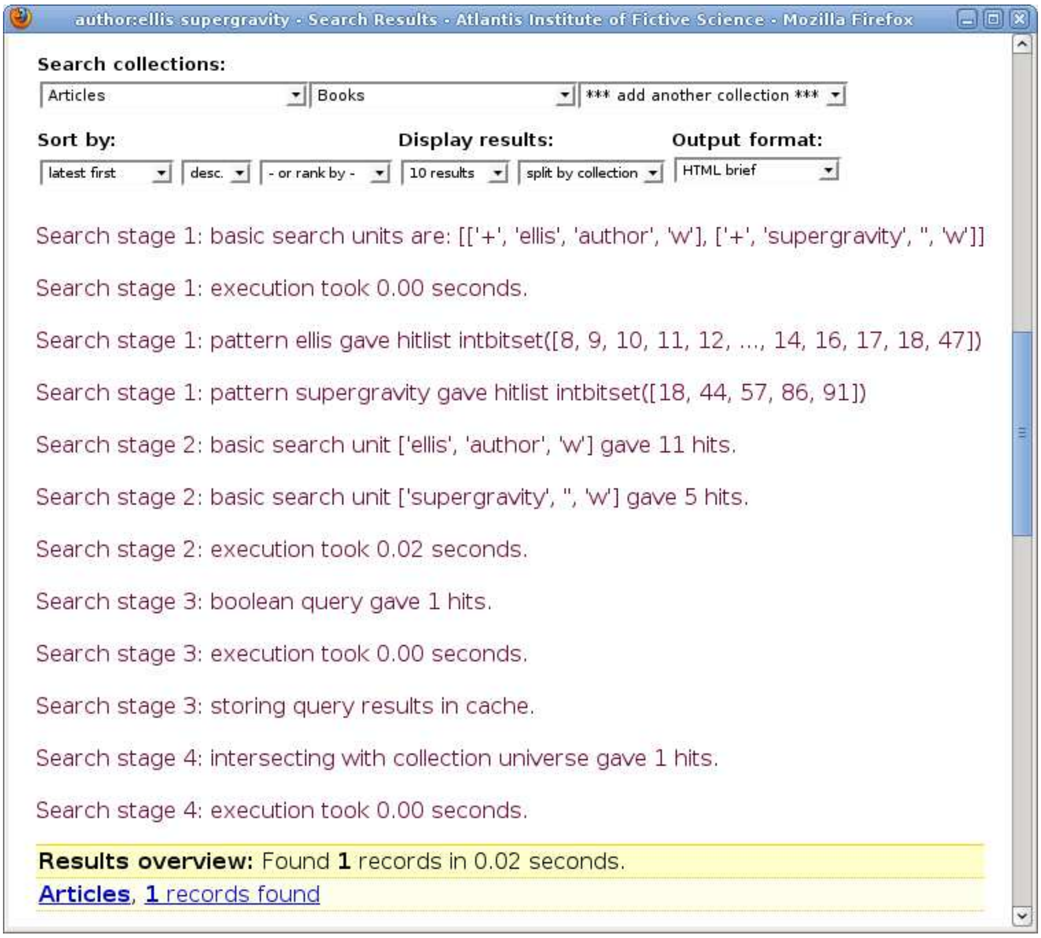
4.1.1 Ανάλυση

Στην ενότητα 3.4.5, σελίδα 35, περιγράφηκαν οι βασικές αρχές λειτουργίας της αυτόνομης λογισμικής μονάδας αναζήτησης του CDS Invenio, WebSearch. Σε μια τυπική περίπτωση, ο χρήστης εισάγει μια λέξη ή φράση προς αναζήτηση και η μηχανή αναζήτησης αναλαμβάνει να του επιστρέψει το συντομότερο δυνατόν τα αποτελέσματα που πλησιάζουν περισσότερο στο ερώτημά που έχει τεθεί. Αυτή η διαδικασία έχει έξι στάδια, τα οποία περιγράφονται παρακάτω.

Τα έξι στάδια της αναζήτησης

Αρχικώς, το ερώτημα του χρήστη χωρίζεται στα βασικά ερωτήματα προς αναζήτηση και στις επιπλέον επιλογές αναζήτησης. Τα βασικά ερωτήματα είναι μια λίστα από ένα ή και περισσότερα ζεύγη, κάθε ένα από τα οποία αποτελείται από τη λέξη/φράση και το πεδίο στο οποίο αναφέρεται η προς αναζήτηση λέξη/φράση, όπως για παράδειγμα το ζευγάρι Παπαδόπουλος, συγγραφέας ή το ζευγάρι Πρωτόνια, τίτλος. Τα ζεύγη μεταξύ τους ενώνονται με λογικές πράξεις, όπως το ¨και¨, το ¨ή¨ και το ¨και όχι¨. Οι επιπλέον επιλογές αναζήτησης αποτελούνται από τις συλλογές εγγραφών στις οποίες επιθυμεί ο χρήστης να πραγματοποιηθεί η αναζήτηση (για παράδειγμα: Άρθρα και Βιβλία) και τα οποιαδήποτε επιπλέον κριτήρια αναζήτησης όπως η ημερομηνία υποβολής του εγγράφου, η γλώσσα του εγγράφου και άλλα.

Στη συνέχεια, για κάθε ένα από τα ζεύγη των βασικών ερωτημάτων του πρώτου βήματος, επαληθεύεται ότι υπάρχουν αποτελέσματα ανεξαρτήτως των όποιων επιπλέον επιλογών αναζήτησης έχει ορίσει ο χρήστης. Πρακτικά, επαληθεύεται ότι στο ευρετήριο του πεδίου του ζεύγους υπάρχει η λέξη/φράση προς αναζήτηση. Στην περίπτωση που επαληθευτεί η ύπαρξη αποτελεσμάτων, το ζεύγος αυτό προστίθεται στη λίστα των ζευγών που έχουν δώσει αποτελέσματα. Αν η εύρεση αποτελεσμάτων δεν κατέστη δυνατή τότε η λέξη/φράση διασπάται σε ακόμη μικρότερα μέρη, εάν αυτό είναι δυνατό, ώστε να μην περιέχουν μη αλφαριθμητικούς χαρακτήρες, και η αναζήτηση πραγματοποιείται για κάθε ένα από αυτά. Τέλος, αν και αυτή η μέθοδος αποτύχει, προτείνονται στο χρήστη άλλες λέξεις/φράσεις, κοντινές στις αρχικές. Μετά την επεξεργασία όλων των ζευγών των βασικών ερωτημάτων η μηχανή αναζήτησης περνά στο επόμενο στάδιο.



Σχήμα 4.1: Απεικόνιση των σταδίων της αναζήτησης

Στο τρίτο στάδιο επεξεργάζονται τα ζεύγη τα οποία είναι επαληθευμένο ότι αποφέρουν αποτελέσματα. Πιο συγκεκριμένα γίνονται μεταξύ τους οι λογικές πράξεις τις οποίες έχει ορίσει ο χρήστης, με προτεραιότητα από τα αριστερά προς τα δεξιά. Αν μετά από αυτό το στάδιο δεν υπάρχουν αποτελέσματα ο χρήστης ενημερώνεται για την ύπαρξη επιμέρους αποτελεσμάτων και του προτείνεται η προσαρμογή της αρχικής του αναζήτησης.

Σε περίπτωση που σε αυτό το σημείο υπάρχουν ακόμη αποτελέσματα, που περιγράφονται ως αναγνωριστικά εγγραφών, ελέγχεται αν ανήκουν στις συλλογές εγγραφών τις οποίες έχει ορίσει αρχικά ο χρήστης. Πρακτικά πραγματοποιείται μια τομή των τρεχόντων αποτελεσμάτων (αναγνωριστικών των εγγραφών) με όλα τα περιεχόμενα κάθε συλλογής (τα αναγνωριστικά των εγγραφών που ανήκουν στην κάθε συλλογή). Αν η τομή αποφέρει αποτελέσματα η αναζήτηση συνεχίζεται στο επόμενο βήμα. Αν η τομή δεν αποφέρει αποτελέσματα σημαίνει ότι υπάρχουν αποτελέσματα σε κάποια από τις υπόλοιπες συλλογές εγγραφών που δεν επέλεξε ο χρήστης, οι οποίες του προτείνονται, και σε αυτό το σημείο.

Στο πέμπτο στάδιο της αναζήτησης εφαρμόζονται στα αποτελέσματα οι όποιες επιπλέον επιλογές αναζήτησης όρισε ο χρήστης στην αρχή. Και σε αυτή την περίπτωση πραγματοποιείται τομή των αποτελεσμάτων με το σύνολο των εγγραφών που εμπίπτουν σε κάθε μια από τις επιπλέον επιλογές. Αν μετά την τομή δεν υπάρχουν αποτελέσματα, ο χρήστης ειδοποιείται και του προτείνεται η ίδια αναζήτηση χωρίς τις επιπλέον επιλογές. Στην περίπτωση που και μετά από αυτό το στάδιο υπάρχουν αποτελέσματα η διαδικασία συνεχίζεται με το έκτο στάδιο.

Στο έκτο και τελευταίο στάδιο τα αποτελέσματα παρουσιάζονται στο χρήστη.

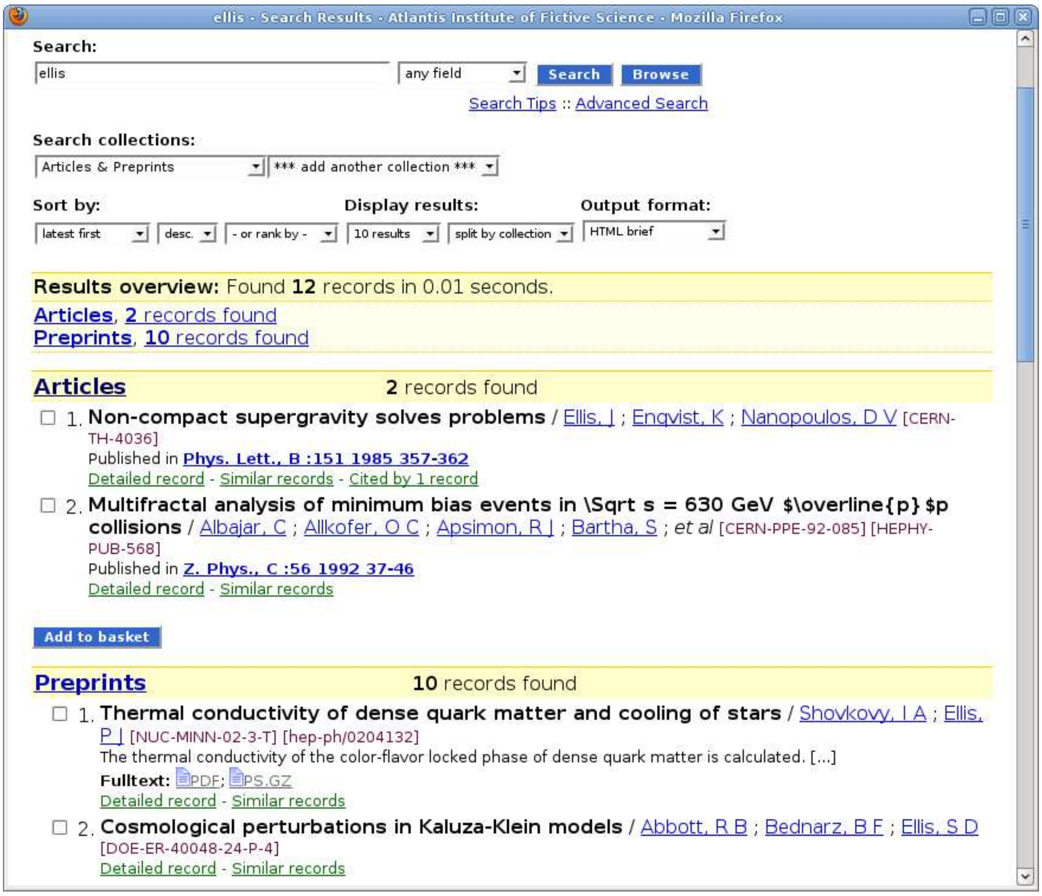
Στο σχήμα 4.1, σελίδα 42, απεικονίζεται μια τυπική αναζήτηση με φανερά τα διάφορα στάδια, ενώ στο σχήμα 4.2, σελίδα 44, φαίνονται τα αποτελέσματα μιας τυπικής αναζήτησης ακριβώς όπως παρουσιάζονται στο χρήστη στο έκτο και τελευταίο στάδιο.

Συλλογές εγγραφών

Στις προηγούμενες ενότητες έγιναν κάποιες αναφορές στις συλλογές εγγραφών στις οποίες ανήκει το σύνολο του εγγράφων του συστήματος. Στην προηγούμενη υποενότητα έγινε περιγραφή μιας τυπικής αναζήτησης και έγινε αναφορά στο πώς οι συλλογές εγγραφών αποτελούν επιπλέον επιλογές αναζήτησης για το χρήστη. Στις επόμενες παραγράφους θα γίνει μια περιγραφή της χρήσης των συλλογών αυτών στο CDS Invenio.

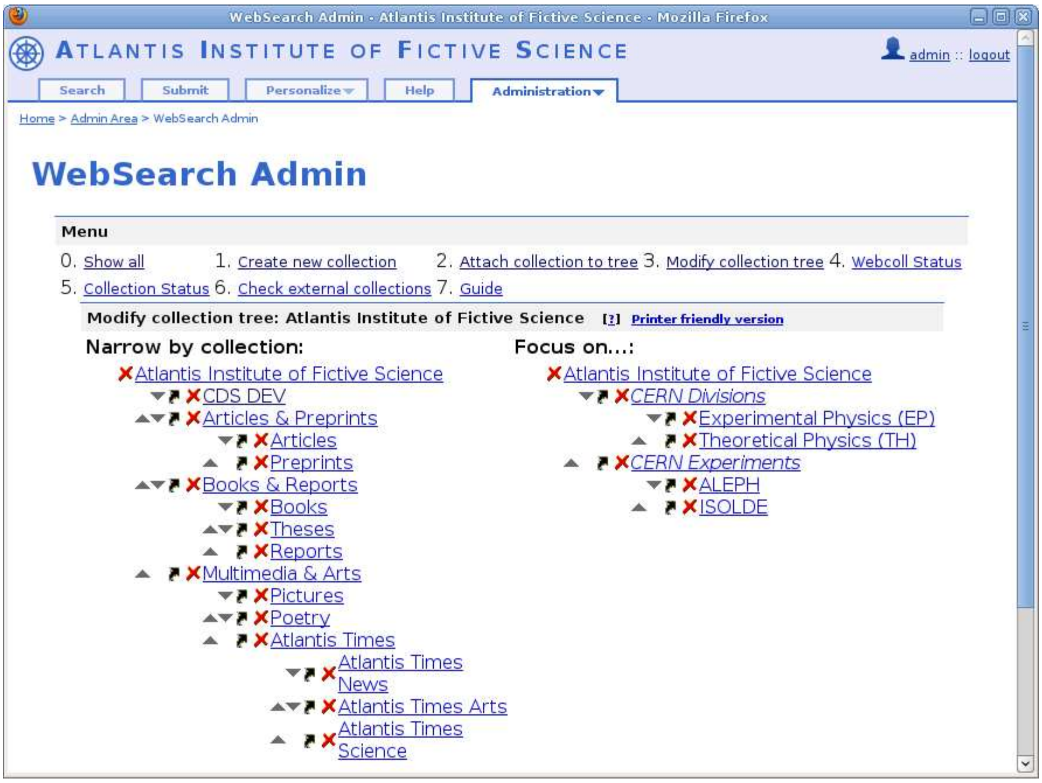
Το σύνολο των εγγραφών του συστήματος σε μια εγκατάσταση του CDS Invenio οργανώνεται σε ξεχωριστές ομάδες, που ονομάζονται συλλογές εγγραφών. Αυτές οι συλλογές είναι οργανωμένες σε ένα δέντρο, η δομή του οποίου χρησιμοποιείται και στη διεπαφή του χρήστη για την πλοήγηση και αναζήτηση σε κάθε ξεχωριστή συλλογή. Η αρχή του δέντρου είναι η πρώτη συλλογή που ορίζεται, και κατά συνέπεια η κυρίως. Όλες οι υπόλοιπες συλλογές ξεκινούν από αυτήν, ως κλαδιά-παιδιά της. Ένα παράδειγμα δέντρου φαίνεται στο σχήμα 4.3, σελίδα 45. Υπάρχουν δύο είδη συλλογών: οι κανονικές και οι εικονικές. Οι διαφορές τους περιγράφονται παρακάτω.

Ο χρήστης μπορεί να ορίσει διάφορες παραμέτρους για την κάθε συλλογή, με βασικές τις δύο παρακάτω: (α) το σύνολο των εγγραφών που ανήκουν στη συλλογή



Σχήμα 4.2: Παρουσιάση στο χρήστη των αποτελεσμάτων μιας τυπικής αναζήτησης

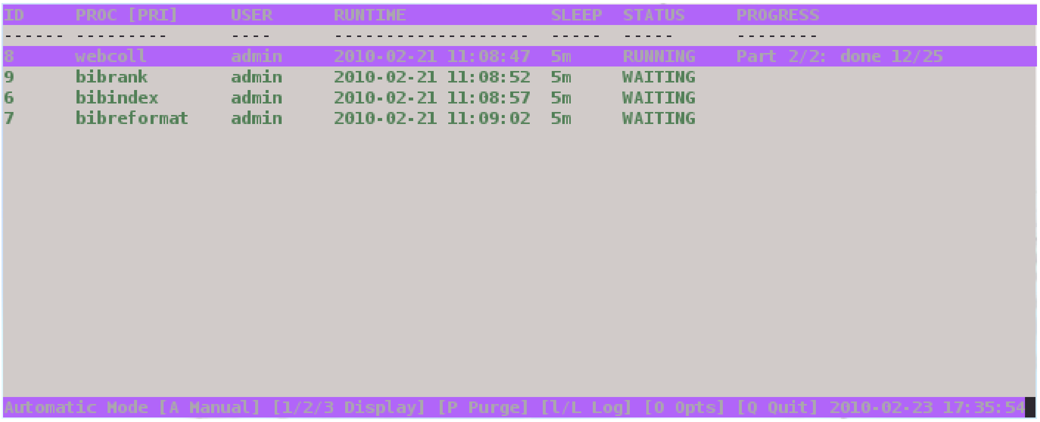
και (β) τη θέση της συλλογής στο δέντρο. Για να ορίσει το σύνολο των εγγραφών που ανήκουν στη συλλογή ο χρήστης αρκεί να ορίσει το ερώτημα της συλλογής (collection query). Το ερώτημα της συλλογής είναι ουσιαστικά το ερώτημα της αναζήτησης το οποίο θα είχε ως αποτέλεσμα όλες τις εγγραφές που θέλουμε να ανήκουν στη συλλογή. Είναι ουσιαστικά δηλαδή ο ορισμός ενός πεδίου και μια τιμή για το πεδίο αυτό. Για παράδειγμα, το ερώτημα μια συλλογής θα μπορούσε να είναι `author:Papadopoulos`. Στο CDS Invenio είθισται να χρησιμοποιείται ως το πεδίο που ορίζει τα περιεχόμενα κάθε συλλογής το πεδίο 980 της δομής MARC21, με όνομα `collection`. Για παράδειγμα μια συλλογή άρθρων θα είχε ως ερώτημα το `collection:articles`. Αν δεν οριστεί το ερώτημα κάποιας συλλογής, τότε τα περιεχόμενα της καθορίζονται ως το σύνολο των περιεχομένων των συλλογών-παιδιών της. Σε αυτό το σημείο γίνεται προφανής η σημασία της θέσης της συλλογής στο δέντρο. Εκτός δηλαδή από τη διευκόλυνση της πλοήγησης του τελικού χρήστη στο σύνολο των συλλογών ορίζεται και το περιεχόμενο των ίδιων των συλλογών.



Σχήμα 4.3: Συλλογές οργανωμένες σε δέντρο

Όπως αναφέρθηκε προηγουμένως, υπάρχουν δύο είδη συλλογών: οι κανονικές και οι εικονικές. Πιο συγκεκριμένα θα μπορούσαμε να πούμε ότι πρόκειται για είδη συγγένειας συλλογών. Θα ορίσουμε τα δύο είδη δίνοντας ένα παράδειγμα, τονίζοντας έτσι και τις διαφορές τους. Αν μια συλλογή Α έχει δύο παιδιά-κλαδιά, τις συλλογές Β και Γ, και η σχέση μεταξύ Α και Β και Α και Γ είναι κανονική, τότε κάθε έγγραφο που ανήκει στη συλλογή Α θα ανήκει είτε στη συλλογή Β, είτε στη συλλογή Γ, και το σύνολο των εγγράφων της συλλογής Α είναι το άθροισμα των εγγραφών της συλλογής Β και των αντιστοίχων της συλλογής Γ. Για παράδειγμα, η συλλογή Α ονομάζεται ¨Πολυμέσα¨, η συλλογή Β ¨Φωτογραφίες¨ και η συλλογή Γ ¨Βίντεο¨. Αν θέλαμε τώρα να ορίσουμε, για τη διευκόλυνση του τελικού χρήστη, ένα υποσύνολο της συλλογής Α που να περιέχει όλο το υλικό, φωτογραφίες και βίντεο, που αφορά για παράδειγμα αθλητικά γεγονότα, θα ορίζαμε μια νέα συλλογή Δ με εικονική σχέση με τη συλλογή Α η οποία θα συγκέντρωνε όλα τα σχετικά περιεχόμενα (προφανώς υποσύνολα των συλλογών Β και Γ). Στην περίπτωση των εικονικών συλλογών δεν ισχύει ότι το άθροισμα των εγγραφών των εικονικών συλλογών-παιδιών μιας συλλογής αποτελεί και το συνολικό περιεχόμενο της συλλογής αυτής. Από τη σχεδίασή τους, οι εικονικές συλλογές αποτελούν ένα βοήθημα για την διευκόλυνση της πλοήγησης του τελικού χρήστη στο σύνολο των εγγραφών της ψηφιακής βιβλιοθήκης.

Στις προηγούμενες ενότητες τονίσθηκε η σημασία της ταχείας απόκρισης του εξυπηρετητή στις αναζητήσεις των χρηστών. Εκτός από το σημαντικό ρόλο του μηχανισμού ευρετηρίασης του CDS Invenio προς το σκοπό αυτό, σημαντικό παράγοντα αποτελεί και ο προ-υπολογισμός του περιεχομένου κάθε συλλογής. Όπως είδαμε στην προηγούμενη υποενότητα, το τέταρτο στάδιο του μηχανισμού αναζήτησης αφορά την τομή των μέχρι στιγμής αποτελεσμάτων με το σύνολο των εγγραφών των επιλεγμένων συλλογών. Τον υπολογισμό αυτό αναλαμβάνει η υπηρεσία WebColl της μηχανής αναζήτησης.



Σχήμα 4.4: Η υπηρεσία WebColl εκτελείται μέσω του προγραμματιστή της μονάδας BibSched

Η υπηρεσία WebColl είθισται να ορίζεται ως μια περιοδική εργασία που προγραμματίζεται μέσω της μονάδας BibSched και εκτελείται σε τακτά χρονικά διαστήματα. Ένα παράδειγμα φαίνεται στο σχήμα 4.4, σελίδα 46. Η εν λόγω υπηρεσία αναλαμβάνει να υπολογίζει, σύμφωνα με το ερώτημα αναζήτησης της κάθε συλλογής, όλες τις εγγραφές που της ανήκουν. Εν συνεχεία αποθηκεύει στη βάση δεδομένων μια λίστα[1] με τα αναγνωριστικά των εγγραφών αυτών, καθώς και το πλήθος τους. Την ίδια στιγμή, για λόγους ταχύτητας απόκρισης της διεπαφής, προ-υπολογίζονται και αποθηκεύονται στη βάση δεδομένων οι ιστοσελίδες περιγραφής της κάθε συλλογής, μια σύντομη λίστα δηλαδή με τις τελευταίες προσθήκες για κάθε μία.

[1]Πρόκειται για ένα αντικείμενο IntBitSet, ένα σύνολο δηλαδή ακεραίων βασισμένο σε μια συστοιχία bit. Στην περίπτωση του CDS Invenio η λίστα αυτή είναι υλοποιημένη στη γλώσσα προγραμματισμού C για λόγους ταχύτητας και εισάγεται στη γλώσσα προγραμματισμού Python μέσω ενώ μεταφραστή (wrapper). Κάθε ακέραιος αναπαριστάται από ένα bit και τη θέση του στη συστοιχία. Το αντικείμενο IntBitSet, περιέχει ακόμη πληροφορίες όπως το πλήθος των ακεραίων που φέρει καθώς και την ποσότητα μνήμης που καταλαμβάνει. Οι πράξεις μεταξύ IntBitSet είναι λογικές (και, ή, όχι και, και άλλα) και γίνονται μεταξύ λέξεων (words) των 64 bit η κάθε μία.

Εξωτερικές συλλογές εγγραφών

Πέραν των συλλογών που περιγράφηκαν στην προηγούμενη υποενότητα, το CDS Invenio προσφέρει τη δυνατότητα ορισμού εξωτερικών συλλογών εγγραφών. Οι συλλογές αυτές, περιορισμένων δυνατοτήτων, επιτρέπουν στο διαχειριστή να ορίσει επιπλέον πηγές αναζήτησης για το χρήστη. Πρακτικά, μια εξωτερική συλλογή είναι μια ψηφιακή βιβλιοθήκη, ή οποιαδήποτε δικτυακή υπηρεσία, η οποία προσφέρει ελεύθερα μια διαδικτυακή διεπαφή. Το αρχικό ερώτημα αναζήτησης του χρήστη στέλνεται, μέσω του πρωτοκόλλου Hypertext Transfer Protocol (HTTP), στην εξωτερική συλλογή (τυπικά μια ιστοσελίδα) και τα αποτελέσματα παρουσιάζονται αυτούσια μαζί με τα αποτελέσματα της ¨τοπικής¨ αναζήτησης. Συγχρόνως δίνεται στο χρήστη ο διαδικτυακός σύνδεσμος προς την εκάστοτε ιστοσελίδα.

Οι εξωτερικές συλλογές παραμένουν ανεξάρτητες από τις ¨τοπικές¨ συλλογές και ουδεμία πληροφορία για αυτές αποθηκεύεται στη βάση δεδομένων

4.1.2 Σχεδίαση

Σκοπός της σχεδίασης είναι η αναζήτηση σε ένα κανανεμημένο δίκτυο ψηφιακών βιβλιοθηκών. Οι ψηφιακές βιβλιοθήκες μπορεί να είναι κατανεμημένες ανά τον κόσμο εξυπηρετώντας διάφορα ινστιτούτα, πανεπιστήμια και οργανισμούς. Τα περιεχόμενά τους μπορεί να αφορούν είτε τα ίδια θέματα προσφέροντας διαφορετικές αλλά συμπληρωματικές πληροφορίες, είτε διαφορετικά θέματα, υποσύνολα ενός υπερ-θέματος. Οι φηφιακές βιβλιοθήκες μπορεί να ανήκουν φυσικά στον ίδιο οργανισμό εξυπηρετώντας διαφορετικές βιβλιογραφικές ανάγκες και απαιτήσεις αυτού του οργανισμού και προσφέροντας έτσι μια κατανομή πόρων εντός αυτού. Σε κάθε περίπτωση, ένα κανανεμημένο δίκτυο συνεργαζόμενων φηφιακών βιβλιοθηκών εγγυάται την ανεξαρτησία διαχείρισης και λειτουργίας ανάμεσα στα μέλη του ενώ συγχρόνως διασφαλίζεται η συνεργασία και κατανομή πληροφορίας μεταξύ τους.

Ένα παράδειγμα των παραπάνω μπορούν να αποτελούν δύο ψηφιακές βιβλιοθήκες φυσικής υψηλής ενέργειας, μια στη Νέα Υόρκη των Η.Π.Α. και μια στην Αθήνα. Οι βιβλιοθήκες αυτές ανήκουν σε διαφορετικούς φορείς, διαχειρίζονται από διαφορετικές ομάδες, βρίσκονται σε διαφορετικά μέρη του κόσμου και υπό διαφορετικές ζώνες ώρας, μπορούν όμως να συνεργάζονται σε ένα κατανεμημένο δίκτυο παρέχοντας μια από κοινού υπηρεσία αναζήτησης. Στο παραπάνω παράδειγμα μπορούν να προστεθούν κι άλλες ψηφιακές βιβλιοθήκες χωρίς να αλλάξει η αντιμετώπιση. Ένα άλλο παράδειγμα μπορούν να αποτελούν δύο ψηφιακές βιβλιοθήκες εντός του ίδιου οργανισμού, μία για τα τρέχοντα έγγραφα και μια για τα αρχεία του οργανισμού. Για λόγους κατανομής πόρων εντός του οργανισμού αλλά και κανόνων διαχείρισης, ενδέχεται αυτές οι δύο βιβλιοθήκες να φιλοξενούνται και να διαχειρίζονται από ξεχωριστές ομάδες και πόρους αλλά στα πλαίσια της μεταξύ τους συνεργασίας να προσφέρουν από κοινού ένα περιβάλλον αναζήτησης για τον τελικό χρήστη.

Σημειώνεται εδώ πως σκοπός της σχεδίασης μας δεν είναι η πλήρης ευρετηρίαση του συνολικού περιεχομένου όλων των μελών ενός πιθανού δικτύου, αλλά η αναζήτηση στο κατανεμημένο τους συνεργαζόμενο δίκτυο. Η λογική αυτή βασίζεται στην ανταλλαγή μεταδεδομένων μεταξύ τους. Τελικός σκοπός μας είναι να παραμείνουν ξεχωριστές οι ψηφιακές βιβλιοθήκες μεταξύ τους, προσφέροντας μια από κοινού αναζήτηση. Στις παρακάτω παραγράφους θα γίνει μια περιγραφή του πώς επιτυγχάνεται αυτό επεκτείνοντας την υπάρχουσα μηχανή αναζήτησης του CDS Invenio.

Όπως έχει αναφερθεί εκτενώς στις προηγούμενες παραγράφους, όλες οι εγγραφές και τα δεδομένα στον εξυπηρετητή είναι οργανωμένα σε συλλογές. Επίσης, έχουν αναφερθεί οι εξωτερικές συλλογές, ως συλλογές περιορισμένων δυνατοτήτων για τον ορισμό επιπλέον επιλογών αναζήτησης. Για τον ορισμό ενός μέλους του κατανεμημένου δικτύου συνεργαζόμενων ψηφιακών βιβλιοθηκών αποφασίστηκε η εισαγωγή ενός νέου τύπου συλλογής, η φιλοξενούμενη συλλογή (hosted collection), με ιδιότητες και από τους δύο υπάρχοντες τύπους. Μια φιλοξενούμενη συλλογή μπορεί να οριστεί δηλαδή σαν μια ακόμη ¨τοπική¨ συλλογή και να τοποθετηθεί στο δέντρο των συλλογών ενώ συγχρόνως χρησιμοποιεί κάποιες από τις τεχνικές των εξωτερικών συλλογών, αφού τις επεκτείνει, για την αναζήτηση αποτελεσμάτων σε αυτήν.

Κατά τον ορισμό της μια τοπικής συλλογής υπάρχουν, όπως έχει πρωτοαναφερθεί, δύο σημαντικές επιλογές διαχείρισης: το ερώτημα αναζήτησης της συλλογής και η θέση της στο δέντρο. Το ερώτημα αναζήτησης μιας φιλοξενούμενης συλλογής θα πρέπει να ορίζει ότι πρόκειται όντως για φιλοξενούμενη και όχι για τοπική και επίσης ενδεχομένως το πεδίο αναζήτησης στην ψηφιακή βιβλιοθήκη την οποία στην πράξη αναπαριστά. Η θέση μιας φιλοξενούμενης συλλογής στο δέντρο των συλλογών, όπως και για οποιαδήποτε τοπική, καθορίζει το πως γίνεται η πλοήγηση σε αυτήν στη διεπαφή αναζήτησης καθώς και την ύπαρξη ή όχι παιδιών-κλαδιών της. Είναι κατανοητό ότι, από σχεδίασης της, μια φιλοξενούμενη συλλογή δεν έχει νόημα να έχει δικά της παιδιά-κλαδιά εφόσον εξ' ορισμού το σύνολο των περιεχομένων των παιδιών-κλαδιών μια συλλογής στο δέντρο αποτελούν και το συνολικό περιεχόμενο της ίδιας της συλλογής. Αυτό όμως για μια φιλοξενούμενη συλλογή δεν ισχύει, εφόσον τα περιεχόμενα της υπολογίζονται με διαφορετικό τρόπο, μέσω του κατενεμημένου δικτύου, που μια από τις επεκταμένες ιδιότητες των εξωτερικών συλλογών. Επιπλέον, οι φιλοξενούμενες συλλογές έχουν σχεδιαστεί να ορίζονται ως κανονικές, όχι ως εικονικές, εφόσον σκοπός τους δεν είναι να προσφέρουν μια πιο ειδική αναζήτηση στις ήδη υπάρχουσες συλλογές, αλλά μια ξεχωριστή νέα συλλογή.

Όπως αναφέρθηκε και προηγουμένως, σκοπός ύπαρξης των φιλοξενούμενων συλλογών δεν είναι η ευρετηρίαση τους, ούτε η αποθήκευση του πλήρους περιεχομένου τους στην ψηφιακή βιβλιοθήκη που τις φιλοξενεί. Συνεπώς, κατά τη διάρκεια της εκτέλεσης της υπηρεσίας WebColl δεν αποθηκεύεται στον εξυπηρετητή το σύνολο των αναγνωριστικών των εγγραφών που ανήκουν σε μια φιλοξενούμενη συλλογή, παρά μόνο το πλήθος τους. Αυτό άλλωστε δε θα είχε νόημα, καθώς τα αναγνωριστικά εγγραφών για κάθε ψηφιακή βιβλιοθήκη είναι τοπικού ενδιαφέροντος και δεν δύναται να εγγυηθεί σε καμία περίπτωση τη μοναδικότητά τους σε περίπτωση ενός δικτύου ψηφιακών βιβλιοθηκών.

Τέλος, οι απαραίτητες μετατροπές και εισαγωγές πραγματοποιούνται στα έξι προαναφερθέντα στάδια της αναζήτησης, ώστε τα αποτελέσματα των φιλοξενούμενων συλλογών να υπολογίζονται και να παρουσιάζονται μαζί με τα υπάρχοντα τοπικά. Οι μετατροπές αυτές γίνονται με τέτοιο τρόπο ώστε η αρχική διαδικασία σε περίπτωση απουσίας φιλοξενούμενων συλλογών να παραμένει άθικτη.

4.2 Υλοποίηση

Στην προηγούμενη ενότητα έγινε μια ανάλυση της λειτουργίας της μηχανής αναζήτησης του CDS Invenio και στη συνέχεια περιγράφηκε η σχεδίαση μιας υλοποίησης για την αναζήτηση σε ένα κατενεμημένο δίκτυο συνεργαζόμενων ψηφιακών βιβλιοθηκών. Στην παρούσα ενότητα θα παρουσιαστεί αναλυτικά η υλοποίηση αυτή.

4.2.1 Ορισμός και κλάσεις μια φιλοξενούμενης συλλογής

Όπως περιγράφηκε στην υποενότητα 4.1.2, σελίδα 47, ο νέος τύπος συλλογών, δηλαδή οι φιλοξενούμενες συλλογές, διατηρούν ιδιότητες και από τους δύο ήδη υπάρχοντες τύπους συλλογών. Για τον ορισμό μιας φιλοξενούμενης συλλογής ξεκινάμε ορίζοντας μια τοπική συλλογή όπως θα κάναμε κανονικά. Αυτό γίνεται μέσω της διεπαφής διαχείρισης του CDS Invenio. Συγχρόνως επιλέγουμε σε ποιο σημείο του δέντρου θα τοποθετηθεί η νέα συλλογή και το είδος/σχέση της με το δέντρο (κανονική/εικονική). Όπως αναφέρθηκε στην υποενότητα 4.1.2, σελίδα 47, είναι θεμιτό οι φιλοξενούμενες συλλογές να είναι κανονικές και δίχως παιδιά-κλαδιά. Η διαδικασία αυτή ορισμού φαίνεται στο σχήμα 4.5, σελίδα 50. Σε αυτό το σημείο θέτουμε και το ερώτημα αναζήτησης της συλλογής, το οποίο πρέπει να ξεκινάει με τη φράση `hostedcollection:`. Αυτό είναι και πρακτικά το σημείο στο οποίο διαφοροποιείται μια φιλοξενούμενη συλλογή από μια κανονική στη διαμόρφωσή της. Παρακάτω θα δούμε πώς μπορούμε να το εκμεταλλευτούμε αυτό. Να σημειωθεί εδώ πως ύστερα από τη φράση `hostedcollection:` μπορούμε να προσθέσουμε κι άλλες λέξεις ή φράσεις, οι οποίες θα μπορούσαν να χρησιμοποιηθούν για την περαιτέρω διαμόρφωση της συλλογής, αν και στην παρούσα υλοποίηση κάτι τέτοιο δεν είναι ακόμη γεγονός.

Στη συνέχεια πρέπει να ορίσουμε τη φιλοξενούμενη συλλογή ως μια επεκταμένη εξωτερική συλλογή. Οι εξωτερικές συλλογές ορίζονται ουσιαστικά σαν ένα ζεύγος κλειδιού-τιμής ενός Python dictionary. Αυτό το Python dictionary με τη σειρά του ορίζεται ως μια μεταβλητή (CFG_EXTERNAL_COLLECTIONS) στο αρχείο `websearch_external_collections_config.py`. Στη συνέχεια η μεταβλητή αυτή μπορεί να εισαχθεί σε άλλα αρχεία για χρήση των εξωτερικών συλλογών που ορίζει. Παρακάτω ακολουθεί ένα δείγμα ορισμού μιας φιλοξενούμενης συλλογής.

```
{...
  'Testing collection':
    {'engine': 'CDSInvenio',
      'base_url': 'http://cdsdev.cern.ch/',
      'parser_params':
```

Σχήμα 4.5: Ορισμός μια φιλοξενούμενης συλλογής και προσθήκης της στο δέντρο των συλλογών

```
        {'host': 'cdsdev.cern.ch',
         'path': '',
         'parser': CDSInvenioXMLExternalCollectionResultsParser,
         'fetch_format': 'xm',
         'num_results_regex_str': r'<!-- Search-Engine-Total-Number-Of-
             Results: ([0-9,]+?) -->',
         'nbrecs_regex_str': r'<!-- Search-Engine-Total-Number-Of-Results
             : ([0-9,]+?) -->',
         'nbrecs_url': 'http://cdsdev.cern.ch/search?rg=0&of=xm'},
     'search_url': 'http://cdsdev.cern.ch/search?p=',
     'record_url': 'http://cdsdev.cern.ch/record/',
     'selected_by_default': True},
...}
```

Ουσιαστικά το ζεύγος κλειδιού-τιμής αποτελείται από το όνομα της συλλογής ως το κλειδί και ακόμη ένα Python dictionary με τις ρυθμίσεις διαμόρφωσης της συλλογής ως την τιμή του κλειδιού. Σε αυτό το σημείο πρέπει να τονιστεί ότι το κλειδί, το όνομα της συλλογής δηλαδή, είναι μοναδικό και θα πρέπει να συμφωνεί με το όνομα που δόθηκε στη συλλογή κατά τον προηγούμενο ορισμό της ως ¨τοπικής¨ συλλογής. Η τιμή του κλειδιού περιέχει τις περισσότερες ρυθμίσεις διαμόρφωσης της φιλοξενούμενης συλλογής. Η συγκέντρωση των ρυθμίσεων σε αυτό το σημείο επιτρέπει την εύκολη αλλαγή και προσαρμογή τους σε περίπτωση που χρειαστεί χωρίς να είναι αναγκαία η προσαρμογή άλλων αρχείων, και κυρίως κώδικα και συναρτήσεων. Ακολουθεί μια επεξήγηση των διάφορων ρυθμίσεων.

* `engine`: Το όνομα της μηχανής αναζήτησης για τη φιλοξενούμενη συλλογή (συνάδει με το όνομα της αντίστοιχης κλάσης Python).

* `base_url`: Η βασική διαδικτυακή διεύθυνση του εξυπηρετητή, χρησιμοποιείται για την κατασκευή συνδέσμων προς αυτόν.

* `parser_params`: Οι παράμετροι που θα περαστούν στην κλάση του αναλυτή. Ορίζοντας αυτές τις παραμέτρους μπορούμε να χρησιμοποιήσουμε τον ίδιο αναλυτή για διαφορετικές συλλογές.

  - `host`: Η διεύθυση του εξυπηρετητή της φιλοξενούμενης συλλογής. Χρησιμοποιείται για τη διόρθωση των συνδέσμων κατά την παρουσιάση των αποτελεσμάτων.
  - `path`: Συμπληρωματική πληροφορία για την παράμετρο host, για το σωστό υπολογισμό της διεύθυνσης του εξυπηρετητή.
  - `parser`: Η κλάση Python του αναλυτή των αποτελεσμάτων αυτής της συλλογής.
  - `fetch_format`: Αναγνωριστικό/συντόμευση που υποδεικνύει σε ποιά μορφή θα γίνει η προσκόμιση των δεδομένων από τον εξυπηρετητή της συλλογής.
  - `num_results_regex_str`: Η κανονική έκφραση που χρησιμοποιείται για τον υπολογισμό του αριθμού των αποτελεσμάτων ενός ερωτήματος αναζήτησης στον εξυπηρετητή.
  - `nbrecs_regex_str`: Η κανονική έκφραση που χρησιμοποιείται για τον υπολογισμό του αριθμού των συνολικών εγγραφών της φιλοξενούμενης συλλογής.
  - `nbrecs_url`: Η διεύθυνση που περιέχει την πληροφορία για τον αριθμό των συνολικών εγγραφών της φιλοξενούμενης συλλογής. Χρησιμοποιείται σε συνδυασμό με την κανονική έκφραση που δηλώθηκε παραπάνω.

* `search_url`: Η διεύθυνση αναζήτησης στον εξυπηρετητή της φιλοξενούμενης συλλογής. Χρησιμοποιείται ως η βάση για την κατασκευή της διεύθυνσης του εκάστοτε ερωτήματος αναζήτησης προς τον εξυπηρετητή.

* `record_url`: Η βασική διεύθυνση στον εξυπηρετητή που χρησιμοποιείται για την κατασκευή της συγκεκριμένης διεύθυνσης κάθε εγγραφής.

* `selected_by default`: Boolean μεταβλητή που ορίζει αν στη βασική διεπαφή αναζήτησης η συλλογή αυτή θα είναι προεπιλεγμένη ή όχι.

Αφού ορίσουμε και τη φιλοξενούμενη συλλογή ως επεκταμένη εξωτερική συλλογή, πρέπει να υλοποιήσουμε τις απαραίτητες κλάσεις Python που έχουμε ορίσει ότι χρησιμοποιεί. Πρόκειται για τη μηχανή αναζήτησης (searcher), τον αναλυτή (parser) και τον προσκομιστή (getter).

Η μηχανή αναζήτησης

Η μηχανή αναζήτησης της κάθε συλλογής είναι ουσιαστικά μια κλάση Python, η οποία ορίζει ένα σύνολο συναρτήσεων, κάθε μια από τις οποίες έχει μια ξεχωριστή λειτουργία. Η βασική λειτουργικότητα της μηχανής αναζήτησης ορίζεται από την βασική κλάση `ExternalSearchEngine`. Από εκεί και πέρα, για κάθε συλλογή μπορεί να γραφτεί μια νέα κλάση, επέκταση της βασικής η οποία κληρονομεί όλες τις ιδιότητες της βασικής και μπορεί να ορίσεις νέες δικές τις παραμέτρους και συναρτήσεις ή να επεκτείνει τις υπάρχουσες επαναορίζοντάς τες. Βασική συνάρτηση της κάθε κλάσης είναι η συνάρτηση-δημιουργός (constructor), η οποία αρχικοποιεί τη μηχανή αναζήτησης και φροντίζει να τη διαμορφώνει σύμφωνα με τις ρυθμίσεις που ορίστηκαν στο ζεύγος κλειδιού-τιμής του προαναφερθέντος Python dictionary. Οι ρυθμίσεις αυτές περνάνε στην συνάρτηση-δημιουργό της κλάσης μέσω την παραμέτρου `configuration`, όπως φαίνεται στον παρακάτω κώδικα.

```
def __init__(self, configuration):
    self.search_url = ""
    self.combiner = " "
    self.name = None
    self.parser_params = None
    self.parser = None
    self.fetch_format = ""
    self.record_url = None
    self.selected_by_default = False
    for (name, value) in configuration.iteritems():
        setattr(self, name, value)
    if self.parser_params:
        setattr(self, 'parser', self.parser_params['parser'](self.
            parser_params))
        if 'fetch_format' in self.parser_params.keys():
            self.fetch_format = self.parser_params['fetch_format']
```

Οι υπόλοιπες βασικές συναρτήσεις της μηχανής αναζήτησης αναλαμβάνουν μεταξύ άλλων την κατασκευή της πλήρους διεύθυνσης αναζήτητης για τον εξυπηρετητή της φιλοξενούμενης συλλογής, η οποία αργότερα θα σταλεί στον προσκομιστή. Ακολουθούν δύο παραδείγματα τέτοιων συναρτήσεων, το πρώτο μιας συνάρτησης η οποία ελέγχει την ύπαρξη των διάφορων παραμέτρων που όρισε ο χρήστης στο τοπικό ερώτημα αναζήτησης και στη συνέχεια τις χρησιμοποιεί για την κατασκευή της διεύθυνσης αναζήτησης προς τον εξυπηρετητή, και το δεύτερο μιας συνάρτησης η οποία υπολογίζει τις διευθύνσεις συγκεκριμένων εγγραφών της φιλοξενούμενης συλλογής, δεδομένων των αναγνωριστικών τους.

```
def build_search_url(self,
                     basic_search_units,
                     req_args=None,
                     lang=CFG_SITE_LANG,
                     limit=CFG_EXTERNAL_COLLECTION_MAXRESULTS):
```

```
    if req_args:
        search_url_params = ""
        if type(req_args) is list:
            conjunction = " or "
            search_url_recids = conjunction.join(['recid:%s'] * len(req_args))
            params = tuple(req_args)
            search_url_recids %= params
            req_args = "p=" + search_url_recids + "&rg=" + str(100)
        req_args_dict = cgi.parse_qs(req_args)
        if req_args_dict.has_key('p'):
            search_url_params += urllib.quote(req_args_dict['p'][0])
        if req_args_dict.has_key('f'):
            search_url_params += '&f=' + req_args_dict['f'][0]
        if req_args_dict.has_key('jrec'):
            search_url_params += '&jrec=' + req_args_dict['jrec'][0]
        if req_args_dict.has_key('rg'):
            search_url_params += '&rg=' + req_args_dict['rg'][0]
        else:
            search_url_params += '&rg=' + str(limit)
        if req_args_dict.has_key('d1d'):
            search_url_params += '&d1d=' + req_args_dict['d1d'][0]
        if req_args_dict.has_key('d1m'):
            search_url_params += '&d1m=' + req_args_dict['d1m'][0]
        if req_args_dict.has_key('d1y'):
            search_url_params += '&d1y=' + req_args_dict['d1y'][0]
        if req_args_dict.has_key('d2d'):
            search_url_params += '&d2d=' + req_args_dict['d2d'][0]
        if req_args_dict.has_key('d2m'):
            search_url_params += '&d2m=' + req_args_dict['d2m'][0]
        if req_args_dict.has_key('d2y'):
            search_url_params += '&d2y=' + req_args_dict['d2y'][0]
        if req_args_dict.has_key('ap'):
            search_url_params += '&ap=' + req_args_dict['ap'][0]
        search_url_params += '&of=' + self.fetch_format
        return self.search_url + search_url_params
    else:
        units = self.build_units(basic_search_units)
        if len(units) == 0:
            return None
        request = self.combine_units(units)
        url_request = urllib.quote(request)
        return self.search_url + url_request + '&rg=' + str(limit) + '&of=' + self.fetch_format
```

```
def build_record_urls(self,
                      recids):
```

```

    if type(recids) is not list:
        recids = [recids]
    recids_urls = []
    for recid in recids:
        recids_urls.append((recid, self.record_url + recid))
    return recids_urls
```

Για να κατασκευαστεί μια όσο το δυνατόν πιο ακριβής διεύθυνση που θα φέρει τις μέγιστες δυνατές παραμέτρους του ερωτήματος αναζήτησης, περνάμε όλες αυτές παραμέτρτους στην αντίστοιχη συνάρτηση της μηχανής αναζήτησης και εκεί τις αποκωδικοποιούμε. Έτσι, όπως φαίνεται στη συνάρτηση του πρώτου παραδείγματος μπορούμε μέσω των συνεχών υποθέσων ¨αν¨ (if clause) να ελέγξουμε την ύπαρξη όσων παραμέτρων μας ενδιαφέρουν από αυτές που όρισε ο χρήστης στο ερώτημα αναζήτησης του τοπικού εξυπηρετητή και να τις μετατρέψουμε καταλλήλως για τη διεύθυνση του ερωτήματος αναζήτησης στον εξυπηρετητή της φιλοξενούμενης συλλογής.

Ο προσκομιστής

Ο προσκομιστής αναλαμβάνει να μεταφορτώνει/προσκομίζει τα περιεχόμενα μιας ή περισσοτέρων διαδικτυακών διευθύνσεων. Είναι κοινός για οποιαδήποτε συλλογή και η λειτουργία είναι πρακτικά ανεξάρτητη από το είδος της πληροφορίας που προσκομίζει. Στην παρούσα υλοποίηση ο προσκομιστής είναι ασύγχρονος, μεταφορτώνει δηλαδή τα περιεχόμενα κάθε διεύθυνσης από μια λίστα διευθύνσεων που δέχεται εν παραλλήλω. Η επικοινωνία με τον εκάστοτε εξυπηρετητή γίνεται με χρήση του πρωτοκόλλου Hypertext Transfer Protocol (HTTP). Η βασική συνάρτηση του προσκομιστή, η οποία και φαίνεται παρακάτω, δέχεται σαν παραμέτρους μια λίστα από αντικείμενα `pagegetter` που ουσιαστικά εμπεριέχουν τις διευθύνσεις προς προσκόμιση, μια προαιρετική συνάρτηση που μπορεί να εκτελεστεί για κάθε μια από τις διευθύνσεις αυτές ύστερα από επιτυχή προσκόμιση, μια λίστα προαιρετικών παραμέτρων για τη συνάρτηση αυτή και ένα όριο χρόνου.

```
def async_download(pagegetter_list,
                   finish_function=None,
                   datastructure_list=None,
                   timeout=15):

    time_start = time.time()
    finished_list = [False] * len(pagegetter_list)
    nb_remaining = 0
    check_redirected(pagegetter_list)
    for pagegetter in pagegetter_list:
        if pagegetter and not pagegetter.done:
            nb_remaining += 1
    while (time.time() - time_start < timeout) and nb_remaining > 0:
```

```
        if sys.hexversion < 0x2040000:
            asyncore.poll(0.01)
        else:
            asyncore.loop(0.01, True, None, 1)
        check_redirected(pagegetter_list)
        for i in range(len(pagegetter_list)):
            if pagegetter_list[i] and not finished_list[i] and pagegetter_list[i].done:
                nb_remaining -= 1
                if finish_function:
                    if datastructure_list:
                        datastructure = datastructure_list[i]
                    else:
                        datastructure = None
                    current_time = time.time() - time_start
                    finish_function(pagegetter_list[i], datastructure, current_time)
                finished_list[i] = True
    return finished_list
```

Κάθε αντικείμενο `pagegetter` είναι αντικείμενο της κλάσης `HTTPAsyncPageGetter` η οποία ουσιαστικά επεκτείνει την κλάση `dispatcher_with_send` της βιβλιοθήκης `asyncore`. Οι τελικές ιδιότητες και συναρτήσεις του αντικειμένου επιτρέπουν τη διαχείρισή του και τον έλεγχο της κατάστασής του. Η προαιρετική συνάρτηση `finish_function` είναι πολύ χρήσιμη όταν θέλουμε να επεξεργαστούμε ή να αποθηκεύσουμε συγκεντρωτικά τα αποτελέσματα του προσκομιστή. Τέλος, το όριο χρόνου προσκόμισης φροντίζει ώστε η συνάρτηση να ολοκληρώνεται σε περιπτώσεις όπου ο εξυπηρετητής, με τον οποίο προσπαθεί να επικοινωνήσει ο προσκομιστής, δεν ανταποκρίνεται.

Στην περίπτωση των φιλοξενούμενων συλλογών οι διευθύνσεις που κατασκευάζονται από τις συναρτήσεις της μηχανής αναζήτησης της κάθε συλλογής δίνονται στον προσκομιστή και εν συνεχεία, τα αποτελέσματα του προσκομιστή περνάνε στον αναλυτή.

Ο αναλυτής

Όπως και στην περίπτωση της μηχανής αναζήτησης, ο αναλυτής είναι μια κλάση Python, η οποία ορίζει ένα σύνολο συναρτήσεων, κάθε μια από τις οποίες έχει μια ξεχωριστή λειτουργία. Κάθε συλλογή μπορεί να ορίσει τη δικιά της κλάση, επεκτείνοντας της βασική κλάση `ExternalCollectionResultsParser`. Βασική λειτουργία του αναλυτή είναι να δέχεται τα δεδομένα από τον προσκομιστή, να τα αναλύει, και στη συνέχεια τα επιστρέφει σε άλλες συναρτήσεις για τελική επεξεργασία και παρουσίαση. Η ανάλυση συνήθως αφορά στην εφαρμογή κανονικών εκφράσεων για την εξαγωγή ή/και διαχωρισμό των αποτελεσμάτων, ή σε πιο απλές περιπτώσεις την εφαρμογή βασικών συναρτήσεων επεξεργασίας και διαχωρισμού λέξεων.

Στην περίπτωση του αναλυτή μιας φιλοξενούμενης συλλογής υλοποιούνται συναρτήσεις για την ανάλυση του συνολικού αριθμού εγγραφών της συλλογής, του αριθμού των αποτελεσμάτων αναζήτησης ενός συγκεκριμένου ερωτήματος και, φυσικά, των ίδιων των αποτελεσμάτων. Ο αναλυτής αναλαμβάνει και προσπαθεί να αναλύσει όλα τα δεδομένα που περιέχονται στην πληροφορία που του παρέχει ο προσκομιστής. Στη βέλτιση περίπτωση τα δεδομένα αυτά είναι δομημένα με τέτοιο τρόπο ώστε να είναι δυνατή η εξόρυξη όλων των μεταδεδομένων για κάθε εγγραφή. Από εκεί και πέρα ο αναλυτής μπορεί με τη βοήθεια άλλων συναρτήσεων, στη βέλτιση αυτή περίπτωση, να παράγει τα τελικά δεδομένα σχεδόν σε όλες τις μορφές που υποστηρίζει το CDS Invenio ώστε να τα παρουσιάσει στον τελικό χρήστη, μέσω της επιλεγμένης μορφής εξόδου.

Παρακάτω παρουσιάζονται μερικές από τις βασικές συναρτήσεις ενός αναλυτή. Η πρώτη συνάρτηση χρησιμοποιεί την κανονική έκφραση του αναλυτή για τον υπολογισμό του πλήθους των αποτελεσμάτων της αναζήτησης ενός ερωτήματος, η δεύτερη παρομοίως υπολογίζει το συνολικό πλήθος των εγγραφών μιας φιλοξενούμενης συλλογής, ενώ η τρίτη αναλαμβάνει να δέχετει τα δεδομένα από τον προσκομιστή και να τα αναλύει/διαχωρίζει.

```
def parse_num_results(self):

    if self.num_results_regex is None:
        return None
    list_matchs = self.num_results_regex.finditer(self.buffer)
    for match in list_matchs:
        return int(match.group(1).replace(',', ''))
    return None

def parse_nbrecs(self,
                 timeout):

    if self.nbrecs_regex is None:
        return None
    html = fetch_url_content([self.nbrecs_url], timeout)
    try:
        if len(html) == 1:
            matches = self.nbrecs_regex.search(html[0])
            return int(matches.group(1).replace(',', ''))
        else: return None
    except AttributeError:
        return -1
    except TypeError:
        return -2

def parse_and_get_results(self,
                          data,
                          of=None,
                          req=None,
                          limit=CFG_EXTERNAL_COLLECTION_MAXRESULTS,
```

```
                            feedonly=False,
                            parseonly=False):

    if not parseonly:
        self.clean()
        self.feed(data)
    if not feedonly:
        self.parse(of, req, limit)
        return self.results
```

Όπως περιγράφεται παραπάνω, η τρίτη συνάρτηση αναλαμβάνει να δεχθεί και να αναλύσει τα δεδομένα. Κάθε συλλογή ορίζει τη δικιά της συνάρτηση ανάλυσης και διαχωρισμού, συγκεκριμένη για τα δεδομένα που αναμένει από τον εξυπηρετητή. Στο παράδειγμα μια τέτοιας συνάρτησης που ακολουθεί, δεχόμαστε δεδομένα από μια φιλοξενούμενη συλλογή τύπου CDS Invenio , τα οποία έχουν τη δομή MARC21.

```
def parse(self,
          of='hb',
          req=None,
          limit=CFG_EXTERNAL_COLLECTION_MAXRESULTS):

    (recids, records) = self.parse_and_extract_records(of)
    if req and cgi.parse_qs(req.args).has_key('jrec'):
        counter = int(cgi.parse_qs(req.args)['jrec'][0]) - 1
    else:
        counter = 0
    for recid in recids:
        counter += 1
        if of == 'hb':
            html = """
            <tr><td valign="top" align="right" style="white-space:
                nowrap;">
            <input name="recid" type="checkbox" value="%(recid)s" />
            %(counter)s.
            </td><td valign="top">%(record)s</td></tr>""" \
            % {'recid': recid,
               'counter': counter,
               'record': records[recid]}
        elif of == 'xm':
            html = records[recid]
        else:
            html = None
        if html:
            self.add_html_result(html, limit)

def parse_and_extract_records(self,
                              of='hb'):

```

```
    record_pat = re.compile(r'(<record.*?>.*?</record>)', re.DOTALL + re.MULTILINE + re.IGNORECASE)
    recid_pat = re.compile(r'<controlfield tag="001">([0-9]+?)</controlfield>', re.DOTALL + re.MULTILINE + re.IGNORECASE)
    if not of:
        of='hb'
    try:
        results = record_pat.finditer(self.buffer)
        records = {}
        recids = []
        for result in results:
            xml_record = result.group(1)
            recid = recid_pat.search(xml_record).group(1)
            recids.append(recid)
            if of != 'xm':
                records[recid] = format_record(None, of, xml_record=xml_record)
            elif of == 'xm':
                records[recid] = xml_record
        return (recids, records)
    except AttributeError:
        return ([], {})
```

4.2.2 Μετατροπές στην υπηρεσία WebColl

Η υπηρεσία WebColl αναλαμβάνει να υπολογίσει το σύνολο των εγγραφών που ανήκουν σε κάθε τοπική συλλογή, τα αναγνωριστικά των εγγραφών δηλαδή καθώς και το πλήθος τους. Στην περίπτωση των φιλοξενούμενων συλλογών όπως εξηγήσαμε ήδη στην υποενότητα 4.1.2, σελίδα 47, έχει νόημα μόνο η αποθήκευση του συνολικού αριθμού των εγγραφών που τους ανήκουν.

Η υπηρεσία WebColl εκτελείται περιοδικά μέσω του προγραμμιστή του CDS Invenio, BibSched.Σε κάθε εκτέλεση της ελέγχεται πρώτα από όλα αν υπάρχει όντως η ανάγκη του υπολογισμού εκ νέου. Αν παρατηρηθεί ότι μια ή περισσότερες συλλογές έχουν μεταβληθεί, τότε η υπηρεσία προχωράει στην εκτέλεση του νέου υπολογισμού. Ο έλεγχος αυτό πραγματοποιείται από την παρακάτω υπόθεση ¨αν¨.

```
if check_nbrecs_for_all_external_collections() or \
    task_has_option("force") or \
    compare_timestamps_with_tolerance(\
        get_database_last_updated_timestamp(),
        get_cache_last_updated_timestamp(),
        cfg_cache_last_updated_timestamp_tolerance) >= 0:
```

Τα 3 σκέλη της υπόθεσης ¨αν¨, χωρισμένα με την πράξη ¨ή¨, σημαίνουν πρακτικά ότι ο υπολογισμός εκ νέου θα πραγματοποιηθεί όταν τουλάχιστον ένα εκ των παρακάτω ισχύει: (α) υπάρχει κάποια αλλαγή στο πλήθος των εγγραφών των φιλοξενούμενων συλλογών, (β) ο διαχειριστής επιβάλλει την εκτέλεση, (γ) παρατηρηθεί οποιαδήποτε μεταβολή που αφορά στις εγγραφές των τοπικών συλλογών στη βάση δεδομένων μετά από την τελευταία εκτέλεση της υπηρεσίας, με ανοχή μερικών δευτερολέπτων. Ας εστιάσουμε στην παρακάτω συνάρτηση που ελέγχει αν υπάρχει αλλαγή στο πλήθος των εγγραφών μιας φιλοξενούμενης συλλογής.

```
def check_nbrecs_for_all_external_collections():
    res = run_sql("SELECT name FROM collection WHERE dbquery LIKE 'hostedcollection:%';")
    for row in res:
        coll_name = row[0]
        if (get_collection(coll_name)).check_nbrecs_for_external_collection():
            return True
    return False
```

Επιλέγονται δηλαδή από τη βάση δεδομένων όλες οι φιλοξενούμενες συλλογές και για κάθε μια ελέγχουμε την υπόθεσή μας. Να σημειωθεί εδώ ότι κάθε συλλογή σε αυτό το σημείο είναι το αντικείμενο μιας γενικής κλάσης συλλογών. Η κλάση αυτή προσφέρει συναρτήσεις για τον υπολογισμό των μεγεθών που χρειαζόμαστε και αποθηκεύει τα αποτελέσματα στις ιδιότητές της. Ο έλεγχος πραγματοποιείται από τις δύο παρακάτω συναρτήσεις αυτής της κλάσης.

```
def check_nbrecs_for_external_collection(self):
    return self.nbrecs != self.calculate_nbrecs_for_external_collection(CFG_HOSTED_COLLECTION_TIMEOUT_NBRECS)

def calculate_nbrecs_for_external_collection(self, timeout=CFG_EXTERNAL_COLLECTION_TIMEOUT):
    if external_collections_dictionary.has_key(self.name):
        engine = external_collections_dictionary[self.name]
        if engine.parser:
            self.nbrecs_tmp = engine.parser.parse_nbrecs(timeout)
            if self.nbrecs_tmp >= 0:
                return self.nbrecs_tmp
            else:
                return self.nbrecs
    return 0
```

Παρατηρούμε ότι για κάθε φιλοξενούμενη συλλογή χρησιμοποιούμε τη μηχανή αναζήτησής της, και από τη μηχανή αναζήτησής της, τη συνάρτηση του αναλυτή της που υπολογίζει το συνολικό πλήθος των εγγραφών της. Η αρχικοποιημένες μηχανές αναζήτησης για κάθε συλλογή βρίσκονται σε ένα Python dictionary το οποίο εισάγεται στην αρχή του κώδικα της υπηρεσίας και αρχικοποιείται βασιζόμενο στο Python dictionary που περιγράφηκε στην υποενότητα 4.2.1, σελίδα 49.

Αν τελικά κριθεί ότι είναι απαραίτητο πραγματοποιείται η εκτέλεση του υπολογισμού εκ νέου, όπως φαίνεται στον παρακάτω κώδικα.

```python
for coll in colls:
    if str(coll.dbquery).startswith("hostedcollection:"):
        coll.set_nbrecs_for_external_collection()
    else:
        coll.calculate_reclist()
        coll.update_reclist()
```

Για τις φιλοξενούμενες συλλογές πραγματοποιείται απλά ο υπολογισμός εκ νέου του συνολικού αριθμού των εγγραφών με την παρακάτω συνάρτηση, ενώ για τις υπόλοιπες συλλογές πραγματοποιείται ο πλήρης υπολογισμός των εγγραφών που τους ανήκουν. Και στις δύο περιπτώσεις οι συλλογές είναι αντικείμενα της γενικής κλάσης συλλογών που αναφέρθηκε παραπάνω.

```python
def set_nbrecs_for_external_collection(self):
    if self.calculate_reclist_run_already:
        return
    if self.nbrecs_tmp:
        self.nbrecs = self.nbrecs_tmp
    else:
        self.nbrecs = self.calculate_nbrecs_for_external_collection(
            CFG_HOSTED_COLLECTION_TIMEOUT_NBRECS)
    self.calculate_reclist_run_already = 1
```

Όπως φαίνεται στη συνάρτηση, αν ο συνολικός αριθμός των εγγραφών έχει ήδη υπολογισθεί και αποθηκευθεί ως ιδιότητα του αντικειμένου από τους προηγούμενους ελέγχους, χρησιμοποιείται κατευθείαν χωρίς να υπολογισθεί εκ νέου.

4.2.3 Μετατροπές στα στάδια της αναζήτησης

Όπως αναφέρθηκε στην υποενότητα 4.1.1, σελίδα 41, η εκτέλεση της αναζήτησης ενός ερωτήματος στο CDS Invenio αποτελείται από έξι στάδια. Για την αναζήτηση σε φιλοξενούμενες συλλογές, παράλληλα με την τυπική αναζήτηση πραγματοποιήσαμε κάποιες μετατροπές σε αυτά τα στάδια, ιδίως πριν και μετά από αυτά, οι οποίες περιγράφονται παρακάτω.

Ένα από τα κύρια χαρακτηριστικά της σχεδίασης του μηχανής αναζήτησης του CDS Invenio είναι η ταχύτητα απόκρισης στα ερωτήματα των χρηστών. Στις τοπικές αναζητήσεις αυτό εξαρτάται από τοπικούς ελέγξιμους παράγοντες και κατά συνέπεια σε ένα σωστά ρυθμισμένο αποδοτικό σύστημα μπορεί να είναι εξαιρετική. Όταν όμως πρόκειται για φιλοξενούμενες συλλογές, τότε από της φύση της σχεδίασης τους η αναζήτηση πραγματοποιείται εμμέσως σε μακρινούς εξυπηρετητές και κατά συνέπεια η ταχύτητα απόκρισης εξαρτάται από εξωγενείς μη ρυθμίσιμους παράγοντες όπως την κατάσταση του δικτύου και του εκάστοτε εξυπηρετητή. Για την αντιμετώπιση αυτής της κατάστασης επιλέξαμε να πραγματοποιείται μια πρόωρη αναζήτηση στις φιλοξενούμενες συλλογές με ένα, επιλεγμένο από το διαχειριστή, αποδεκτά μικρό όριο χρόνου και σε περίπτωση που δεν είναι δυνατή η προσκόμιση αποτελεσμάτων σε αυτό το διάστημα, η αναζήτηση επαναλαμβάνεται στο τέλος της τοπικής αναζήτησης. Κατά αυτόν τον τρόπο ουσιαστικά η αναζήτηση στις φιλοξενούμενες συλλογές πραγματοποιείται ακριβώς πριν και ακριβώς μετά (αν είναι απαραίτητο) τα έξι στάδια της αναζήτησης και κατα τη διάρκεια αυτών ελέγχεται απλά η ύπαρξη τυχών αποτελεσμάτων και παρουσιάζονται στο χρήστη τα συνολικά αποτελέσματα.

Πριν ξεκινήσει οποιαδήποτε αναζήτηση διαχωρίζονται μεταξύ τους οι τοπικές και φιλοξενούμενες συλλογές οι οποίες θα χρησιμοποιηθούν στα στάδια της αναζήτησης. Η μέθοδος διαχωρισμού ακολουθεί διάφορα στάδια για να εξασφαλίσει ότι η αναζήτηση θα πραγματοποιηθεί μόνο στις συλλογές που έχει επιλέξει ο χρήστης. Πρακτικά υπάρχει πάντα μια συλλογή από την οποία ξεκινάει η αναζήτηση και η λίστα των συλλογών στις οποίες επιθυμεί ο χρήστης να πραγματοποιήσει την αναζήτηση. Για παράδειγμα αν ο χρήστης δεν επιλέξει ρητά κάποια συλλογή τότε η λίστα των συλλογών υπολογίζεται ως τα παιδιά-κλαδιά της συλλογής από την οποία ξεκίνησε η αναζήτηση. Η αρχική αυτή συλλογή ορίζεται με τη σειρά της από τη διεπαφή αναζήτησης του χρήστη. Η συνάρτηση διαχωρισμού αναλαμβάνει καταρχήν να εξασφαλίσει ότι η συλλογή από την οποία ξεκινά η αναζήτηση είναι πραγματική και να αφαιρέσει τις όποιες μη πραγματικές συλλογές. Στη συνέχεια υπολογίζει τις συλλογές παιδια-κλαδιά της αρχικής συλλογής (φιλοξενούμενες και μη) και αφαιρεί τις συλλογές που ενδέχεται να δώσουν ίδια αποτελέσματα. Τέλος, υπολογίζει τις φιλοξενούμενες συλλογές και επιστρέφει τις λίστες των συλλογών που θα χρησιμοποιηθούν στα στάδια της αναζήτησης. Ο κώδικας της συνάρτησης φαίνεται παρακάτω.

```
def wash_colls(cc,
               c,
               split_colls=0)

    colls_out = []
    colls_out_for_display = []
    hosted_colls_out = []

    if type(cc) is list:
        for ci in cc:
            if collection_reclist_cache.cache.has_key(ci):
                cc = ci
                break
    else:
        if not collection_reclist_cache.cache.has_key(cc):
            if cc:
                raise InvenioWebSearchUnknownCollectionError(cc)
            else:
                cc = CFG_SITE_NAME

    if type(c) is list:
        colls = c
    else:
        colls = [c]
```

```

    colls_real = []
    for coll in colls:
        if collection_reclist_cache.cache.has_key(coll):
            colls_real.append(coll)
        else:
            if coll:
                raise InvenioWebSearchUnknownCollectionError(coll)
    colls = colls_real

    if len(colls)==0:
        colls = [cc]

    res = run_sql("""SELECT c.name FROM collection AS c,
                                        collection_collection AS cc,
                                        collection AS ccc
                     WHERE c.id=cc.id_son AND cc.id_dad=ccc.id
                       AND ccc.name=%s AND cc.type='r'""", (cc,))

    l_cc_nonrestricted_sons_and_nonhosted_colls = []

    res_hosted = run_sql("""SELECT c.name FROM collection AS c,
                         collection_collection AS cc,
                         collection AS ccc
                         WHERE c.id=cc.id_son AND cc.id_dad=ccc.id
                         AND ccc.name=%s AND cc.type='r'
                         AND (c.dbquery NOT LIKE 'hostedcollection:%%'
                             OR c.dbquery IS NULL)""", (cc,))

    for row_hosted in res_hosted:
        l_cc_nonrestricted_sons_and_nonhosted_colls.append(row_hosted
            [0])
    l_cc_nonrestricted_sons_and_nonhosted_colls.sort()
    l_cc_nonrestricted_sons = []
    l_c = colls

    for row in res:
        if not collection_restricted_p(row[0]):
            l_cc_nonrestricted_sons.append(row[0])
    l_c.sort()
    l_cc_nonrestricted_sons.sort()
    if l_cc_nonrestricted_sons == l_c:
        colls_out_for_display = [cc]
    elif set(l_cc_nonrestricted_sons_and_nonhosted_colls).issubset(set(
        l_c)):
        colls_out_for_display = colls
        split_colls = 0
    else:
        colls_out_for_display = colls
```

```
    colls_out_for_display = list(set(colls_out_for_display))

    colls_to_be_removed = []
    for coll in colls_out_for_display:
        for ancestor in get_coll_ancestors(coll):
            if ancestor in colls_out_for_display and not is_hosted_collection(coll): colls_to_be_removed.append(coll)
    for coll in colls_to_be_removed:
        colls_out_for_display.remove(coll)

    if colls_out_for_display == [cc]:
        if is_hosted_collection(cc):
            hosted_colls_out.append(cc)
        else:
            for coll in get_coll_sons(cc):
                if is_hosted_collection(coll):
                    hosted_colls_out.append(coll)
    else:
        for coll in colls_out_for_display:
            if is_hosted_collection(coll):
                hosted_colls_out.append(coll)

    if split_colls == 0:
        colls_out = colls_out_for_display
    else:
        for coll in colls_out_for_display:
            coll_sons = get_coll_sons(coll)
            if coll_sons == []:
                colls_out.append(coll)
            else:
                for coll_son in coll_sons:
                    if not is_hosted_collection(coll_son):
                        colls_out.append(coll_son)

    colls_out = list(set(colls_out))

    if hosted_colls_out:
        for coll in hosted_colls_out:
            try:
                colls_out.remove(coll)
            except ValueError:
                pass

    return (cc, colls_out_for_display, colls_out, hosted_colls_out, debug)
```

Αφού πραγματοποιηθεί ο διαχωρισμός, και πριν ξεκινήσουν τα έξι στάδια της αναζήτησης που έχουν προαναφερθεί, πραγματοποιείται η πρόωρη αναζήτηση στις φιλοξενούμενες συλλογές. Η αναζήτηση αυτή επιστρέφει τα επιτυχή και ανεπιτυχή αποτελέσματα καθώς και τις συλλογές που δεν αποκρίθηκαν στο δοσμένο χρόνο. Στη συνέχεια υπολογίζονται τα πραγματικά υπαρκτά αποτελέσματα και τίθενται μερικές μεταβλητές που θα χρησιμοποιηθούν στα στάδια της αναζήτησης. Παρακάτω φαίνεται ο σχετικός κώδικας.

```
(hosted_colls_results , hosted_colls_timeouts ) =
    calculate_hosted_collections_results(req , [p, p1, p2, p3] , f ,
    hosted_colls , CFG_HOSTED_COLLECTION_TIMEOUT_ANTE_SEARCH)

if hosted_colls_results:
    hosted_colls_true_results = []
    for result in hosted_colls_results:
        if result[1] == None or result[1] == False:
            ...verbose message...
        else:
            hosted_colls_true_results.append(result)

hosted_colls_actual_or_potential_results_p = not (not hosted_colls or
    not ((hosted_colls_results and hosted_colls_true_results) or
    hosted_colls_timeouts))

hosted_colls_potential_results_p = not (not hosted_colls or not
    hosted_colls_timeouts)

only_hosted_colls_actual_or_potential_results_p = not colls_to_search
    and hosted_colls_actual_or_potential_results_p
```

Με τη μεταβλητή `CFG_HOSTED_COLLECTION_TIMEOUT_ANTE_SEARCH` ο διαχειριστής μπορεί να ελέγξει το όριο χρόνου για την πρόωρη αναζήτηση στις φιλοξενούμενες συλλογές. Η μέγιστη τιμή του ενός δευτερολέπτου φαίνεται λογική για αυτήν την περίπτωση. Παρακάτω θα αναλυθεί περαιτέρω η διαδικασία υπολογισμού των τυχών αποτελεσμάτων των φιλοξενούμενων συλλογών.

Εν συνεχεία, η αναζήτηση προχωρά στα έξι στάδια που περιλαμβάνει, λαμβάνοντας πλέον υπ'όψιν της τις διάφορες μεταβλητές που ορίστηκαν παραπάνω σχετικά με τα αποτελέσματα των φιλοξενούμενων συλλογών. Ειδικότερα, η αναζήτηση δεν ολοκληρώνεται σε περίπτωση μη εύρεσης τοπικών αποτελεσμέτων, και μεταξύ άλλων αναστέλλεται και η παρουσίαση προτεινόμενων αναζητήσεων. Επίσης, τα αποτελέσματα των φιλοξενούμενων συλλογών λαμβάνονται υπ'όψιν στην παρουσίαση των συνολικών αποτελεσμάτων και του χρόνου απόκρισης της αναζήτησης.

Μετά την ολοκλήρωση των έξι σταδίων της αναζήτησης, και σε περίπτωση που κάποια φιλοξενούμενη συλλογή είχε αποτύχει να ανταποκριθεί στην πρόωρη αναζήτηση λόγω του μικρού ορίου χρόνου, πραγματοποιείται μια δεύτερη αναζήτηση σε αυτές τις συλλογές, όπως φαίνεται στον παρακάτω κώδικα.

```python
if hosted_colls_timeouts:
    (hosted_colls_timeouts_results, hosted_colls_timeouts_timeouts) =
        do_calculate_hosted_collections_results(req, None,
        hosted_colls_timeouts, CFG_HOSTED_COLLECTION_TIMEOUT_POST_SEARCH
        )
```

Σε αυτήν την περίπτωση χρησιμοποιείται μια πιο άμεση συνάρτηση καθώς κάποιοι από τους υπολογισμούς είχαν ήδη πραγματοποιηθεί και για αυτές τις συλλογές στην πρόωρη αναζήτηση. Ας δούμε όμως πως ακριβώς γίνεται ο υπολογισμός των αποτελεσμάτων των φιλοξενούμενων συλλογών. Στην περίπτωση της πρόωρης αναζήτησης, πρώτα υπολογίζουμε τις παραμέτρους της αναζήτησης και στη συνέχεια εκτελούμε την αναζήτηση αυτή καθεαυτή. Η υπολογισμός των παραμέτρων της αναζήτησης συνίσταται στον υπολογισμό του ερωτήματος της αναζήτησης και στην ετοιμασία των μηχανών αναζήτησης των φιλοξενούμενων συλλογών. Οι μηχανές αναζήτησης, καθώς και οι αναλυτές τους, εισάγονται, όπως και στην περίπτωση της υπηρεσίας WebColl, από το αρχικοποιημένο Python dictionary που περιέχει τις αντίστοιχες κλάσεις για όλες τις φιλοξενούμενες συλλογές. Οι συναρτήσεις που αναλαμβάνουν τις διαδικασίες που μόλις περιγράφηκαν φαίνονται παρακάτω.

```python
def calculate_hosted_collections_results(req,
                                         pattern_list,
                                         field,
                                         hosted_collections,
                                         timeout=
                                             CFG_EXTERNAL_COLLECTION_TIMEOUT
                                             ):

    (hosted_search_engines, basic_search_units) =
        calculate_hosted_collections_search_params(
        req, pattern_list, field, hosted_collections)

    if not hosted_search_engines:
        return (None, None)

    return do_calculate_hosted_collections_results(req,
        basic_search_units, hosted_search_engines, timeout)

def calculate_hosted_collections_search_params(req,
                                               pattern_list,
                                               field,
                                               hosted_collections):

    pattern = bind_patterns(pattern_list)
    basic_search_units = create_basic_search_units(None, pattern, field)

    hosted_search_engines = select_hosted_search_engines(
        hosted_collections)

```

```
    return (hosted_search_engines, basic_search_units)

def select_hosted_search_engines(selected_hosted_collections):

    if not type(selected_hosted_collections) is list:
        selected_hosted_collections = [selected_hosted_collections]

    hosted_search_engines = set()

    for hosted_collection_name in selected_hosted_collections:
        if external_collections_dictionary.has_key(
            hosted_collection_name):
            engine = external_collections_dictionary[
                hosted_collection_name]
            if engine.parser:
                hosted_search_engines.add(engine)

    return hosted_search_engines
```

Μετά τον υπολογισμό των παραμέτρων, ή και κατευθείαν στην περίπτωση που πρόκειται για την αναζήτηση φιλοξενούμενων συλλογών, μετά την ολοκλήρωση των έξι σταδίων της τοπικής αναζήτησης, εκτελείται ουσιαστικά η αναζήτηση. Αρχικά υπολογίζεται για κάθε συλλογή η διεύθυνση αναζήτησης βασισμένη στο ερώτημα αναζήτησης που έχει θέσει ο χρήστης, και δημιουργείται μια λίστα από αντικείμενα `pagegetter`, ένα για κάθε μία από αυτές. Επίσης ορίζεται μια συνάρτηση `finish_function` για τον προσκομιστή η οποία φροντίζει να αποθηκεύει σε μια λίστα όλα τα αποτελέσματα των φιλοξενούμενων συλλογών. Στη συνέχεια καλείται ο ίδιος ο προσκομιστής, με το όριο χρόνου που έχει ορίσει ο διαχειριστής για αυτήν την περίπτωση, και το οποίο μπορεί να διαφέρει αν πρόκειται για πρόωρη ή μη αναζήτηση όπως εξηγήσαμε παραπάνω. Τέλος, αποθηκεύονται σε δυο διαφορετικές λίστες τα αποτελέσματα των συλλογών που αποκρίθηκαν στο δοσμένο όριο χρόνου και αυτών που δεν αποκρίθηκαν. Επίσης, υπολογίζεται το πλήθος των αποτελεσμάτων για κάθε συλλογή (σε περίπτωση επιτυχίας) με χρήση της αντίστοιχης συνάρτησης του αναλυτή. Ακολουθεί η συνάρτηση της αναζήτησης.

```
def do_calculate_hosted_collections_results(req,
                                            basic_search_units,
                                            hosted_search_engines,
                                            timeout=
                                                CFG_EXTERNAL_COLLECTION_TIMEOUT
                                                ):

    engines_list = []
    results_list = []
    full_results_list = []
    timeout_list = []

```

```
    if type(hosted_search_engines) is set:
        for engine in hosted_search_engines:
            url = engine.build_search_url(basic_search_units, req.args, lang)
            if url:
                engines_list.append([url, engine])
    elif type(hosted_search_engines) is list:
        for engine in hosted_search_engines:
            engines_list.append(engine)

    pagegetters_list = [HTTPAsyncPageGetter(engine[0]) for engine in engines_list]

    def finished(pagegetter, data, current_time):
        results_list.append((pagegetter, data, current_time))

    finished_list = async_download(pagegetters_list, finished, engines_list, timeout)

    for (finished, engine) in zip(finished_list, engines_list):
        if finished:
            for result in results_list:
                if result[1] == engine:
                    engine[1].parser.parse_and_get_results(result[0].data, feedonly=True)
                    full_results_list.append(
                        (engine, engine[1].parser.parse_num_results(), result[2])
                    )
                    break
        elif not finished:
                timeout_list.append(engine)

    return (full_results_list, timeout_list)
```

Για λόγους αποδοτικότητας, προτιμούμε να αναλύουμε τα επιτυχή αποτελέσματα των φιλοξενούμενων συλλογών (με χρήση των συναρτήσεων του αναλυτή, δηλαδή κανονικών εκφράσεων και συναρτήσεων διαχωρισμού λέξεων και φράσεων) κατά τη διάρκεια της παρουσίασης τους στο χρήστη και όχι κατά τη διάρκεια της προσκόμισής τους.

Στην πετίπτωση μιας ψηφιακής βιβλιοθήκης που δεν συντηρεί τοπικά δεδομένα, αλλά αποτελεί αμιγώς τον βασικό εξυπηρετητή ενός κατενεμημένου δικτύου συνεργαζόμενων φηφιακών βιβλιοθηκών, μπορούμε να θέσουμε την πρόωρη αναζήτηση ως τη βασική αναζήτηση δίνοντας στον ασύγχρονο προσκομιστή ένα ικανό αρχικό όριο χρόνου για την εύρεση αποτελεσμάτων (για παράδειγμα από ένα έως πέντε δευτερόλεπτα).

Κεφάλαιο 5

Υπηρεσίες Εξατομίκευσης

5.1 Μεθοδολογία

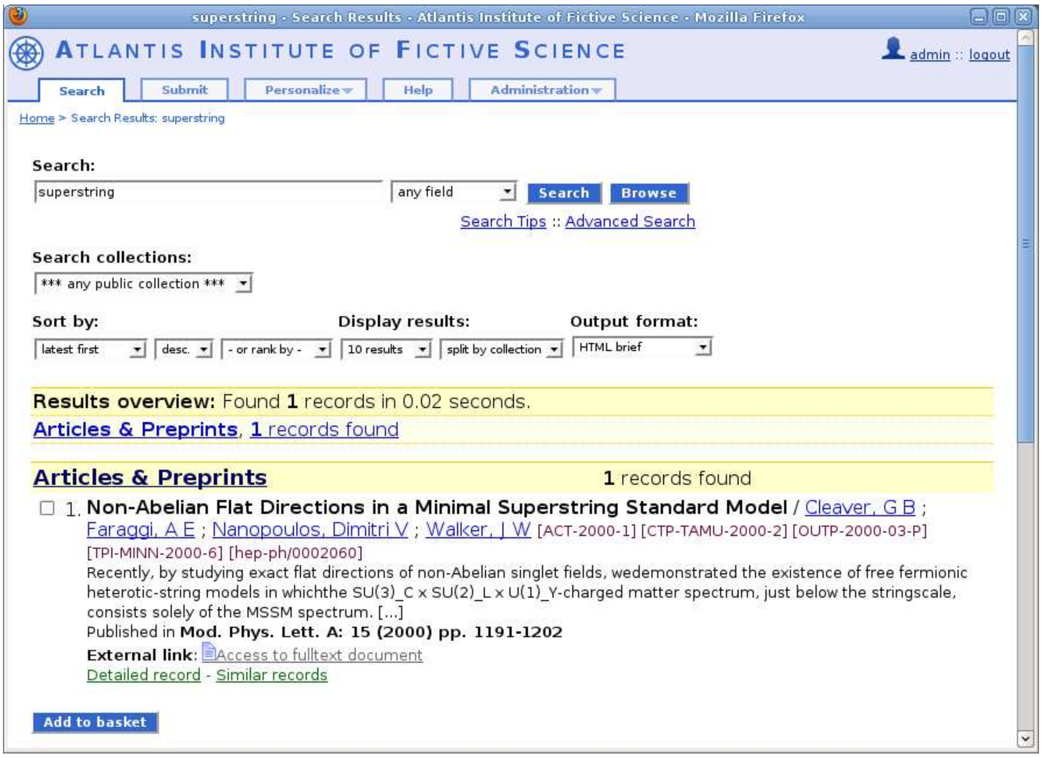
Σε αυτήν την ενότητα αναλύεται ο υπάρχων τρόπος λειτουργίας των μονάδων WebBasket και WebAlert (υποενότητα 5.1.1, σελίδα 69), που αποτελούν και τις κύριες υπηρεσίες εξατομίκευσης του CDS Invenio, και περιγράφεται η σχεδίαση των αλλαγών που προτάθηκαν (υποενότητα 5.1.2, σελίδα 78), ώστε να είναι δυνατή η προσαρμογή τους σε ένα κατανεμημένο δίκτυο συνεργαζόμενων ψηφιακών βιβλιοθηκών.

5.1.1 Ανάλυση

Περιγράφηκαν ήδη (ενότητα 3.4.5, σελίδα 35) οι βασικές αρχές λειτουργίας των αυτόνομων λογισμικών μονάδων υπηρεσιών εξατομίκευσης του CDS Invenio, WebBasket και WebAlert. Η μονάδα WebBasket επιτρέπει στο χρήστη να ορίσει τις δικές του προσωπικές συλλογές εγγράφων, βασισμένες στα ενδιαφέροντα του. Οι προσωπικές αυτές συλλογές μπορούν ύστερα, σύμφωνα με τις ρυθμίσεις του χρήστη, να γίνουν διαθέσιμες για προβολή, αλλά και για επεξεργασία και σε άλλους χρήστες, μέλη ομάδων. Η μονάδα WebAlert επιτρέπει στο χρήστη να ορίσει προσωπικές ειδοποιήσεις ηλεκτρονικού ταχυδρομείου (alerts) σχετικές με την καταχώρηση στην ψηφιακή βιβλιοθήκη νέων εγγράφων σχετικών με τα ενδιαφέροντα του. Οι δύο μονάδες μπορούν να συνεργασούν μεταξύ τους κατά τέτοιο τρόπο, ώστε τα αποτελέσματα μιας ειδοποίησης ενός χρήστη να αποθηκεύονται απ' ευθείας σε ένα από τα προσωπικά του καλάθια (baskets).

Καλάθια

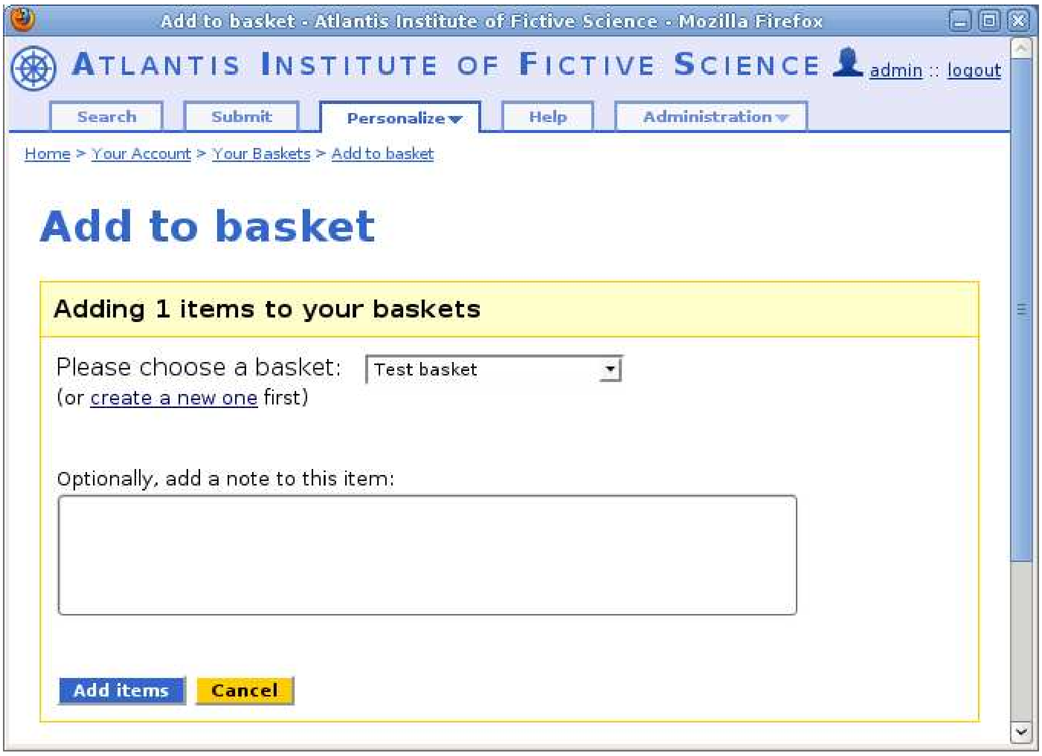
Ο χρήστης μπορεί να προσθέσει εγγραφές στα προσωπικά του καλάθια μέσω της διεπαφής αναζήτησης. Μπορεί να ορίσει προσωπικά καλάθια πριν αρχίσει να προσθέτει εγγραφές σε αυτά ή να ορίσει το πρώτο του καλάθι κατά τη διάρκεια της πρώτης προσθήκης εγγραφών. Η διαδικασία προσθήκης είναι άμεσα συνδεδεμένη με τη διεπαφή αναζήτησης της ψηφιακής βιβλιοθήκης. Ο χρήστης καλείται δηλαδή να διαλέξει ποιές εγγραφές από τις συνολικές, που αποτελούν το αποτέλεσμα κάποιας αναζήτησης, θα προσθέσει σε κάποια από τα καλάθια του. Στη σχήμα 5.1, σελίδα 70, παρουσιάζεται το αποτέλεσμα της αναζήτησης του ερωτήματος `superstring` μέσω της διεπαφής του χρήστη και φαίνονται το κουτί επιλογής της εγγραφής και το κουμπί υποβολής προσθήκης σε κάποιο καλάθι (Add to basket).



Σχήμα 5.1: Επιλογή προσθήκης εγγραφών σε κάποιο καλάθι του χρήστη μέσω της διεπαφής αναζήτησης

Ο χρήστης, στη συνέχεια, επιλέγει το καλάθι στο οποίο επιθυμεί να προσθέσει την ή τις επιλεγμένες εγγραφές, όπως φαίνεται στο σχήμα 5.2, σελίδα 71. Σε αυτό το σημείο δίνεται στο χρήστη και η δυνατότητα προσθήκης ενός σχολίου σχετικό με την ή τις εγγραφές, που αποτελεί ακόμη ένα χαρακτηριστικό της υπηρεσίας των καλαθιών.

Στην περίπτωση που ο χρήστης δεν έχει ήδη ορίσει κάποιο καλάθι καλείται να ορίσει ένα καινούργιο. Μπορεί επίσης να επαναλάβει τη διαδικασία ορισμού νέων καλαθιών για να δημιουργήσει μια δομή ομαδοποίησης των εγγραφών που τον ενδιαφέρουν. Κατά τη διάρκεια ορισμού ενός καλαθιού, ο χρήστης ορίζει και το θέμα του, εισάγοντας έτσι ένα ακόμη επίπεδο ομαδοποίησης στην υπηρεσία αυτή. Δεν υπάρχει κάποιο όριο στον αριθμό των θεμάτων που μπορεί να ορίσει ο χρήστης, είτε στο αριθμό των καλαθιών που μπορεί να ανήκουν σε κάποιο θέμα, είτε στον αριθμό



Σχήμα 5.2: Επιλογή προσθήκης εγγραφών σε κάποιο καλάθι του χρήστη μέσω της διεπαφής καλαθιών

των εγγραφών που μπορεί να ανήκουν σε κάποιο καλάθι. Σε ένα παράδειγμα ομαδοποίησης, ο χρήστης έχει ορίσει δύο θέματα: *Φυσική* και *Πληροφορική*. Στο θέμα Φυσική ανήκουν δύο καλάθια, *Θεωρητική Φυσική* και *Πρακτική Φυσική*. Στο θέμα Πληροφορική ανήκουν τρία καλάθια, *Software*, *Hardware* και *Middleware*. Σε αυτά τα καλάθια ο χρήστης έχει προσθέσει σχετικές εγγραφές από τα αποτελέσματα των αναζητήσεών του. Στο σχήμα 5.3, σελίδα 72, παρουσιάζεται μια οργανωμένη προβολή των θεμάτων και καλαθιών του χρήστη σύμφωνα με το παραπάνω παράδειγμα, ενώ στο σχήμα 5.4, σελίδα 73, φαίνεται ένα από αυτά τα καλάθια με τα περιεχόμενά του.

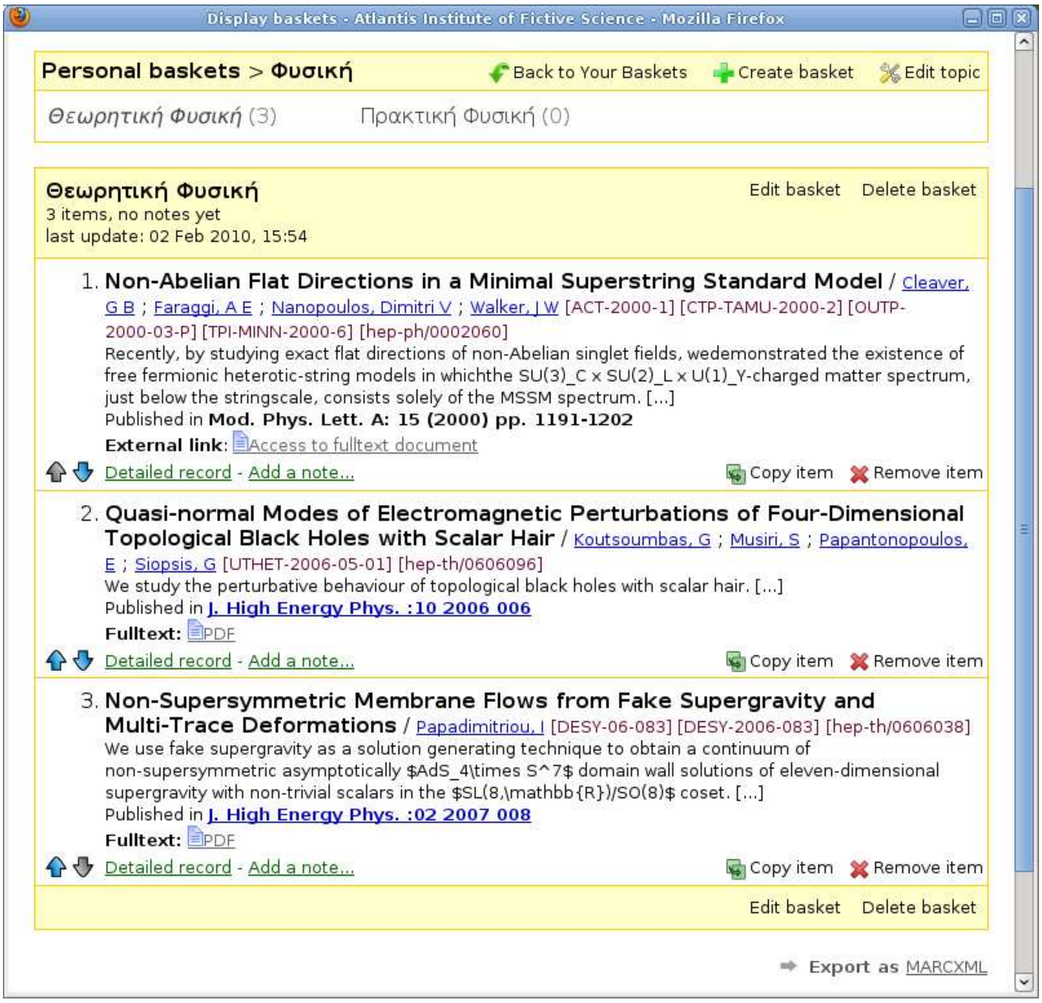
Δίνεται στο χρήστη η δυνατότητα επεξεργασίας των θεμάτων, των καλαθιών και των ίδιων των εγγραφών. Η επεξεργασία των καλαθιών, πέραν της μετονομασίας, της μετακίνησης σε κάποιο άλλο θέμα και της διαγραφής, επιτρέπει στο χρήστη να μοιραστεί το καλάθι με τα μέλη κάποιας ομάδας στην οποία ανήκει ή και με όλους τους επισκέπτες της ψηφιακής βιβλιοθήκης, όπως θα δούμε παρακάτω. Κάθε εγγραφή μπορεί να μετακινηθεί μέσα στο ίδιο το καλάθι, αναδεικνύοντας την προτεραιότητά της, να αντιγραφεί σε κάποιο άλλο καλάθι ή να αφαιρεθεί από το παρόν.

Κάθε εγγραφή που ανήκει σε κάποιο καλάθι αποθηκεύεται στη βάση δεδομένων στο συγκεκριμένο καλάθι, χρησιμοποιώντας το αναγνωριστικό της, το οποίο είναι

Σχήμα 5.3: Οργανωμένη προβολή των θεμάτων και καλαθιών του χρήστη

μοναδικό (θετικός ακέραιος) για όλες τις τοπικές συλλογές. Για την προβολή των εγγραφών στη διεπαφή προβολής καλαθιών χρησιμοποιούνται για λόγους ταχύτητας οι προ-αποθηκευμένες διαμορφωμένες εγγραφές (ιστοσελίδες) ή διαμορφώνονται επί τόπου και αποθηκεύονται για μελλοντική χρήση.

Όπως αναφέρθηκε προηγουμένως, καθώς και στην ενότητα 3.4.5, σελίδα 35, τα καλάθια μπορούν να μοιράζονται μεταξύ των μελών μιας ομάδας, καθώς και να είναι διαθέσιμα για προβολή και σχολιασμό σε όλα τα μέλη της ψηφιακής βιβλιοθήκης. Τις ρυθμίσεις αυτές μπορεί να κάνει αρχικά μόνο ο χρήστης που δημιούργησε το καλάθι. Για να θέσει ο χρήστης το καλάθι ως μοιραζόμενο στα μέλη μιας ομάδας, θα πρέπει πρώτα να είναι ο ίδιος μέλος της ομάδας αυτής. Η περιγραφή του ορισμού των ομάδων μελών και της συμμετοχής σε αυτές είναι πέραν του σκοπού της παρούσας εργασίας, οπότε θα υποθέσουμε ότι ο χρήστης είναι ήδη μέλος μερικών τέτοιων ομάδων. Για κάθε ομάδα ο χρήστης μπορεί να ορίσει ένα διαφορετικό επίπεδο δικαιωμάτων για ένα καλάθι, που ξεκινά από το απλό δικαίωμα προβολής και καταλήγει στα πλήρη δικαιώματα επεξεργασίας του καλαθιού και των περιεχομένων του. Ενδιάμεσα επίπεδα περιλαμβάνουν δικαιώματα προβολής και προσθήκης σχολιασμού, αφαίρεσης εγγραφών, κ.α. Ο χρήστης μπορεί επίσης να επιλέξει να μοιράζεται το καλάθι με όλα τα μέλη της ψηφιακής βιβλιοθήκης, μετατρέποντάς το έτσι σε *δημόσιο καλάθι*. Το μέγιστο δικαίωμα που μπορεί να οριστεί για ένα δημόσιο καλάθι είναι αυτό της προσθήκης σχολίων. Κάθε χρήστης μπορεί να επιλέξει να γίνει συνδρομητής (συβσςριβερ) σε κάποιο καλάθι ώστε να μπορεί να πλοηγηθεί σε αυτό άμεσα μέσω της διεπαφής προβολής καλαθιών. Ένα καλάθι παραμένει πάντα προσωπικό, ως το καλάθι ενός χρήστη, ενώ μπορεί συγχρόνως να μοιράζεται σε δια-

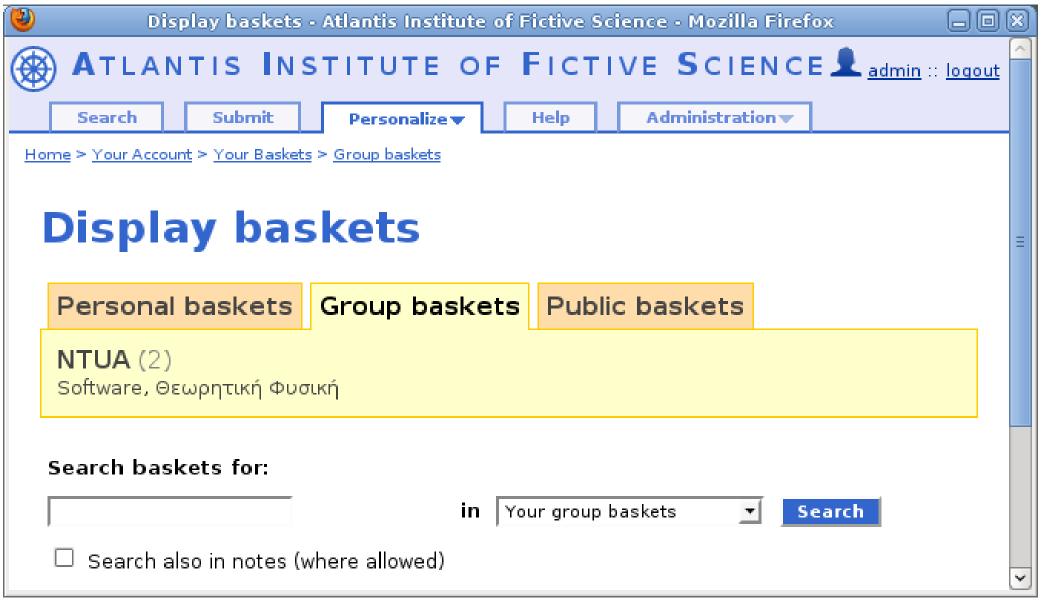


Σχήμα 5.4: Προβολή των περιεχομένων ενός καλαθιού

φορετικές ομάδες με διαφορετικά δικαιώματα στην κάθε μια, καθώς και σε όλα τα μέλη της κοινότητας της ψηφιακής βιβλιοθήκης. Ο χρήστης μπορεί να πλοηγηθεί σε όλα τα καλάθια που του ανήκουν, στα οποία έχει δικαιώματα ως μέλος κάποιας ομάδας, ή στα δημόσια καλάθια, στα οποία είναι συνδρομητής, μέσω της σχετικής διεπαφής που φαίνεται στα σχήματα 5.5, σελίδα 74, και 5.6, σελίδα 74.

Η διεπαφή προβολής όλων των δημόσιων καλαθιών είναι επίσης διαθέσιμη για όλη την κοινότητα χρηστών, ώστε να μπορούν να γίνουν συνδρομητές σε αυτά αν το επιθυμούν, όπως φαίνεται στο σχήμα 5.7, σελίδα 75.

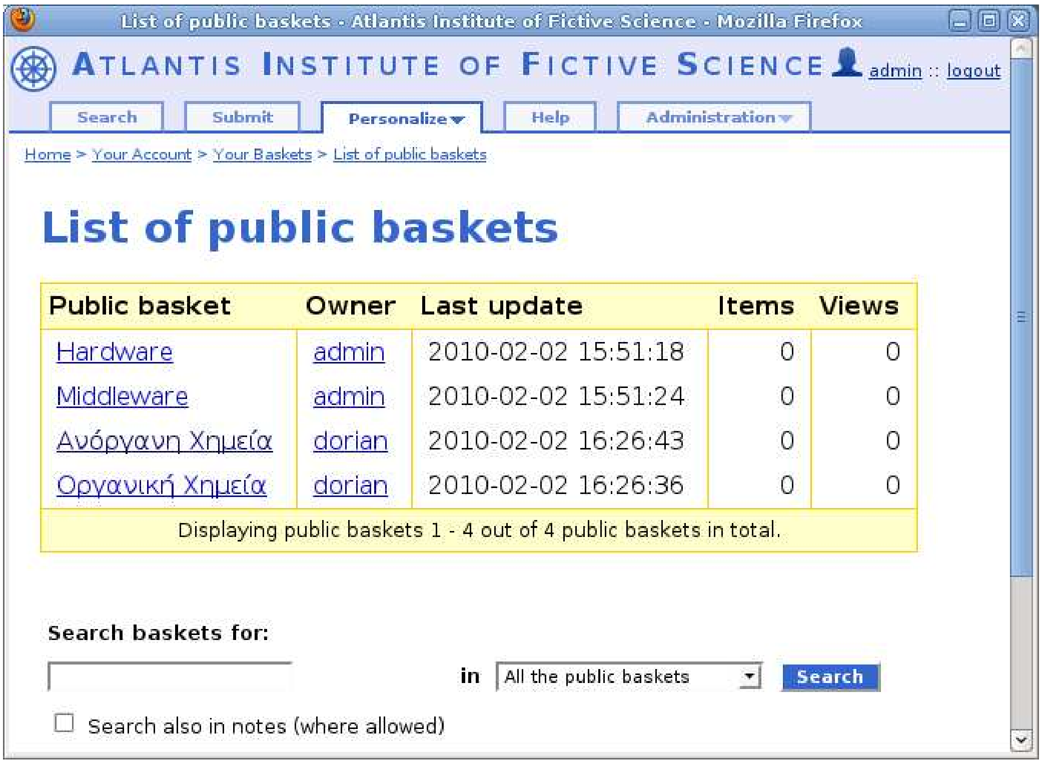
Ένα ακόμη χαρακτηριστικό που προσφέρει η υπηρεσία των καλαθιών είναι η δυνατότητα αναζήτησης στα περιεχόμενα τους, καθώς και η αναζήτηση στα σχόλια που έχουν γίνει στις εγγραφές τους. Για την αναζήτηση ενός ερωτήματος χρησιμοποιούνται οι συναρτήσεις αναζήτησης της μονάδας WebSearch και στη συνέχεια



Σχήμα 5.5: Προβολή των μοιραζόμενων σε ομάδες καλαθιών ενός χρήστη, ανά ομάδα

Σχήμα 5.6: Προβολή των δημόσιων καλαθιών στα οποία είναι συνδρομητής ο χρήστης

γίνεται τομή με τα συνολικά περιεχόμενα του κάθε καλαθιού για την εύρεση κοινών εγγραφών. Για την αναζήτηση στα σχόλια χρησιμοποιείται η απ' ευθείας αναζήτηση στα αποθηκευμένα σχόλια στη βάση δεδομένων.



Σχήμα 5.7: Προβολή όλων των δημόσιων καλαθιών

Ειδοποιήσεις

Ο χρήστης μπορεί να ορίσει νέες ειδοποιήσεις με δύο τρόπους: είτε απευθείας μέσω της διεπαφής αναζήτησης κατά την παρουσίαση των αποτελεσμάτων ενός ερωτήματος, είτε μέσω του ιστορικού των αναζητήσεων του χρήστη. Στην πρώτη περίπτωση, ο χρήστης εκτελεί μια οποιαδήποτε αναζήτηση, και ακολούθως του δίνεται η δυνατότητα να ορίσει μια ειδοποίηση σχετική με το ερώτημα της αναζήτησης αυτής, όπως φαίνεται στο σχήμα 5.8, σελίδα 76. Στη δεύτερη περίπτωση, ο χρήστης μπορεί να επιλέξει να ορίσει μια νέα ειδοποίηση βασισμένη σε κάποια αναζήτηση που πραγματοποίησε στο παρελθόν, μέσω της διεπαφής ιστορικού αναζητήσεων, όπως φαίνεται στο σχήμα 5.9, σελίδα 77.

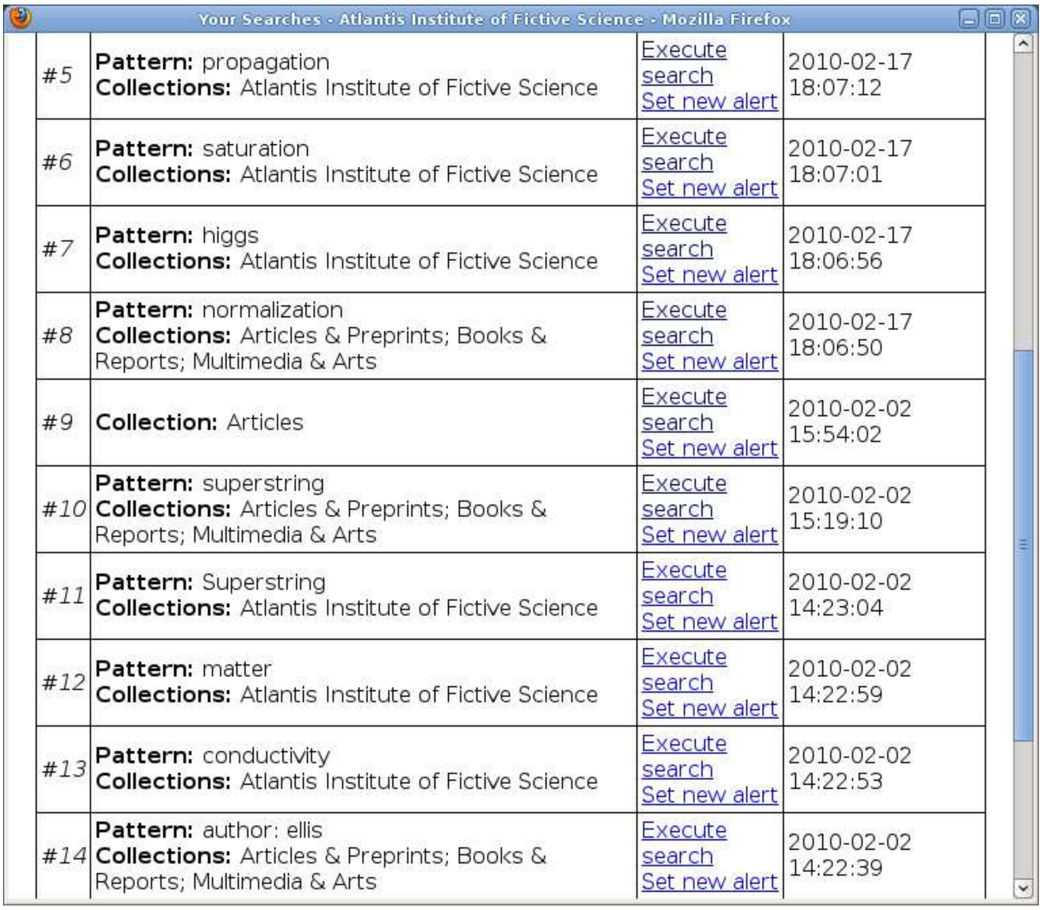
Και στις δύο παραπάνω περιπτώσεις, ο χρήστης οδηγείται στη διεπαφή ορισμού και αποθήκευσης της ειδοποίησης, η οποία φαίνεται στο σχήμα 5.10, σελίδα 78. Στη διεπαφή αυτή ο χρήστης καλείται, αφού δώσει ένα όνομα στη νέα ειδοποίηση, να ορίσει τις παραμέτρους της. Η πρώτη παράμετρος αφορά στη συχνότητα της ειδοποίησης, το κάθε πότε δηλαδή θα εκτελείται αναζητώντας νέα αποτελέσματα (πιθανές συχνότητες είναι η ημερησία, ή εβδομαδιαία και η μηνιαία). Η δεύτερη παράμετρος αφορά στο αν θα στέλνεται ένα αυτόματο μήνυμα ηλεκτρονικού ταχυδρομείου στο χρήστη ενημερώνοντάς τον για τα αποτελέσματα της ειδοποίησης, ενώ η τρίτη πα-

Σχήμα 5.8: Ορισμός ειδοποίησης μέσω της διεπαφής αναζήτησης

ράμετρος αφορά στο αν η ειδοποίηση θα αποθηκεύει αυτομάτως τα αποτελέσματά της σε κάποιο καλάθι του χρήστη, καθώς και ποιο καλάθι θα είναι αυτό. Τουλάχιστον μια εκ των δύο τελευταίων παραμέτρων πρέπει να είναι επιλεγμένη.

Ο χρήστης μπορεί να διαχειριστεί όλες τις ειδοποιήσεις που έχει ορίσει μέσω της διεπαφής που φαίνεται στο σχήμα 5.11, σελίδα 79.

Από την πλευρά του εξυπηρετητή, ο μηχανισμός των ειδοποιήσεων πρέπει από τη φύση του να εκτελείται σε καθημερινή βάση. Σε έναν τυπικό εξυπηρετητή Unix-like αυτή η διαδικασία μπορεί να απλοποιηθεί θέτοντας μια νέα εργασία στο χρονοπρογραμματιστή Cron. Ο μηχανισμός αναλαμβάνει καταρχήν να επεξεργαστεί τις ειδοποιήσεις ξεχωριστά, αναλόγως με τη συχνότητά τους. Για κάθε μία ειδοποίηση, χρησιμοποιούνται συναρτήσεις της μονάδας WebSearch για την αναζήτηση αποτελεσμάτων στο χρονικό πλαίσιο που ορίζει η συχνότητα της αναζήτησης (για παράδειγμα στην εβδομαδιαία συχνότητα, για τις τελευταίες επτά ημέρες, κ.ο.κ.). Η αναζήτηση πραγματοποιείται όπως αναλύεται στην υποενότητα 4.1.1, σελίδα 41, και χρησιμοποιούνται οι αρχικές παράμετροι του ερωτήματος αναζήτησης του χρήστη, για το οποίο ορίστηκε η ειδοποίηση. Αν υπάρχουν αποτελέσματα ειδοποιείται ο χρήστης μέσω ηλεκτρονικού ταχυδρομείου ή/και αποθηκεύονται οι εγγραφές που βρέθηκαν στο καλάθι που έχει οριστεί στις παραμέτρους της ειδοποίησης. Το μήνυμα ηλε-



#5	**Pattern:** propagation **Collections:** Atlantis Institute of Fictive Science	Execute search Set new alert	2010-02-17 18:07:12
#6	**Pattern:** saturation **Collections:** Atlantis Institute of Fictive Science	Execute search Set new alert	2010-02-17 18:07:01
#7	**Pattern:** higgs **Collections:** Atlantis Institute of Fictive Science	Execute search Set new alert	2010-02-17 18:06:56
#8	**Pattern:** normalization **Collections:** Articles & Preprints; Books & Reports; Multimedia & Arts	Execute search Set new alert	2010-02-17 18:06:50
#9	**Collection:** Articles	Execute search Set new alert	2010-02-02 15:54:02
#10	**Pattern:** superstring **Collections:** Articles & Preprints; Books & Reports; Multimedia & Arts	Execute search Set new alert	2010-02-02 15:19:10
#11	**Pattern:** Superstring **Collections:** Atlantis Institute of Fictive Science	Execute search Set new alert	2010-02-02 14:23:04
#12	**Pattern:** matter **Collections:** Atlantis Institute of Fictive Science	Execute search Set new alert	2010-02-02 14:22:59
#13	**Pattern:** conductivity **Collections:** Atlantis Institute of Fictive Science	Execute search Set new alert	2010-02-02 14:22:53
#14	**Pattern:** author: ellis **Collections:** Articles & Preprints; Books & Reports; Multimedia & Arts	Execute search Set new alert	2010-02-02 14:22:39

Σχήμα 5.9: Ορισμός ειδοποίησης μέσω του ιστορικού αναζητήσεων

κτρονικού ταχυδρομείου περιλαμβάνει μια λίστα με σύντομες πληροφορίες για τις εγγαφρές που βρέθηκαν, χωρισμένες ανά συλλογή, καθώς επίσης και την ηλεκτρονική διεύθυνση που θα αναπαράγει τα αποτελέσματα αυτά στον εξυπηρετητή. Για την προσθήκη εγγραφών στο καλάθι χρησιμοποιείται η συνάρτηση προσθήκης στη βάση δεδομένων της μονάδας WebBasket. Σημειώνεται εδώ ότι ο διαχειριστής μπορεί να ορίσει ένα μέγιστο όριο εγγραφών για τις οποίες μπορεί να ενημερωθεί ο χρήστης για κάθε ειδοποίηση. Το ίδιο όριο ισχύει και για το μέγιστο αριθμό εγγραφών που μπορούν να προστεθούν αυτοματοποιημένα στο καλάθι του χρήστη μέσω κάποιας ειδοποίησης. Με αυτό τον τρόπο αποφεύγεται σε ένα βαθμό ο υπερπληθυσμός καλαθιών σε περιπτώσεις ¨ξεχασμένων¨ ειδοποιήσεων, αλλά και η παραλαβή ιδιαίτερα μεγάλων μηνυμάτων ηλεκτρονικού ταχυδρομείου από τους χρήστες, σε περίπτωση που μια ειδοποίηση αποφέρει πολλαπλά αποτελέσματα.

Σχήμα 5.10: Ορισμός και αποθήκευση μιας ειδοποίησης

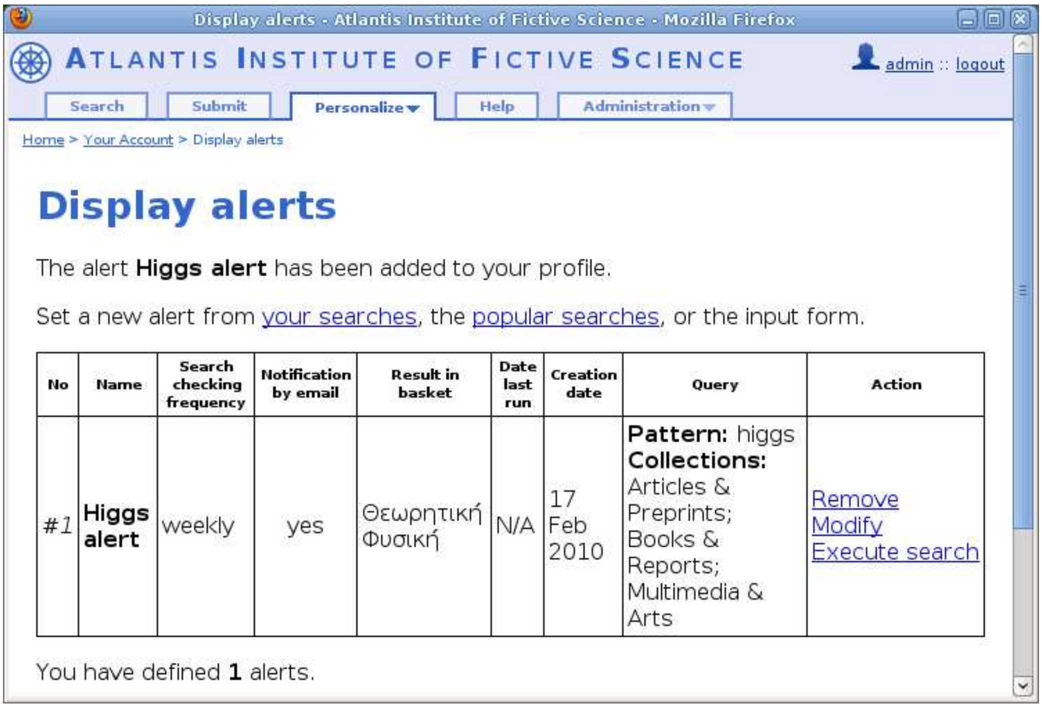
5.1.2 Σχεδίαση

Στην παρούσα υποενότητα μελετάται η προταθείσα σχεδίαση της υλοποίησης που επιτρέπει στις μονάδες WebBasket και WebAlert να ανταποκρίνονται στις απαιτήσεις των υπηρεσιών εξατομίκευσης σε ένα κατανεμημένο δίκτυο συνεργαζόμενων ψηφιακών βιλιοθηκών.

Καλάθια

Αναφέρθηκε ήδη ότι σκοπός της υπηρεσίας καλαθιών είναι η προσθήκη και η διαχείριση εγγραφών που προέχονται από οποιαδήποτε φιλοξενούμενη συλλογή, όπως αυτές ορίστηκαν στην υποενότητα 4.1.2, σελίδα 47. Οι εγγραφές αυτές θα πρέπει να μπορούν να προστεθούν στα καλάθια, όπως και οι τοπικές εγγραφές, και η διαχείρισή τους να είναι παρόμοια.

Μια βασική διαφορά μεταξύ των τοπικών εγγραφών και αυτών που προέρχονται από τις φιλοξενούμενες συλλογές, είναι ότι οι πρώτες χαρακτηρίζονται από το μοναδικό τους αναγνωριστικό, ένα θετικό ακέραιο. Αυτό υπαγορεύεται από τη σχεδίαση της βάσης δεδομένων και τον τρόπο τρόπο αποθήκευσής τους. Οι δεύτερες εγγραφές, όταν "φεύγουν" πια από τον τοπικό τους εξυπηρετητή και "περνάνε" στον



Σχήμα 5.11: Διεπαφή διαχείρισης των ειδοποιήσεων ενός χρήστη

εξυπηρετητή της ψηφιακής βιβλιοθήκης του κατανεμημένου δικτύου, χάνουν αυτή την ιδιότητα. Ανήκουν πλέον σε μια φιλοξενούμενη συλλογή, η οποία χαρακτηρίζεται όμως με ένα μοναδικό αναγνωριστικό. Ένας τρόπος λοιπόν να διατηρήσει κάθε εγγραφή τη μοναδικότητά της είναι να χαρακτηριστεί με το μοναδικό αναγνωριστικό της, από τη φιλοξενούμενη συλλογή από την οποία προήρθε, καθώς και το μοναδικό αναγνωριστικό της φιλοξενούμενης συλλογής στον εξυπηρετητή του κατανεμημένου δικτύου. Αυτά τα δύο δεδομένα χρησιμοποιούνται για την προσθήκη κάθε τέτοιας εγγραφής μέσω της διεπαφής αναζήτησης, και τελικά η προσθήκη πραγματοποιείται όπως ακριβώς και για οποιαδήποτε τοπική εγγραφή.

Από τη στιγμή που μια εγγραφή φιλοξενούμενης συλλογής προστεθεί σε κάποιο καλάθι, αποθηκεύεται και στη βάση δεδομένων. Όπως αναφέρθηκε στην προηγούμενη υποενότητα, για τις τοπικές εγγραφές χρησιμοποιείται το μοναδικό αναγνωριστικό της κάθε μιας, το οποίο είναι ένας θετικός ακέραιος. Για τις εγγραφές των φιλοξενούμενων συλλογών μπορούμε να χρησιμοποιήσουμε έναν μοναδικό αρνητικό ακέραιο, ώστε να μπορούμε να αναφερθούμε σε αυτές χρησιμοποιώντας ένα μοναδικό αναγνωριστικό, αφού τις αποθηκεύσουμε στη βάση δεδομένων, και να τις χρησιμοποιήσουμε εφεξής σε όλο το εύρος της υπηρεσίας καλαθιών. Η πλήρης χρήση και διαχείριση γίνεται δυνατή με την εισαγωγή δύο νέων πινάκων στη βάση δεδομένων, ένα για την αποθήκευση γενικών πληροφοριών για κάθε εγγραφή, όπως κάποια από τα χαρακτηριστικά της στον εξυπηρετητή από τον οποίο προήρθε, και ένα για την αποθήκευση των συμπιεσμένων προ-υπολογισμένων διαμορφώσεων κάθε εγγραφής, το οποίο εξυπηρετεί τη γρήγορη προβολή τους κάθε φορά που ο χρήστης χρησιμοποιεί τη διεπαφή προβολής των καλαθιών.

Όσον αφορά την αναζήτηση των περιεχομένων των καλαθιών, αδυνατούμε να χρησιμοποιήσουμε τις κλασικές συναρτήσεις αναζήτησης της μονάδας WebSearch για τις εγγραφές φιλοξενούμενων συλλογών, εφόσον οι συναρτήσεις αυτές χρησιμοποιούν αφενώς μεθόδους ευρετηρίασης και αφετέρου τη βάση δεδομένων. Όπως είδαμε στην υποενότητα 4.1.2, σελίδα 47, η αναζήτηση εγγραφών φιλοξενούμενων συλλογών γίνεται με διαφορετικό τρόπο. Αποφεύγουμε επίσης να χρησιμοποιήσουμε παρόμοιες τεχνικές με αυτές που περιγράφηκαν στην προαναφερθείσα υποενότητα για την αναζήτηση εγγραφών φιλοξενούμενων συλλογών, εφόσον θα ήταν απαραίτητη κάθε φορά η χρήση του προσκομιστή, και αυτό μπορεί να εισάγει εξωγενείς καθυστερήσεις. Αντ' αυτών, εκμεταλλευόμαστε το γεγονός ότι αποθηκεύουμε στη βάση δεδομένων διαμορφωμένες όλες τις εγγραφές φιλοξενούμενων συλλογών που έχουν προσθεθεί σε κάποιο καλάθι, και η αναζήτηση πραγματοποιείται απ' ευθείας στη βάση δεδομένων, όπως και για τα σχόλια των εγγραφών αυτών.

Στην υποενότητα 5.2.1, σελίδα 81, θα δούμε αναλυτικότερα πώς η υλοποιείται παραπάνω σχεδίαση.

Ειδοποιήσεις

Σκοπός της υπηρεσίας των ειδοποιήσεων είναι η ενημέρωση του χρήστη σε περίπτωση που υπάρχουν καινούργια αποτελέσματα για κάποια αναζήτησή του. Στην περίπτωση των φιλοξενούμενων συλλογών, φροντίζουμε ώστε η διαδικασία προσθήκης και διαχείρισης ειδοποιήσεων να παραμένει αμετάβλητη, ακριβώς όπως περιγράφηκε στην προηγούμενη ενότητα.

Από την πλευρά του εξυπηρετητή, ο μηχανισμός των ειδοποιήσεων συνεχίζει να εκτελείται ως μια διαδικασία του χρονοπρογραμματιστή, σε καθημερινή βάση. Η λειτουργία του ίδιου του μηχανισμού προσαρμόζεται στις φιλοξενούμενες συλλογές. Για την εύρεση των πιθανών νέων εγγραφών φιλοξενούμενων συλλογών δε χρησιμοποιούμε τις κλασικές συναρτήσεις της μονάδας WebSearch, που είναι βελτιστοποιημένες για τις τοπικές εγγραφές, αλλά προτιμούμε ειδικές συναρτήσεις οι οποίες χρησιμοποιούν απευθείας τις κλάσεις των φιλοξενούμενων συλλογών που περιγράφηκαν στην υποενότητα 4.2.1, σελίδα 49.

Αυτές οι συναρτήσεις μας επιστρέφουν τα επιθυμητά αποτελέσματα τα οποία μπορούμε στη συνέχεια να χρησιμοποιήσουμε για την προσθήκη των εγγραφών σε κάποιο καλάθι και τη δημιουργία του μηνύματος ειδοποίησης προς το χρήστη. Πρέπει βεβαίως η συνάρτηση προσθήκης εγγραφών σε κάποιο καλάθι της μονάδας WebBasket να μπορεί να δέχεται την προσθήκη τέτοιων εγγραφών. Τέλος, το μήνυμα ηλεκτρονικού ταχυδρομείου που λαμβάνει ο χρήστης πρέπει να παρουσιάζει με ομογενή τρόπο τα αποτελέσματα των τοπικών και φιλοξενούμενων συλλογών.

Στην υποενότητα 5.2.2, σελίδα 90, θα δούμε αναλυτικότερα πώς υλοποιείται η παραπάνω σχεδίαση.

5.2 Υλοποίηση

Στην προηγούμενη ενότητα έγινε μια ανάλυση της λειτουργίας των βασικών υπηρεσιών εξατομίκευσης του CDS Invenio και στη συνέχεια, στην υποενότητα 5.1.2, σελίδα 78, περιγράφηκε η προταθείσα σχεδίαση μιας υλοποίησης για τις υπηρεσίες αυτές σε ένα κατανεμημένο δίκτυο συνεργαζόμενων ψηφιακών βιβλιοθηκών. Στην παρούσα ενότητα θα παρουσιαστεί αναλυτικά η υλοποίηση αυτή.

5.2.1 Καλάθια

Όπως περιγράφηκε στην υποενότητα της σχεδίασης, κατά της προσθήκη σε καλάθια, εγγραφών που ανήκουν σε φιλοξενούμενες συλλογές, χρησιμοποιούμε για το χαρακτηρισμό τους το μοναδικό τους αναγνωριστικό από τον εξυπηρετητή απ' όπου προήρθαν, καθώς και το μοναδικό αναγνωριστικό της φιλοξενούμενης συλλογής στον εξυπηρετητή του κανανεμημένου δικτύου. Οι συναρτήσεις σχεδίασης των ιστοσελίδων των αποτελεσμάτων της αναζήτησης, αναλαμβάνουν να περιλαμβάνουν το αναγνωριστικό της εκάστοτε φιλοξενούμενης συλλογής, στην HTML φόρμα προσθήκης εγγραφών στα καλάθια. Όσον αφορά το μοναδικό αναγνωριστικό κάθε εγγραφής, ο αναλυτής που περιγράφηκε στην ομόνυμη παράγραφο της υποενότητας 4.2.1, σελίδα 49, αναλαμβάνει να το εξάγει από τα δεδομένα που λαμβάνει από τον προσκομιστή και να το συμπεριλάβει στην HTML φόρμα προσθήκης εγγραφών στα καλάθια, για κάθε εγγραφή.

Αφού ο χρήστης επιλέξει, μέσω της διεπαφής αναζήτησης και της HTML φόρμας, ποιές εγγραφές επιθυμεί να προσθέσει σε κάποιο καλάθι του, την προσθήκη αναλαμβάνει η μονάδα WebBasket. Οι εγγραφές στέλνονται στη βάση δεδομένων, με τα μοναδικά εξωτερικά αναγνωριστικά τους και το τοπικό αναγνωριστικό της συλλογής στην οποία ανήκουν. Πριν αναλύσουμε την υλοποίηση της συνάρτησης προσθήκης στη βάση δεδομένων, ας δούμε τους σχετικούς πίνακές της.

```
mysql> describe bskREC;
+-------------------------+---------------------+------+-----+
| Field                   | Type                | Null | Key |
+-------------------------+---------------------+------+-----+
| id_bibrec_or_bskEXTREC  | int(16)             | NO   | PRI |
| id_bskBASKET            | int(15) unsigned    | NO   | PRI |
| id_user_who_added_item  | int(15)             | NO   |     |
| score                   | int(15)             | NO   | MUL |
| date_added              | datetime            | NO   | MUL |
+-------------------------+---------------------+------+-----+

mysql> describe bskEXTREC;
+-------------------+---------------------+------+-----+
| Field             | Type                | Null | Key |
+-------------------+---------------------+------+-----+
| id                | int(15) unsigned    | NO   | PRI |
| external_id       | int(15)             | NO   |     |
```

```
| collection_id     | int(15) unsigned | NO  |     |
| original_url      | text             | YES |     |
| creation_date     | datetime         | NO  |     |
| modification_date | datetime         | NO  |     |
+-------------------+------------------+-----+-----+

mysql> describe bskEXTFMT;
+---------------+------------------+------+-----+
| Field         | Type             | Null | Key |
+---------------+------------------+------+-----+
| id            | int(15) unsigned | NO   | PRI |
| id_bskEXTREC  | int(15) unsigned | NO   | MUL |
| format        | varchar(10)      | NO   | MUL |
| last_updated  | datetime         | NO   |     |
| value         | longblob         | YES  |     |
+---------------+------------------+------+-----+
```

Ο πρώτος πίνακας, `bskREC`, αφορά όλες τις εγγραφές που βρίσκονται σε όλα τα καλάθια, και όπως προαναφέραμε, χρησιμοποιεί το μοναδικό αναγνωριστικό (θετικό ακέραιο) για τις τοπικές εγγράφες, και ένα μοναδικό αναγνωριστικό (αρνητικό ακέραιο), του οποίου η ανάθεση γίνεται επί τούτου, για όλες τις εγγραφές φιλοξενούμενων συλλογών. Στον πίνακα αυτό το πεδίο ονομάζεται `id_bibrec_or_bskEXTREC`. Ο εν λόγω πίνακας περιέχει επίσης πληροφορίες, όπως σε ποιο καλάθι ανήκει η κάθε εγγραφή, ποιος χρήστης την πρόσθεσε αρχικά κ.α.

Ο δεύτερος πίνακας, `bskEXTREC`, είναι ειδικά για τις εγγραφές φιλοξενούμενων συλλογών, και περιέχει διάφορα από τα χαρακτηριστικά τους και τα δεδομένα τους, όπως το μοναδικό αναγνωριστικό που τους έχουμε αναθέσει (`id`), το αναγνωριστικό της συλλογής στην οποία ανήκουν (`collection_id`), το μοναδικό αναγνωριστικό που είχαν στον εξυπηρετητή από τον οποίο προήρθαν (`external_id`) και άλλα. Σημειώνεται εδώ ότι σε αυτόν τον πίνακα το πεδίο `id` είναι αυτό που αντιστοιχεί στο πεδίο `id_bibrec_or_bskEXTREC` του πρώτου πίνακα. Είναι οι ίδιοι δηλαδή μοναδικοί ακέραιοι, με εξαίρεση ότι στο δεύτερο πίνακα αποθηκεύουμε τον αντίθετο ακέραιο του αρνητικού ακεραίου του πρώτου πίνακα, ένα θετικό ακέραιο. Ο λόγος που το κάνουμε αυτό είναι γιατί τα μοναδικά αναγνωριστικά του δεύτερου πίνακα αναθέτονται αρχικά αυτομάτως, ως θετικοί ακέραιοι, όταν προσθέτουμε νέες εγγραφές μιας φιλοξενούμενης συλλογής σε κάποιο καλάθι και ακολούθως αποθηκέυονται, ως τα αντίθετα τους αρνητικά αναγνωριστικά, στον πρώτο πίνακα, όπως θα δούμε και στη συνάρτηση προσθήκης στη βάση δεδομένων παρακάτω. Σκοπός του πίνακα αυτού είναι να αποθηκεύυει μία μόνο φορά την ύπαρξη μιας εγγραφής κάποιας φιλοξενούμενης συλλογής στα καλάθια, ανεξαρτήτως από το αν αυτή η εγγραφή είναι αποθηκευμένη περισσότερες από μια φορά σε διαφορετικά καλάθια. Για κάθε μία από τις επιμέρους αυτές εμφανίσεις, η αποθήκευσή τους γίνεται στον πρώτο πίνακα.

Ο τρίτος πίνακας, `bskEXTFMT`, χρησιμοποιείται επίσης αποκλειστικά για τις εγγραφές φιλοξενούμενων συλλογών, και συγκεκριμένα για την αποθήκευση συμπιεσμένων προ-υπολογισμένων διαμορφώσεων για κάθε εγγραφή. Η συμπιεσμένη αυτή διαμόρφωση, που ουσιαστικά αποτελεί μια ιστοσελίδα ή μια δομή XML, εξασφαλίζει την γρήγορη προβολή των εγγραφών αυτών κάθε φορά που ο χρήστης προβάλλει το καλάθι στο οποίο ανήκουν, καθώς επίσης και την ταχεία αναζήτηση σε αυτές. Κάθε μια από αυτές τις διαμορφώσεις χαρακτηρίζεται από ένα τοπικό μοναδικό αναγνωριστικό (θετικό ακέραιο), `id`, καθώς και από το μοναδικό αναγνωριστικό που έχουμε αναθέσει στην εγγραφή, `id_bskEXTREC`, το οποίο φυσικά αντιστοιχεί στο πεδίο `id` του δεύτερου πίνακα. Με αυτόν τον τρόπο μπορούμε να αποθηκεύουμε διαφορετικούς τύπους διαμορφώσεων για κάθε εγγραφή. Άλλα πεδία του τρίτου πίνακα αφορούν στο είδος της διαμόρφωσης κάθε εγγραφής, στην τελευταία ημερομηνία αποθήκευσης, καθώς και βεβαίως στην ίδια την συμπιεσμένη διαμόρφωσή της.

Στη συνέχεια παρουσιάζεται και αναλύεται η συνάρτηση προσθήκης εγγραφών φιλοξενούμενων συλλογών στη βάση δεδομένων.

```python
def add_to_basket(uid,
                  recids=[],
                  colid=0,
                  bskid=0):

    if recids and bskid > 0:
        query_max_score = """   SELECT   MAX(score)
                                FROM     bskREC
                                WHERE    id_bskBASKET=%s"""
        params_max_score = (bskid,)
        res_max_score = run_sql(query_max_score, params_max_score)
        max_score = __wash_sql_count(res_max_score)
        if not max_score:
            max_score = 1

        if colid > 0:
            query_existing = """    SELECT  id,
                                            external_id
                                    FROM    bskEXTREC
                                    WHERE   %s
                                    AND     collection_id=%s"""
            sep_or = ' OR '
            query_existing %= (sep_or.join(['external_id=%s'] * len(recids)), colid)
            params_existing = tuple(recids)
            res_existing = run_sql(query_existing, params_existing)
            existing_recids = [int(external_ids_couple[1]) for external_ids_couple in res_existing]
            existing_ids = [int(ids[0]) for ids in res_existing]
            new_recids = [recid for recid in recids if int(recid) not in existing_recids]
            if new_recids:
                query_new = """ INSERT INTO bskEXTREC
```

```
                                                    (external_id,
                                                     collection_id,
                                                     creation_date,
                                                     modification_date)
                                        VALUES """
                now = convert_datestruct_to_datetext(localtime())
                records = ["(%s, %s, %s, %s)"] * len(new_recids)
                query_new += ', '.join(records)
                params_new = ()
                for new_recid in new_recids:
                    params_new += (int(new_recid), colid, now, now)
                res_new = run_sql(query_new, params_new)
                recids = [-int(recid) for recid in existing_ids]
                recids.extend(range(-res_new,-(res_new+len(new_recids))
                    ,-1))
            else:
                recids = [-int(recid) for recid in existing_ids]

        query_insert = """ INSERT IGNORE INTO bskREC
                                                    (id_bibrec_or_bskEXTREC,
                                                     id_bskBASKET,
                                                     id_user_who_added_item,
                                                     date_added,
                                                     score)
                           VALUES """
        if colid == 0 or (colid > 0 and not new_recids):
            now = convert_datestruct_to_datetext(localtime())
        records = ["(%s, %s, %s, %s, %s)"] * len(recids)
        query_insert += ', '.join(records)
        params_insert = ()
        i = 1
        for recid in recids:
            params_insert += (recid, bskid, uid, now, max_score + i)
            i += 1
        run_sql(query_insert, params_insert)

        query_update = """ UPDATE bskBASKET
                           SET    date_modification=%s
                           WHERE  id=%s"""
        params_update = (now, bskid)
        run_sql(query_update, params_update)
        return recids
    return 0
```

Η συνάρτηση ουσιαστικά προσθέτει όλες τις νέες εγγραφές στον πρώτο πίνακα που περιγράφηκε παραπάνω (bskREC), τοποθετώντας τις μετά την τελευταία εγγραφή στο επιλεγμένο καλάθι, ενώ αναλαμβάνει επίσης να ενημερώσει και την ημερομηνία τροποποίησης του καλαθιού. Εάν κάποιες από αυτές τις εγγραφές υπάρχουν ήδη στο επιλεγμένο καλάθι δεν προσθέτονται εκ νέου, εφόσον απορρίπτονται από το ερώτημα SQL προς τη βάση δεδομένων, μέσω των πρωτευόντων κλειδιών του. Για τις εγγραφές φιλοξενούμενων συλλογών εισάγεται ένας επιπλέον ειδικός έλεγχος, ο οποίος αναλαμβάνει να προσθέσει πρώτα αυτές τις εγγραφές στο δεύτερο πίνακα της βάσης δεδομένων (`bskEXTREC`). Όπως αναφέρθηκε παραπάνω, σκοπός του πίνακα αυτού είναι να αποθηκεύσει μία μόνο φορά την ύπαρξη μιας εγγραφής κάποιας φιλοξενούμενης συλλογής στα καλάθια. Κατά συνέπεια, καταρχήν διερευνάται αν και ποιές από τις προς προσθήκη εγγραφές υπάρχουν ήδη στον πίνακα. Στη συνέχεια, αν υπάρχουν νέες εγγραφές φιλοξενούμενων συλλογών, προσθέτονται στον πίνακα. Τέλος, τα μοναδικά αναγνωριστικά που αναθέσαμε αυτομάτως στις νέες εγγραφές, καθώς και αυτά των υπαρχουσών εγγραφών, αποθηκεύονται στον πρώτο πίνακα, αφού τα μετατρέψουμε στους αντίθετούς τους (δηλαδή σε αυτό το σημείο σε αρνητικούς ακεραίους), όπως αναλύσαμε ήδη παραπάνω.

Παράλληλα με την προσθήκη των εγγραφών στη βάση δεδομένων αναλαμβάνουμε και να υπολογίσουμε και να αποθηκεύσουμε στον τρίτο πίνακα της βάσης δεδομένων (`bskEXTFMT`) τις συμπιεσμένες διαμορφώσεις τους. Αυτό επιτυγχάνεται με τις παρακάτω συναρτήσεις.

```python
def format_external_records(recids,
                            of='hb'):

    formatted_records = []

    if type(recids) is not list:
        recids = [recids]

    existing_xml_formatted_records = db.get_external_records(recids, "xm")
    for existing_xml_formatted_record in existing_xml_formatted_records:
        xml_record = decompress(existing_xml_formatted_record[2])
        xml_record_id = existing_xml_formatted_record[1]
        xml_record_colid = existing_xml_formatted_record[0]
        recids.remove(-xml_record_id)
        if of == "hb":
            if xml_record_colid > 0:
                htmlbrief_record = format_record(None, of, xml_record=xml_record)
            formatted_records.append((xml_record_id, htmlbrief_record))
        elif of == "xm":
            formatted_records.append((xml_record_id, xml_record))

    if formatted_records and of == "hb":
        db.store_external_records(formatted_records, of)

    records_grouped_by_collection = db.get_external_records_by_collection(recids)

    if records_grouped_by_collection:
```

```
        for records in records_grouped_by_collection:
            colid = records[2]
            if colid:
                external_records = fetch_and_store_external_records(records, of)
                formatted_records.extend(external_records)

    return formatted_records

def fetch_and_store_external_records(records,
                                     of="hb"):
    results = []
    formatted_records = []

    if of == 'xm':
        re_controlfield = re.compile(r'<controlfield\b[^>]*>.*?</controlfield>', re.DOTALL + re.MULTILINE + re.IGNORECASE)
        re_blankline = re.compile(r'\s*\n', re.DOTALL + re.MULTILINE + re.IGNORECASE)

    local_ext_ids = records[0].split(",")
    external_ids = records[1].split(",")
    collection_name = get_collection_name_by_id(records[2])
    collection_engine_set = select_hosted_search_engines(collection_name)
    collection_engine = collection_engine_set.pop()

    external_ids_urls = collection_engine.build_record_urls(external_ids)
    external_urls = [external_id_url[1] for external_id_url in external_ids_urls]
    db.store_external_urls(zip(local_ext_ids, external_urls))

    url = collection_engine.build_search_url(None, req_args=external_ids)
    pagegetters = [HTTPAsyncPageGetter(url)]

    def finished(pagegetter, dummy_data, dummy_time):
        results.append(pagegetter)

    finished_list = async_download(pagegetters, finish_function=finished, timeout=CFG_EXTERNAL_COLLECTION_TIMEOUT)

    if finished_list[0]:
        collection_engine.parser.parse_and_get_results(results[0].data, feedonly=True)
        (dummy, parsed_results_dict) = collection_engine.parser.parse_and_extract_records(of=of)
        for (local_ext_id, external_id) in zip(local_ext_ids,
```

```
            external_ids):
            formatted_record = parsed_results_dict[external_id]
            if of == 'xm':
                formatted_record = re_controlfield.sub('',
                    formatted_record)
                formatted_record = re_blankline.sub('\n',
                    formatted_record)
            formatted_records.append((int(local_ext_id),
                formatted_record))
        db.store_external_records(formatted_records, of)
    else:
        for (local_ext_id, external_id) in zip(local_ext_ids,
            external_ids):
            formatted_records.append((int(local_ext_id), "There was a
                timeout when fetching the record."))

    return formatted_records

def store_external_records(records,
                           of="hb"):

    if records:
        query = """ INSERT
                    INTO bskEXTFMT
                        (id_bskEXTREC,
                        format,
                        last_updated,
                        value)
                    VALUES """
        now = convert_datestruct_to_datetext(localtime())
        formatted_records = ["(%s, %s, %s, %s)"] * len(records)
        query += ', '.join(formatted_records)
        params = ()
        for record in records:
            params += (record[0], of, now, compress(record[1]))
        run_sql(query,params)
```

Οι παραπάνω συναρτήσεις αναλαμβάνουν να υπολογίσουν τις διαμορφώσεις των δοσμένων εγγραφών στον επιθυμητό τύπο, να τις αποθηκεύσουν στη βάση δεδομένων και να τις επιστρέψουν στο χρήστη. Αν οι εγγραφές υπάρχουν ήδη στη βάση δεδομένων υπό τον μορφή MARC XML, τότε υπολογίζουμε κατευθείαν οποιαδήποτε άλλη ζητούμενη μορφή χρησιμοποιώντας τις συναρτήσεις μορφοποίησης του CDS Invenio. Σε αντίθετη περίπτωση χρησιμποποιούμε, όπως παρατηρούμε, τις κλάσεις της μονάδας WebSearch για τις φιλοξενούμενες συλλογές. Οι παραπάνω συναρτήσεις χρησιμοποιούνται επίσης και από τη διεπαφή προβολής των καλαθιών, για τις εγγραφές φιλοξενούμενων συλλογών.

Για την αναζήτηση των περιεχομένων των καλαθιών τα οποία περιλαμβάνουν και εγγραφές φιλοξενούμενων συλλογών εκμεταλλευόμαστε το γεγονός ότι έχουμε ήδη αποθηκεύσει στη βάση δεδομένων τουλάχιστον ένα τύπο διαμόρφωσης για κάθε εγγραφή. Ο σχετικός κώδικας φαίνεται παρακάτω.

```
    if p:
        personal_search_results = None
        total_no_personal_search_results = 0
        local_search_results = set(search_unit(p))

        pattern = re.compile(r'%s' % (re.escape(p),), re.DOTALL + re.
            MULTILINE + re.IGNORECASE + re.UNICODE)
        format = 'xm'

        if b.startswith("P") or not b:
            personal_search_results = {}
            personal_items = db.get_all_items_in_user_personal_baskets(
                uid, selected_topic, format)
            personal_local_items = personal_items[0]
            personal_external_items = personal_items[1]

            for local_info_per_basket in personal_local_items:
                bskid       = local_info_per_basket[0]
                basket_name = local_info_per_basket[1]
                topic       = local_info_per_basket[2]
                recid_list  = local_info_per_basket[3]
                local_recids_per_basket = set(eval(recid_list + ','))
                intsec = local_search_results.intersection(
                    local_recids_per_basket)
                if intsec:
                    personal_search_results[bskid] = [basket_name, topic
                        , len(intsec), list(intsec)]
                    total_no_personal_search_results += len(intsec)

            for external_info_per_basket in personal_external_items:
                bskid       = external_info_per_basket[0]
                basket_name = external_info_per_basket[1]
                topic       = external_info_per_basket[2]
                recid       = external_info_per_basket[3]
                value       = external_info_per_basket[4]
                text = remove_html_markup(decompress(value))
                result = pattern.search(text)
                if result:
                    if personal_search_results.has_key(bskid):
                        personal_search_results[bskid][2] += 1
                        personal_search_results[bskid][3].append(recid)
                    else:
                        personal_search_results[bskid] = [basket_name,
                            topic, 1, [recid]]
                    total_no_personal_search_results += 1

```

```
            if n:
                personal_items_by_matching_notes = db.
                    get_all_items_in_user_personal_baskets_by_matching_notes
                    (uid, selected_topic, p)
                for info_per_basket_by_matching_notes in
                    personal_items_by_matching_notes:
                    bskid       = info_per_basket_by_matching_notes[0]
                    basket_name = info_per_basket_by_matching_notes[1]
                    topic       = info_per_basket_by_matching_notes[2]
                    recid_list  = info_per_basket_by_matching_notes[3]
                    recids_per_basket_by_matching_notes = set(eval(
                        recid_list + ','))
                    if personal_search_results.has_key(bskid):
                        no_personal_search_results_per_basket_so_far =
                            personal_search_results[bskid][2]
                        personal_search_results[bskid][3] = list(set(
                            personal_search_results[bskid][3]).union(
                            recids_per_basket_by_matching_notes))
                        personal_search_results[bskid][2] = len(
                            personal_search_results[bskid][3])
                        total_no_personal_search_results += (
                            personal_search_results[bskid][2] -
                            no_personal_search_results_per_basket_so_far
                            )
                    else:
                        personal_search_results[bskid] = [basket_name,
                            topic, len(
                            recids_per_basket_by_matching_notes), list(
                            recids_per_basket_by_matching_notes)]
                        total_no_personal_search_results += len(
                            recids_per_basket_by_matching_notes)
```

Σε αυτό το τμήμα κώδικα πραγματοποιείται η αναζήτηση στα προσωπικά καλάθια ενός χρήστη. Η ίδια διαδικασία πραγματοποιείται και για τα καλάθια τα οποία μοιράζεται με ομάδες χρηστών, καθώς και για τα δημόσια καλάθια, στα οποία είναι συνδρομητής, με αντίστοιχες συναρτήσεις. Καταρχήν συλλέγουμε από τη βάση δεδομένων όλες τις εγγραφές που ανήκουν στα προσωπικά καλάθια του χρήστη. Από αυτές, για τις τοπικές εγγραφές, όπως περιγράφηκε και στην υποενότητα 5.1.1, σελίδα 69, χρησιμοποιούμε τη συνάρτηση `search_unit` της μονάδας WebSearch για την εύρεση όλων των εγγραφών του εξυπηρετητή που ταιριάζουν, και στη συνέχεια βρίσκουμε ποιες εγγραφές πληρούν τα κριτήρια της αναζήτησης κάνοντας τομή με τα περιεχόμενα του καλαθιού. Για τις εγγραφές των φιλοξενούμενων συλλογών, όμως, κάτι τέτοιο δεν είναι δυνατό. Γι αυτές τις εγγραφές καταρχήν ορίζουμε μια κανονική έκφραση για την αναζήτηση του ερωτήματος που έθεσε ο χρήστης, στη συνέχεια αποσυμπιέζουμε τη διαμόρφωση κάθε εγγραφής που έχουμε αποθηκεύσει στη βάση δεδομένων και, τέλος, εκτελούμε την κανονική έκφραση σε αυτήν. Για την αναζήτηση στα σχόλια οποιασδήποτε εγγραφής χρησιμοποιούμε απευθείας το ταίριασμα του ερωτήματος αναζήτησης του χρήστη με τα σχόλια που βρίσκονται στη βάση δεδομένων, αφού αυτά δεν είναι αποθηκευμένα υπό συμπίεση.

5.2.2 Ειδοποιήσεις

Ο μηχανισμός των ειδοποιήσεων, όπως περιγράφηκε στην υποενότητα της σχεδίασης, χρησιμοποιεί ειδικές συναρτήσεις για την εύρεση νέων εγγραφών φιλοξενούμενων συλλογών, σύμφωνα με τις παραμέτρους που έχει θέσει ο χρήστης. Οι παράμετροι αυτοί είναι ουσιαστικά οι παράμετροι του ερωτήματος αναζήτησης για το οποίο έχει οριστεί η ειδοποίηση, καθώς και οι ημερομηνίες εμφάνισης των νέων εγγραφών σύμφωνα με τη συχνότητα της ειδοποίησης. Την εύρεση αυτών των εγγραφών, καθώς και των εγγραφών των τοπικών συλλογών, αναλαμβάνει η παρακάτω συνάρτηση.

```
def get_record_ids(argstr,
                   date_from,
                   date_until):

    argd = wash_urlargd(parse_qs(argstr), websearch_templates.
        search_results_default_urlargd)
    p       = argd.get('p', [])
    c       = argd.get('c', [])
    cc      = argd.get('cc', [])
    aas     = argd.get('aas', [])
    f       = argd.get('f', [])
    so      = argd.get('so', [])
    sp      = argd.get('sp', [])
    ot      = argd.get('ot', [])
    p1      = argd.get('p1', [])
    f1      = argd.get('f1', [])
    m1      = argd.get('m1', [])
    op1     = argd.get('op1', [])
    p2      = argd.get('p2', [])
    f2      = argd.get('f2', [])
    m2      = argd.get('m2', [])
    op2     = argd.get('op3', [])
    p3      = argd.get('p3', [])
    f3      = argd.get('f3', [])
    m3      = argd.get('m3', [])
    sc      = argd.get('sc', [])

    d1y, d1m, d1d = _date_to_tuple(date_from)
    d2y, d2m, d2d = _date_to_tuple(date_until)

    washed_colls = wash_colls(cc, c, sc, 0)
    hosted_colls = washed_colls[3]
    if hosted_colls:
```

```
        req_args = "p=%s&f=%s&d1d=%s&d1m=%s&d1y=%s&d2d=%s&d2m=%s&d2y=%s&ap=%i" % (p, f, d1d, d1m, d1y, d2d, d2m, d2y, 0)
        external_records = calculate_external_records(req_args, [p, p1, p2, p3], f, hosted_colls, CFG_EXTERNAL_COLLECTION_TIMEOUT, CFG_EXTERNAL_COLLECTION_MAXRESULTS_ALERTS)
    else:
        external_records = ([],[])

    recids = perform_request_search(of='id', p=p, c=c, cc=cc, f=f, so=so, sp=sp, ot=ot,
                                    aas=aas, p1=p1, f1=f1, m1=m1, op1=op1, p2=p2, f2=f2,
                                    m2=m2, op2=op2, p3=p3, f3=f3, m3=m3, sc=sc, d1y=d1y,
                                    d1m=d1m, d1d=d1d, d2y=d2y, d2m=d2m, d2d=d2d)

    return (recids, external_records)
```

Η συνάρτηση αυτή καταρχήν απομονώνει τις παραμέτρους που μας ενδιαφέρουν από όλες τις παραμέτρους του ερωτήματος αναζήτησης και υπολογίζει τις ημερομηνίες που αντιστοιχούν στη συχνότητα της ειδοποίησης. Από αυτές τις παραμέτρους υπολογίζει τις φιλοξενούμενες συλλογές στις οποίες χρειάζεται να αναζητήσουμε νέες εγγραφές και στη συνέχεια καλεί μια νέα συνάρτηση, η οποία υπολογίζει τις εγγραφές αυτές. Η συνάρτηση αυτή, καθώς και οι συναρτήσεις τις οποίες καλεί στην πορεία, φαίνονται παρακάτω.

```
def calculate_external_records(req_args,
                               pattern_list,
                               field,
                               hosted_colls,
                               timeout=CFG_EXTERNAL_COLLECTION_TIMEOUT,
                               limit=CFG_EXTERNAL_COLLECTION_MAXRESULTS_ALERTS):

    (external_search_engines, basic_search_units) = calculate_external_search_params(pattern_list, field, hosted_colls)

    return do_calculate_external_records(req_args, basic_search_units, external_search_engines, timeout, limit)

def calculate_external_search_params(pattern_list,
                                     field,
                                     hosted_colls):

    pattern = bind_patterns(pattern_list)
```

```python
    basic_search_units = create_basic_search_units(None, pattern, field)

    external_search_engines = select_external_search_engines(hosted_colls)

    return (external_search_engines, basic_search_units)

def do_calculate_external_records(req_args,
                                  basic_search_units,
                                  external_search_engines,
                                  timeout=CFG_EXTERNAL_COLLECTION_TIMEOUT,
                                  limit=CFG_EXTERNAL_COLLECTION_MAXRESULTS_ALERTS):

    engines_list = []
    results_list = []
    full_results_list = []
    timeout_list = []

    for engine in external_search_engines:
        url = engine.build_search_url(basic_search_units, req_args, limit=limit)
        if url:
            engines_list.append([url, engine])

    pagegetters_list = [HTTPAsyncPageGetter(engine[0]) for engine in engines_list]

    def finished(pagegetter, data, dummy_time):
        results_list.append((pagegetter, data))

    finished_list = async_download(pagegetters_list, finished, engines_list, timeout)

    for (finished, engine) in zip(finished_list, engines_list):
        if finished:
            for result in results_list:
                if result[1] == engine:
                    engine[1].parser.parse_and_get_results(result[0].data, feedonly=True)
                    full_results_list.append((engine[1].name, engine[1].parser.parse_and_extract_records(of="xm")))
                    break
        elif not finished:
            timeout_list.append(engine[1].name)

    return (full_results_list, timeout_list)
```

Οι συναρτήσεις αυτές χρησιμοποιούν απευθείας τις συναρτήσεις των κλάσεων των φιλοξενούμενων συλλογών, δηλαδή της μηχανής αναζήτησης, του προσκομιστή και του αναλυτή. Αφού υπολογισθούν τα αποτελέσματα, οι εγγραφές δηλαδή που πληρούν τις παραμέτρους της αναζήτησης, τις προσθέτουμε στο καλάθι που έχει ορίσει ο χρήστης ή/και του αποστέλλεται ένα μήνυμα ηλεκτρονικού ταχυδρομείου.

Η προσθήκη των εγγραφών στο καλάθι πρακτικά δεν απαιτεί περαιτέρω τροποποιήσεις σε αυτό το σημείο για τις εγγραφές φιλοξενούμενων συλλογών, εφόσον αυτές έχουν ήδη γίνει στην υλοποίηση της μονάδας WebBasket. Χρειάζεται απλά ο υπολογισμός του μοναδικού αναγνωριστικού της φιλοξενούμενης συλλογής, ενώ πραγματοποιείται επίσης ο υπολογισμός και η αποθήκευση των διαμορφώσεων των νεών εγγραφών στη βάση δεδομένων όπως περιγράφεται στην υποενότητα 5.2.1, σελίδα 81. Η αποθήκευση των διαμορφώσεων σε αυτό το σημείο, οι οποίες γίνονται στη μορφή MARC XML, εξυπηρετούν το χρήστη να πραγματοποιήσει αναζητήσεις στα καλάθια του συμπεριλαμβάνοντας τις νέες αυτές εγγραφές. Ο σχετικός κώδικας της προσθήκης των εγγραφών στο καλάθι φαίνεται παρακάτω.

```
owner_uid = get_basket_owner_id(basket_id)
collection_id = get_collection_id(external_collection_results[0])
added_items = add_to_basket(owner_uid, external_collection_results[1][0][:nrec_to_add], collection_id, basket_id)
format_external_records(added_items, of="xm")
```

Η δημιουργία και αποστολή του μηνύματος ηλεκτρονικού ταχυδρομείου στο χρήστη παραμένει η ίδια, εκτός από μερικές προσαρμογές, όπως για παράδειγμα ο υπολογισμός των διαφόρων συλλογών που έχουν παράγει αποτελέσματα με την παρακάτω συνάρτηση.

```
def calculate_desired_collection_list(c,
                                      cc,
                                      sc):

    if not cc[0]:
        cc = [CFG_SITE_NAME]

    is_not_hosted_collection = lambda coll: not is_hosted_collection(coll)
    washed_cc_sons = filter(is_not_hosted_collection, get_coll_sons(cc[0]))
    washed_c = filter(is_not_hosted_collection, c)

    if washed_cc_sons == washed_c:
        if sc == 0:
            return cc
        elif sc == 1:
            return washed_c
    else:
        if sc == 0:
```

```python
            return washed_c
        elif sc == 1:
            washed_c_sons = []
            for coll in washed_c:
                if coll in washed_cc_sons:
                    washed_c_sons.extend(get_coll_sons(coll))
                else:
                    washed_c_sons.append(coll)
            return washed_c_sons
```

Τέλος, σημειώνεται εδώ ότι το μέγιστο όριο νέων εγγραφών για τις οποίες μπορεί να ειδοποιηθεί ο χρήστης ισχύει και για τις εγγραφές φιλοξενούμενων συλλογών, τόσο όσον αφορά την προσθήκη αυτών σε κάποιο καλάθι, όσο και τη λίστα αυτών στο μήνυμα ειδοποίησης προς το χρήστη.

Κεφάλαιο 6

Επίλογος

Με βάση τις ιδιότητες και τον τρόπο λειτουργίας του λογισμικοαύ CDS Invenio, που αναπτύσσεται στο CERN, αναλύσαμε τη σχεδίαση και υλοποίηση των αλλαγών για τη δημιουργία ενός κατανεμημένου δικτύου συνεργαζόμενων ψηφιακών βιβλιοθηκών, με έμφαση την αναζήτηση και τις υπηρεσίες εξατομίκευσης.

Αντιμετωπίσαμε το αναμενόμενο πρόβλημα της αβεβαιότητας στη διαθεσιμότητα πόρων του δικτύου, καθώς και των έμμεσων καθυστερήσεων που αυτό εισάγει, διατηρώντας την ταχύτητα απόκρισης του λογισμικού σε αποδεκτά επίπεδα, με σχετική ενημέρωση του τελικού χρήστη. Επίσης, αντιμετωπίσαμε την αναμενόμενη αδυναμία της κατανεμημένης αναζήτησης, την έλλειψη δηλαδή πλήρους ελέγχου και διαχείρισης των διαφόρων πόρων του κατανεμημένου δικτύου των ψηφιακών βιβλιοθηκών.

Μέσω της προτεινόμενης υλοποίησης, καταλήξαμε στο συμπέρασμα ότι η αναζήτηση και οι υπηρεσίες εξατομίκευσης σε ένα κατανεμημένο δίκτυο ψηφιακών βιβλιοθηκών μπορούν να προσφέρουν σε μεγάλο βαθμό την ίδια ποιότητα με αυτή μιας αυτόνομης ψηφιακής βιβλιοθήκης.

6.1 Αξιοποίηση στην πράξη

Οι αλλαγές που προτάθηκαν και περιγράφονται στο τέταρτο και πέμπτο κεφάλαιο της παρούσας εργασίας, περιλαμβάνονται ήδη στην τελευταία πειραματική έκδοση του λογισμικού του CDS Invenio [15], ενώ θα αποτελέσουν και τμήμα της επερχόμενης έκδοσης 1.0 του λογισμικού. Συγχρόνως προορίζονται για χρήση από τον ίδιο τον εξυπηρετητή αρχείων του CERN, μετά την έναρξη του έργου INSPIRE.

Το INSPIRE έχει ως σκοπό τη δημιουργία μιας συγκεντρωτικής ψηφιακής βιβλιοθήκης για όλα τα έγγραφα σχετικά με φυσική υψηλών ενεργειών, και θα είναι βασισμένο στο λογισμικό CDS Invenio. Όσα σχετικά έγγραφα φιλοξενούνται τώρα από τον εξυπηρετητή αρχείων του CERN, θα μεταφερθούν στον εξυπηρετητή αρχείων του INSPIRE. Στόχος είναι αυτά τα έγγραφα, με χρήση των προτεινόμενων αλλαγών, να συνεχίσουν να φιλοξενούνται και από τον εξυπηρετητή αρχείων του CERN, αυτή τη φορά ως έγγραφα σε ένα κατανεμημένο δίκτυο συνεργαζόμενων ψηφιακών βιβλιοθηκών. Ειδικότερος στόχος της μετάβασης αυτής είναι να μην έχει αντίκτυπο στον τελικό χρήστη. Για το λόγο αυτό, παράλληλα με την αξιοποίηση των χαρακτηριστικών που προτάθηκαν, υλοποιήθηκαν και συναρτήσεις ομαλής μετάβασης των υπαρχουσών εγγραφών από τις τοπικές συλλογές στις φιλοξενούμενες.

6.2 Μελλονικές επεκτάσεις

Η παρούσα εργασία αποτελεί μια αξιοποίησιμη λύση για την αναζήτηση και τις υπηρεσίες εξατομίκευσης σε ένα κατανεμημένο δίκτυο συνεργαζόμενων ψηφιακών βιβλιοθηκών, επιδέχεται όμως βελτιώσεις και επεκτάσεις.

Όσον αφορά την αναζήτηση, προβλέπεται η υποστήριξη περισσότερων μορφών προβολής για τις εγγραφές φιλοξενούμενων συλλογών, καθώς επίσης και η υποστήριξη περισσότερων παραμέτρων για τα ερωτήματα αναζήτησης. Επίσης, στα άμεσα σχέδια μας, περιλαμβάνεται η δημιουργία μιας εύχρηστης διεπαφής πλήρους ορισμού και επεξεργασίας φιλοξενούμενων συλλογών, όπως και η περαιτέρω αξιοποίηση του ερωτήματος αναζήτησής τους. Τέλος, σχεδιάζεται η υλοποιήση ενός μετατροπέα XML, που θα επιτρέπει το χειρισμό διαφορετικών δομών μεταδεδομένων ανάμεσα στα διάφορα μέλη του κατανεμημένου δικτύου συνεργαζόμενων ψηφιακών βιβλιοθηκών. Όσον αφορά τις υπηρεσίες εξατομίκευσης, μια άμεσα προτεινόμενη προσθήκη είναι η επιλογή ενημέρωσης των περιεχόμενων ενός καλαθιού, του ελέγχου δηλαδή για τις τελευταίες ενημερωμένες εκδόσεις εγγραφών φιλοξενούμενων συλλογών.

Βιβλιογραφία

[1] Pepe, Alberto ; Baron, Thomas ; Gracco, Maja ; Le Meur, Jean-Yves ; Robinson, Nicholas ; Simko, Tibor ; Vesely, Martin. (2005) CERN Document Server Software: the integrated digital library. In: 9th ICCC International Conference on Electronic Publishing : From Author to Reader : Challenges for the Digital Content Chain, Leuven, Belgium, 8 - 10 Jun 2005, pp.297-302.

[2] Pepe, A ; Le Meur, Jean-Yves ; Simko, T. (2006) Dissemination of Scientific Results in High Energy Physics : the CERN Document Server Vision. In: 15th International Conference on Computing In High Energy and Nuclear Physics, Mumbai, India, 13 - 17 Feb 2006, pp.466-469.

[3] Robinson, N ; Le Meur, Jean-Yves ; Simko, T. (2007) Managing an Institutional Repository with CDS Invenio. In: International Conference on Computing in High Energy and Nuclear Physics, Victoria, Canada, 2 - 7 Sep 2007.

[4] Vesely, M ; Baron, T ; Le Meur, Jean-Yves ; Simko, Tibor. (2002) Creating Open Digital Library Using XML : Implementation of OAi-PMH Protocol at CERN. In: International Conference on Electronic Publishing, Karlovy Vary, Czeck Republic, 6 - 8 Nov 2002, pp.231-237.

[5] Pepe, A ; Simko, T. (2005) Open Access aspects at the CERN Document Server. In: ELAG 2005 conference, Geneva (Switzerland), 1-3 June 2005.

[6] Kaplun, S. (2007) Authentication/authorization issues and fulltext document migration for the CERN Document Server. MSc, CERN, Geneva, IT Department.

[7] Caffaro, J ; Le Meur, J-Y (dir.) ; Pu Faltings, Pearl (dir.). (2006) Improving the Formatting Tools of CDS Invenio. MSc, Ecole Polytechnique Fédérale de Lausanne, 2006.

[8] Vesely, M. ; Baron, T. ; Le Meur, J.-Y. ; Simko, T. (2003) CERN Document Server: Document Management System for Grey Literature in Networked Environment. In: GL5 conference, Amsterdam (The Netherlands), 4-5 December 2003.

[9] Le Meur, Jean-Yves. (2008) Information systems in HEP get INSPIREd. From: CERN computer newsletter No.2008-002.

[10] Comparison of Selected Software Systems for Creation of Digital Libraries. From: Field of Open Source for the Needs of the NRGL STL.

[11] Naomi Dushay ; James C. French ; Carl Lagoze. (1999) Using Query Mediators for Distributed Searching in Federated Digital Libraries. In: ACM DL99 Conference, arXiv:cs/9902020v1.

[12] Carl Logoze, David Fielding. (1998) Defining Collections in Distributed Digital Libraries. From: D-Lib Magazine, November 1998, ISSN 1082-9873.

[13] http://invenio-demo.cern.ch/help/admin/

[14] http://invenio-demo.cern.ch/help/hacking/

[15] http://cdsware.cern.ch/repo/?p=cds-invenio.git

[16] http://www.eprints.org/

[17] http://www.dspace.org/

[18] http://www.fedora-commons.org/

[19] http://www.greenstone.org/

[20] http://www.openarchives.org/

[21] http://el.wikipedia.org/wiki/Ψηφιακή_βιβλιοθήκη

[22] http://el.wikipedia.org/wiki/Σωματιδιακή_φυσική

[23] http://el.wikipedia.org/wiki/CERN