

Αυτόνομη διαδικτυακή πλατφόρμα για την περιήγηση και εξερεύνηση σημασιολογικών (RDF) δεδομένων

**Alexandros Vasilakis  
CSD 3612**

Επίβλεψη: Παύλος Φαφαλιός

Επόπτης καθηγητής: Γιάννης Τζίτζικας

Μέλος της Επιτροπής Παρακολούθησης/Αξιολόγησης: Δημήτρης Πλεξουσάκης

Ηράκλειο, Ιούλιος 2024

Table of Contents

[Λίστα εικόνων 3](#_Toc170400325)

[Περίληψη 5](#_Toc170400326)

[Υπόβαθρο και Πλαίσιο Εργασίας 6](#_Toc170400327)

[Το μοντέλο δεδομένων RDF 6](#_Toc170400328)

[URI 6](#_Toc170400329)

[SPARQL 6](#_Toc170400330)

[Το έργο SeaLiT και τα δεδομένα του 7](#_Toc170400331)

[Δεδομένα εφαρμογής 8](#_Toc170400332)

[Εφαρμογή Πλοήγησης σε RDF δεδομένα 9](#_Toc170400333)

[Αρχιτεκτονική και Βασικές λειτουργίες συστήματος 9](#_Toc170400334)

[**Ταξινόμηση** 9](#_Toc170400335)

[**Φίλτρα** 11](#_Toc170400336)

[**Διαγράμματα(Charts)** 15](#_Toc170400337)

[Διεπαφή χρήστη (User Interface) 16](#_Toc170400338)

[Τεχνολογίες 22](#_Toc170400339)

[Γλώσσα Προγραμματισμού 22](#_Toc170400340)

[Βιβλιοθήκες που χρησιμοποιήθηκαν: 22](#_Toc170400341)

[Μοντέλο Διαμόρφωσης(Configuration model) 24](#_Toc170400342)

[Ιδιότητες αρχείου διαμόρφωσης 24](#_Toc170400343)

[Σχεδιασμός αρχείου διαμόρφωσης 24](#_Toc170400344)

[Παράδειγμα αρχείου διαμόρφωσης 24](#_Toc170400345)

[Εγκατάσταση Συστήματος 29](#_Toc170400346)

[Απαιτήσεις Συστήματος 30](#_Toc170400347)

[Troubleshooting 30](#_Toc170400348)

[Επίλογος 31](#_Toc170400349)

[Βασικά συμπεράσματα 31](#_Toc170400350)

[Αδυναμίες 31](#_Toc170400351)

[Μελλοντική εργασία (Βελτιστοποίηση εφαρμογής – Βελτιώσεις) 31](#_Toc170400352)

[Αναφορές / Πηγές 31](#_Toc170400353)

[Παραρτήματα 31](#_Toc170400354)

# Λίστα εικόνων

[Εικόνα 1: Ταξινόμηση Α-Ζ 11](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170581201)

[Εικόνα 2: Ταξινόμηση Ζ-Α 11](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170581202)

[Εικόνα 3: Τυχαία ταξινόμηση ενός column 12](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170581203)

[Εικόνα 4: Αύξουσα Ταξινόμηση 13](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170581204)

[Εικόνα 5: Φθίνουσα Ταξινόμηση 13](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170581205)

[Εικόνα 6: Εμφάνιση αποτελεσμάτων με κενή τιμή καταχώρησης 14](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170581206)

[Εικόνα 7:Αναζήτηση αποτελεσμάτων που "ξεκινούν με" 14](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170581207)

[Εικόνα 8: Αναζήτηση αριθμούς "μεγαλύτερων ή ίσων από" 15](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170581208)

[Εικόνα 9: Αναζήτηση αριθμών ενός εύρους τιμών 15](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170581209)

[Εικόνα 10: Αναζήτηση ημερομηνιών "μετά από την" 16](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170581210)

[Εικόνα 11: Αναζήτηση ακριβής ημερομηνίας 16](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170581211)

[Εικόνα 12: Ταξινόμηση που αρκεί να ισχύει **"ένα από τα δύο"** φίλτρα 17](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170581212)

[Εικόνα 13: Αρχική σελίδα εφαρμογής 18](#_Toc170581213)

[Εικόνα 14: Σελίδα εμφάνισης μιας κατηγορίας 19](#_Toc170581214)

[Εικόνα 15: Εμφάνιση διαγραμμάτων columns μιας κατηγορίας 20](#_Toc170581215)

[Εικόνα 16: Δυνατότητα εμφάνισης πολλαπλών διαγραμμάτων την ίδια στιγμή 21](#_Toc170581216)

[Εικόνα 17: Σελίδα με περισσότερες πληροφορίες μιας καταχώρησης 22](#_Toc170581217)

[Εικόνα 18: Ένα URI, μπορεί να κληθεί από ένα πεδίο πολλαπλές φορές 23](#_Toc170581218)

[Εικόνα 19: Αποκλειστικά βασικά στοιχεία της εφαρμογής 26](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170581219)

[Εικόνα 20: Πεδίο με αριθμό των κατηγοριών των οντοτήτων 27](#_Toc170581220)

[Εικόνα 21: Στοιχεία που αφορούν κάθε κατηγορία 27](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170581221)

[Εικόνα 22: Παράδειγμα με στοιχεία κάθε κατηγορίας 27](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170581222)

[Εικόνα 23: Σελίδα χωρίς την παράμετρο ΟPTIONAL στο query που δίνει τα αποτελέσματα 28](#_Toc170581223)

[Εικόνα 24: Σελίδα με την παράμετρο ΟPTIONAL στο query που δίνει τα αποτελέσματα 29](#_Toc170581224)

[Εικόνα 25: Σελίδα με περίπτωση που δεν έχουν όλες οι μεταβλητές τιμή 30](#_Toc170581225)

1 Εισαγωγή (1-2 σελίδες για τον σκοπό της εφαρμογής και τη λειτουργικότητά της + στο τέλος μία παράγραφο με σύντομη επισκόπηση των επόμενων κεφαλαίων)  
  
2 Υπόβαθρο και Πλαίσιο Εργασίας  
~~2.1 Το μοντέλο δεδομένων RDF (σύντομη περιγραφή του μοντέλου, ...)  
2.2 Το έργο SeaLiT και τα δεδομένα του (σύντομη περιγραφή έργου, τι πληροφορίες περιγράφουν τα δεδομένα,~~ ~~ποια είναι η οντολογία που τα περιγράφει, κτλ.)~~  
  
3. Εφαρμογή πλοήγησης σε RDF δεδομένα  
~~3.1 Αρχιτεκτονική και Βασικές λειτουργίες (τι υποστηρίζει το σύστημα -- φόρτωση αρχείων, επισκόπηση κατηγοριών, προβολή δεδομένων μιας κατηγορίας σε πίνακα, οπτικοποίηση δεδεδομένων πίνακα σε διάγραμμα, προβολή λεπτομερειών οντότητας, ...)~~  
~~3.2 Διεπαφή χρήστη (User Interface) (εδώ μπορείς να βάλεις πολλά screenshots με σύντομες περιγραφές)~~  
3.3 Τεχνολογίες (γλώσσα προγραμματισμού, βιβλιοθήκες java, βιβλιοθήκες javascript, κτλ. κτλ.)  
  
~~4. Μοντέλο διαμόρφωσης  
4.1 Ιδιότητες αρχείου διαμόρφωσης  
4.2 Παράδειγμα αρχείου διαμόρφωσης~~  
  
5. Εγκατάσταση εφαρμογής και απαιτήσεις συστήματος  
(τρόπος/βήματα εγκατάστασης, απαιτήσεις υλικού/λογισμικού, κτλ.)  
  
6. Επίλογος  
6.1 Βασικά συμπεράσματα  
6.2 Αδυναμίες  
6.3 Μελλοντική εργασία

# Περίληψη

Ο σκοπός της παρούσας διπλωματικής εργασίας ήταν η δημιουργία μιας αυτόνομης διαδικτυακής πλατφόρμας για την εύκολη και γρήγορη περιήγηση και εξερεύνηση σημασιολογικών δεδομένων. Η προτεινόμενη μέθοδος αποθήκευσης και μεταφοράς δεδομένων είναι μια τεχνολογία του Σημασιολογικού Ιστού, το RDF. Με τον όριο «αυτόνομη» διαδικτυακή πλατφόρμα, σημαίνει ότι δεν απαιτεί την σύνδεση με κάποιο σημασιολογικό αποθετήριο / triplestores, αλλά φορτώνει στη μνήμη τον γράφο γνώσης για την εκτέλεση των (SPARQL) ερωτημάτων. Σύμφωνα με το RDF(Resource Description Framework), τα δεδομένα αποθηκεύονται με την μορφή triples(subject-predicate-object)(τριπλέτες(Υποκείμενο-Κατηγόρημα-Αντικείμενο)).

Η SPARQL αποτελεί την βασική γλώσσα για την υποβολή ερωτημάτων και χρησιμοποιείται για την επεξεργασία μιας RDF βάσης δεδομένων.

Προκειμένου να γίνει η φόρτωση και η διαχείριση των RDF δεδομένων ήταν απαραίτητο να χρησιμοποιηθεί ένας εξυπηρετητής δικτύου (web server). Στην συγκεκριμένη εφαρμογή χρησιμοποιήθηκε ο Αpache Tomcat, που αποτελεί έναν open source Semantic Web Framework(πλαίσιο ανοικτού κώδικα Σημασιολογικού Ιστού) που παρέχει υποστήριξη Java.

…

Στα πλαίσια της διπλωματικής εργασίας χρησιμοποιήθηκαν βιβλιοθήκες όπως AG Grid και Chart JS, για την δημιουργία πινάκων και διαγραμμάτων που αφορούσαν τα δεδομένα που που χρειάζονταν να διαχειριστούν

Η παρούσα εφαρμογή αποτελεί την δημιουργία ενός Java Web Application και χρήση της βιβλιοθήκης Apache Tomcat(server) για διάβασμα των RDF αρχείων που υπάρχουν σε ένα φάκελο

Εμφάνιση στην εφαρμογή των αποτελεσμάτων των SPARQL ερωτημάτων.

Το Apache Jena είναι ένα πλαίσιο ανοιχτού κώδικα Σημασιολογικού Ιστού για Java. Παρέχει ένα API για εξαγωγή δεδομένων και εγγραφή σε γραφήματα RDF

Δημιουργήθηκε ένα configuration αρχειου (config.properties) στο οποίο έχουμε ορίσει διάφορες παραμέτρους του συστήματος καθώς και λεπτομέρειες που αφορούν τα δεδομένα, όπως τις κατηγορίες οντοτήτων που μας ενδιαφέρει να εμφανίζονται στην εφαρμογή.

Χρησιμοποιήθηκαν δεδομένα του έργου SeaLit(στον τομέα της Ναυτικής Ιστορίας)

Οι βασικές λειτουργίες που υποστηρίζει η εφαρμογή είναι:

* Ταξινόμηση δεδομένων
* Σελιδοποίηση πίνακα
* Φιλτράρισμα αποτελεσμάτων
* Λεπτομερής περιγραφή μιας καταχώρησης
* Προβολή διαγράμματος μιας στήλης

Τα κεφάλαια που θα παρουσιαστούν παρακάτω είναι τα εξής:

* Υπόβαθρο και Πλαίσιο Εργασίας
* Εφαρμογή πλοήγησης σε RDF δεδομένα
* Μοντέλο διαμόρφωσης
* Εγκατάσταση εφαρμογής και απαιτήσεις συστήματος

# Υπόβαθρο και Πλαίσιο Εργασίας

## Το μοντέλο δεδομένων RDF

Το RDF (Resource Description Framework) αποτελεί ένα μοντέλο δεδομένων για το διαδίκτυο. Χρησιμοποιείται για την αναπαράσταση πληροφοριών που αφορούν πηγές του διαδικτύου, χωρίς να είναι απαραίτητο ότι τις παίρνουμε απευθείας από αυτό

(Μπορεί να υπάρχουν σε ένα βιβλίο, ή να αναφέρονται από κάποιον άνθρωπο).

Το RDF προορίζεται για καταστάσεις όπου οι πληροφορίες υποβάλλονται σε επεξεργασία από εφαρμογές. Αυτές οι εφαρμογές θα δώσουν το τελικό αποτέλεσμα που θα εμφανιστεί στον χρήστη.

Υπάρχουν διάφορες μορφές RDF δεδομένων:

* TRIG
* RDF/XML
* N-Triples
* N-Quads
* Turtle

Αυτή η μέθοδος χρησιμοποιείται για την καλύτερη σύνδεση, εύρεση και κοινοποίηση δεδομένων.

Ουσιαστικά υποδηλώνει τον Σημασιολογικό Ιστό(Semantic Web/Network)

Ο βασικός τρόπος αναπαράστασης των δεδομένων είναι triples(τριπλέτες)  
(subject-predicate-object)(Υποκείμενο-Κατηγόρημα-Αντικείμενο), όπου:

* Υποκείμενο: αποτελεί το «στοιχείο» που ενεργεί
* Κατηγόρημα: η ιδιότητα του υποκειμένου που μας ενδιαφέρει
* Αντικείμενο: Η τιμή ή η παραπομπή σε άλλο υποκείμενο(για να βρεθεί η τιμή) στην οποία γίνεται η ενέργεια του κατηγορήματος

## URI

Το URI (Uniform Resource Identifiers) αποτελεί ένα σύνολο χαρακτήρων που ορίζουν ένα όνομα ή μια αναφορά του διαδικτύου.

Τύποι URI είναι:

* URL (Uniform Recource Locator): Υποδηλώνει την τοποθεσία μιας αναφοράς στο διαδίκτυο. (Συνήθως ξεκινάει με ”https://” )
* URN (Uniform Resource Name): Αναφέρεται σε ένα όνομα, χωρίς την απαραίτητη χρήση μιας τοποθεσίας (όπως του URL)

## SPARQL

Η SPARQL (Structured Protocol and RDF Query Language) αποτελεί γλώσσα ερωτημάτων για την αναζήτηση και χρήση δεδομένων που είναι αποθηκευμένα σε RDF μορφή.

Η κύρια χρήση της είναι η αναζήτηση δεδομένων RDF, εξαγωγή πληροφοριών και τρόπο σύνδεσης μεταξύ τους.

Η δομή που φτιάχνεται ένα SPARQL ερώτημα είναι:

* SELECT: επιλέγει τα στοιχεία που θέλουμε να εμφανίσει μέσα από το RDF αρχείο
* WHERE: Καθορίζει τις συνθήκες που θα βρεθεί η ιδιότητα του στοιχείου που θέλουμε

Όπως αναφέρθηκε τα RDF αρχεία, αναπαρίστανται σαν τριπλέτες.

Έτσι και το πρότυπο αναζήτησης τους από το query, θα είναι αντίστοιχο.

Επιτρέπεται να χρησιμοποιηθούν συνθήκες (REPLACE) , φίλτρα (FILTER) , περιορισμοί(MAX), συγκεντρωτικές λειτουργίες (COUNT, AVG) για την βελτίωση των αποτελεσμάτων που θα επιστρέψει το ερώτημα.

## Το έργο SeaLiT και τα δεδομένα του

Το έργο SeaLit αποτελεί ένα project συνεργασίας ερευνητών της Μεσογείου

(ICS-FORTH-Hellas, University of Barcelona, University of Genoa, University of Aix-Marseille and the Centre for Cultural Informatics/ICS/FORTH) χρηματοδοτούμενο από το ERC Starting Grant 2016, που χρησιμοποιείται στον τομέα της Ναυτικής Ιστορίας.  
Το πλήρες όνομα του έργου είναι Seafaring Lives in Transition, Mediterranean Maritime Labour and Shipping, 1850s-1920s, αφού η μελέτη αφορά την διαδικασία μετάβασης από τα ιστιοπλοϊκά στα ατμόπλοια και τον τρόπο με τον οποίο οι ναυτικοί αντέδρασαν σε αυτήν την αλλαγή στις περιοχές της Μεσογείου και της Μαύρης Θάλασσας, κατά το χρονικό διάστημα 1850 και 1920(19ος αιώνας).

Αυτή η εποχή ήταν μια σημαντική αλλαγή για την ναυτική κοινότητα, αφού έκτοτε τα ατμόπλοια αποτέλεσαν τον κυρίαρχο τρόπο μεταφοράς στην θάλασσα.

Τα δεδομένα του έργου έχουν αντληθεί από διάφορες πηγές, 16 διαφορετικών τύπων αρχείων όπως ημερολόγια των πλοίων, βιβλία με διαδρομές, σημειώσεις, αφίξεις και αναχωρήσεις των πλοίων από κάθε λιμάνι, λίστες πληρωμάτων με ατομικές πληροφορίες μεταξύ αυτών όπως αναγνωριστικά, ονοματεπώνυμα, πληρωμές-μισθοδοσίες.

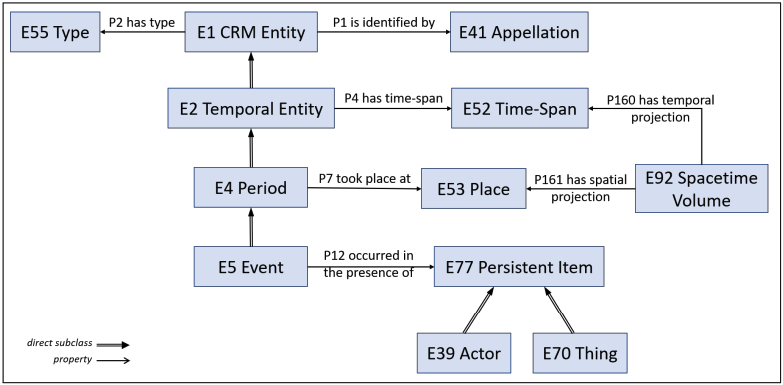
Κάποια από τα σημαντικότερα στοιχεία που χρησιμοποιήθηκαν στο παρόν project είναι:

* Πρόσωπα (Persons)
* Τοποθεσίες (Locations)
* Ταξίδια(Voyages)
* Πλοία (Ships)
* Ημερομηνίες (Dates)
* Χρονικά διαστήματα (Spans)
* Αναγνωριστικά (Identifiers)

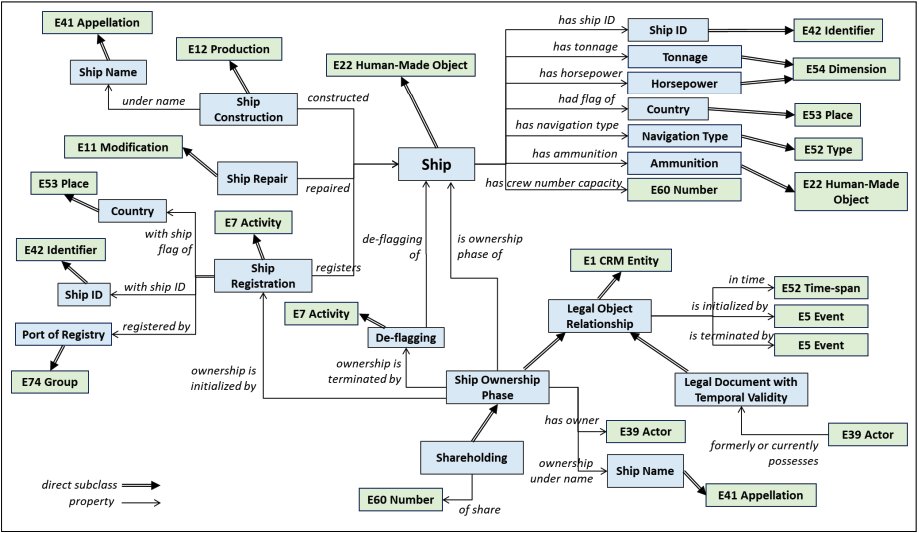
#### Οντολογία των δεδομένων SeaLit

Όπως έχει ήδη αναφερθεί τα δεδομένα SeaLit, αποτελούν ένα μοντέλο δεδομένων RDF. Η οντολογία λοιπόν που τα περιγράφει έχει στόχο τη διευκόλυνση της ανταλλαγής, ενσωμάτωσης και διαμεσολάβησης ανόμοιων πληροφοριών που σχετίζονται με τη ναυτική ιστορία. Έτσι προσφέρει σημασιολογικούς ορισμούς για τον συνδυασμό διαφορετικών πηγών των δεδομένων. Με αυτό τον τρόπο, δημιουργείται μια κοινή γλώσσα για όσους ειδικεύονται στον τομέα, επιτυγχάνοντας μια συνεργασία μεταξύ τους.

Η οντολογία χρησιμοποιεί μια επέκταση του προτύπου CIDOC Conceptual Reference Model (ISO 21127:2014) version 7.2.1, ως μια γενική οντολογία της ανθρώπινης δραστηριότητας, πραγμάτων και γεγονότων που συμβαίνουν στο χώρο και στο χρόνο. Η ανάπτυξης της, ακολούθησε μια από κάτω προς τα πάνω διαδικασία από πρωτογενή δεδομένα που συλλέχθηκαν σε αυτό το έργο.



Βασικές έννοιες που χρησιμοποιεί το CIDOC CRM, συνδέονται όπως φαίνονται στο παρακάτω σχήμα.



Με την χρήση των παραπάνω εννοιών, γίνεται πιο εμφανές σε ένα URI αυτό που θα μπορούσε να αντιπροσωπεύει, όπως είναι μια οντότητα ή μια ιδιότητα μιας οντότητας.

*(Για καλύτερη κατανόηση, μπορείτε να δείτε το* [*Παράδειγμα αρχείου διαμόρφωσης*](#_Παράδειγμα_αρχείου_διαμόρφωσης)*)*

Έχοντας λοιπόν, την δομή του προτύπου CIDOC CRM, μπορούμε να ορίσουμε τα SPARQL ερωτήματα που.

Στην περίπτωση μας, η μελέτη του μοντέλου που χρησιμοποιούν τα δεδομένα είναι σημαντική για να μπορούμε να ορίσουμε τα ερωτήματα στο configuration file, τα οποία θα μας δώσουν τα αποτελέσματα που χρειαζόμαστε.

Η χρήση του CIDOC-CRM για το SeaLit project, παρέχει ένα δομημένο και ξεκάθαρο τρόπο περιγραφής και αναζήτησης των δεδομένων του έργου. Εφόσον, γίνει κατανοητή η οντολογία και του τρόπου που περιγράφονται οι οντότητες και οι σχέσεις τους, επιτυγχάνεται η βέλτιστη ενοποίηση και ανάκτηση των δεδομένων, επιτρέποντας την ανάλυση τους.

# **Εφαρμογή Πλοήγησης σε RDF δεδομένα**

## **Αρχιτεκτονική και Βασικές λειτουργίες συστήματος**

### Ανάγνωση RDF αρχείων και Φόρτωση γράφου στη μνήμη

Η ανάγνωση RDF αρχείων και φόρτωση γράφου στην μνήμη, επιτρέπει την αποτελεσματική επεξεργασία και αναζήτηση δεδομένων.

Με την χρήση του framework Apache Jena, γίνεται η αποθήκευση και διαχείριση των RDF δεδομένων. Αυτό που γίνεται από το σύστημα είναι η ανάλυση των αρχείων RDF για εξαγωγή των δεδομένων και ύστερα η φόρτωση τους στην μνήμη για γρήγορη πρόσβαση.

Τα δεδομένα που χρησιμοποιήθηκαν στην παρούσα εφαρμογή αφορούσαν πληροφορίες από την Ναυτική Ιστορία και έχουν εξαχθεί από λίστες πληρωμάτων.

Συγκεκριμένα υπάρχει ένα έγγραφο τεσσάρων σελίδων που παραδίδεται από τις λιμενικές αρχές που βρίσκονται στο λιμάνι νηολόγησης ενός πλοίου και αφορούν καταλόγους πληρωμάτων που φυλάσσονται από το κράτος. Αυτά τα έγγραφα, εκδόθηκαν κατά την αναχώρηση του πλοίου από το λιμάνι μιας χώρας, τη στιγμή της εγγραφής του πληρώματος, και κάλυπταν περιόδους συνεχούς ναυσιπλοΐας (μέγιστης περιόδου δύο χρόνων).

Στο εσωτερικό, οι λίστες πληρωμάτων περιέχουν πληροφορίες:

* για το πλοίο,
* τους πλοιοκτήτες και
* τα μέλη του πληρώματος,
* συμπεριλαμβανομένων προσωπικών και επαγγελματικών στοιχείων.

Στην τελευταία σελίδα υπήρχε συνήθως ένας κατάλογος των λιμανιών, αποτελούμενος από προξενικές σφραγίδες και δηλώσεις που χρειαζόταν κάθε καπετάνιος πριν φύγει από λιμάνι, ανεξάρτητα από το αν σταματούσε εκεί από ανάγκη (για υγειονομικούς και τελωνειακούς ελέγχους) ή για εμπορικές εργασίες. Παρέχουν στοιχεία ταυτότητας πλοίου:

* όνομα,
* χωρητικότητα,
* τύπος,
* τόπος και ημερομηνία κατασκευής.

Παρέχουν επίσης πολύτιμες πληροφορίες για τους ναυτικούς:

* ημερομηνία επιβίβασης,
* ημερομηνία απόρριψης,
* επώνυμο,
* όνομα,
* ημερομηνία και τόπος γέννησης,
* κατάταξη στο πλοίο,
* αύξων αριθμός και
* μισθοί.

Αυτά τα στοιχεία αποτελούσαν τα κύρια δεδομένα, από τα οποία συλλέχτηκαν πληροφορίες για την διαμόρφωση της διαδικτυακής πλατφόρμας περιήγησης και εξερεύνησης.

### **Επισκόπηση έργου και βασικών οντοτήτων**

Κατά την είσοδο στην διαδικτυακή εφαρμογή, υπάρχει:

* Μια περιγραφή του έργο για το οποίο παρουσιάζονται τα δεδομένα
* Μια επισκόπηση των κατηγοριών και του αριθμού οντοτήτων τους, για τα δεδομένα του έργου

ΛΕΙΤΟΥΡΓΙΕΣ ΠΟΥ ΥΠΟΣΤΗΡΙΖΕΙ Ο ΠΙΝΑΚΑΣ

Για την παρουσίαση των δεδομένων κάθε κατηγορίας, έχει δημιουργηθεί ένας πίνακας που εμφανίζει, ανάλογα την περίπτωση, στήλες με βασικά δεδομένα που ενδιαφέρουν τον χρήστη με την πρώτη ματιά

Ο πίνακας υποστηρίζει τις εξής λειτουργίες:

### **Σελιδοποίηση**

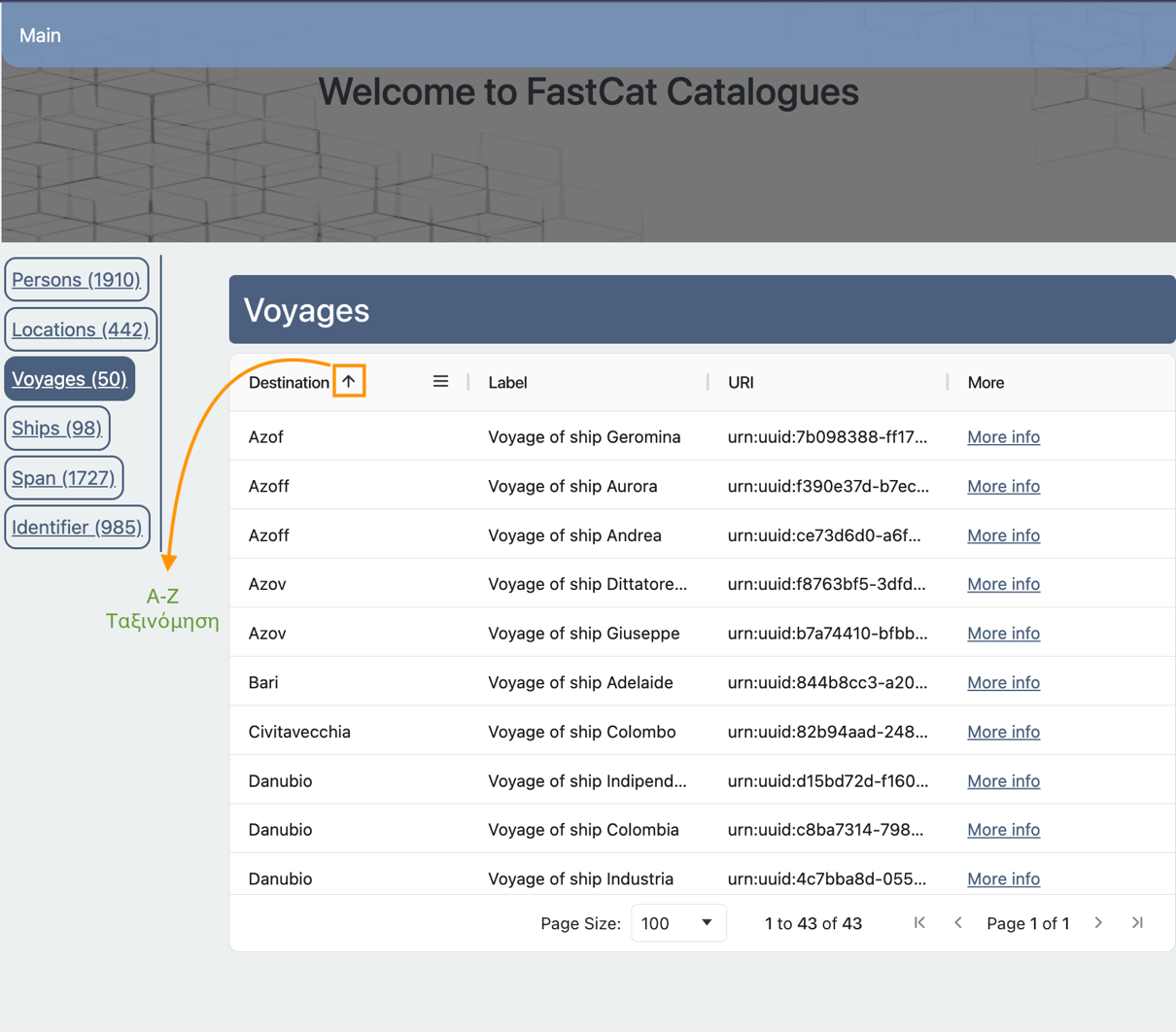
Επειδή ο όγκος των δεδομένων είναι μεγάλος, σελιδοποιεί τα αποτελέσματα, πετυχαίνοντας και καλύτερη διαχείριση τους(ανάγνωση) και ευκολότερη πλοήγηση στα αποτελέσματα.  
Επιπλέον υπάρχει η τροποποίησης του αριθμού των απεικονιζόμενων γραμμών του πίνακα.

(Γίνεται κατανοητό παρακάτω στην [Διεπαφή χρήστη](#_Διεπαφή_χρήστη_(User))

### **Ταξινόμηση**

Για κάθε στήλη του πίνακα, έχει βρεθεί ο τύπος των περιεχομένων που περιλαμβάνει.  
Σε εκείνες που περιλαμβάνουν:

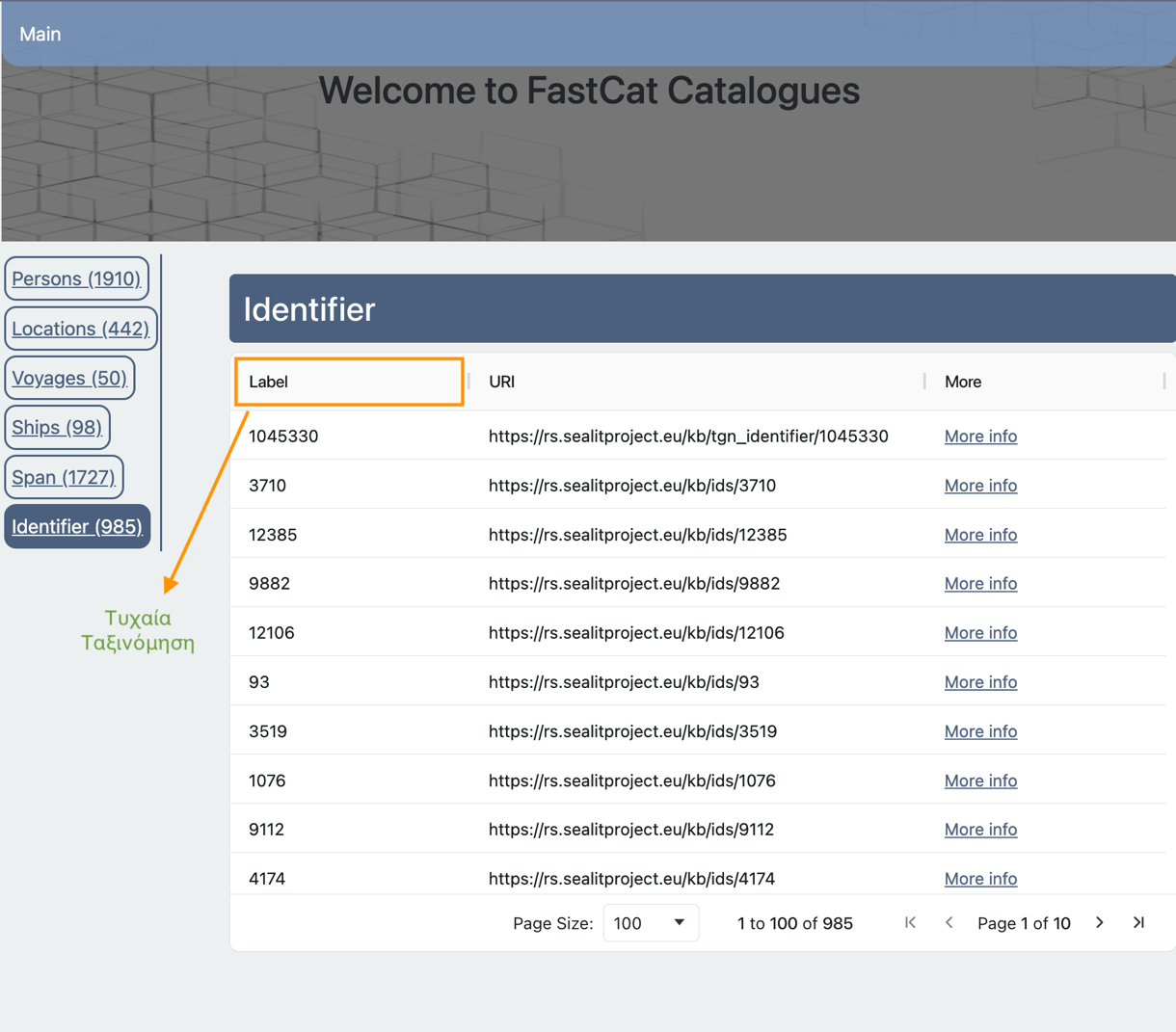
* **Strings(χαρακτήρες):**
  + Default αναπαράσταση: τυχαία(συγκεκριμένα είναι με την σειρά που βρέθηκαν στα αρχεία που αναζητήθηκαν)
  + Γίνεται αλφαβητική ταξινόμηση (A🡪Z or Z🡪A)



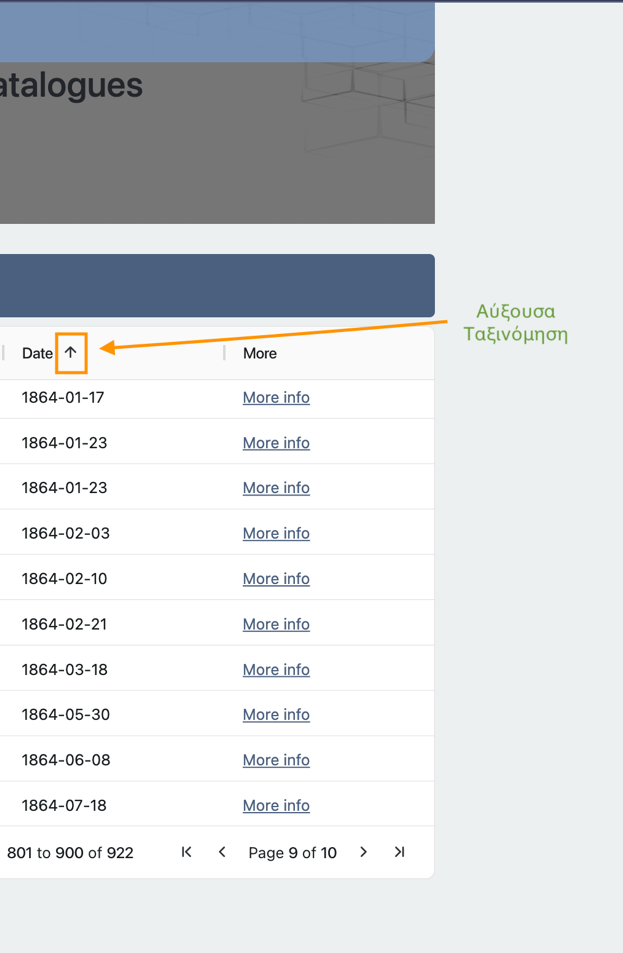
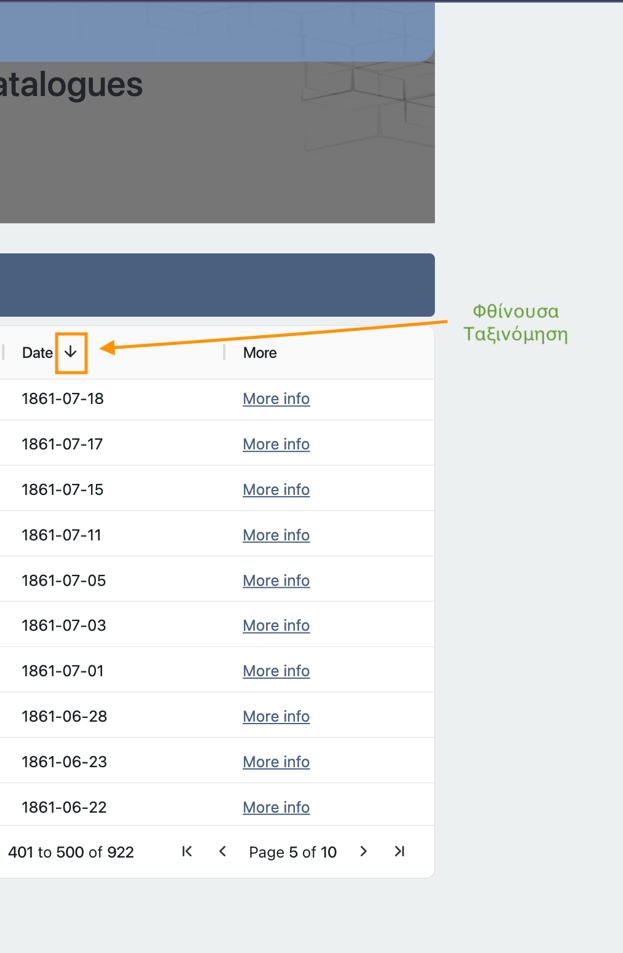
Εικόνα 1: Ταξινόμηση Α-Ζ

Εικόνα 2: Ταξινόμηση Ζ-Α

* **Αριθμούς:**
  + Default αναπαράσταση: τυχαία(συγκεκριμένα είναι με την σειρά που βρέθηκαν στα αρχεία που αναζητήθηκαν)
  + Γίνεται Αύξουσα (1 🡪 9) ή Φθίνουσα (9🡪1) ταξινόμηση

****

Εικόνα 3: Τυχαία ταξινόμηση ενός column

* **Ημερομηνίες:**
  + Default αναπαράσταση: τυχαία(συγκεκριμένα είναι με την σειρά που βρέθηκαν στα αρχεία που αναζητήθηκαν)
  + Γίνεται Αύξουσα ή Φθίνουσα ημερολογιακή ταξινόμηση

Εικόνα 4: Αύξουσα Ταξινόμηση

Εικόνα 5: Φθίνουσα Ταξινόμηση

Με κλικ πάνω στο column που μας ενδιαφέρει γίνεται μια από τις τρεις ταξινομήσεις κάθε φορά.

*(\*Για τις απαιτήσεις των παραδειγμάτων, προστέθηκαν στην εφαρμογή δυο επιπλέον κατηγορίες που περιλαμβάνουν ημερομηνίες και αριθμητικά δεδομένα.)*

### **Φίλτρα**

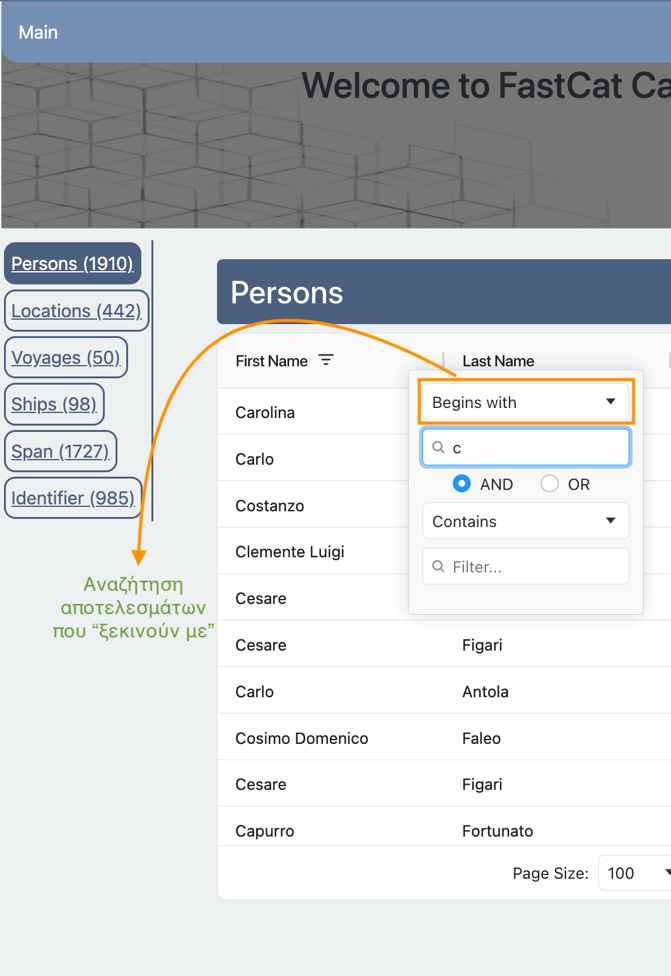
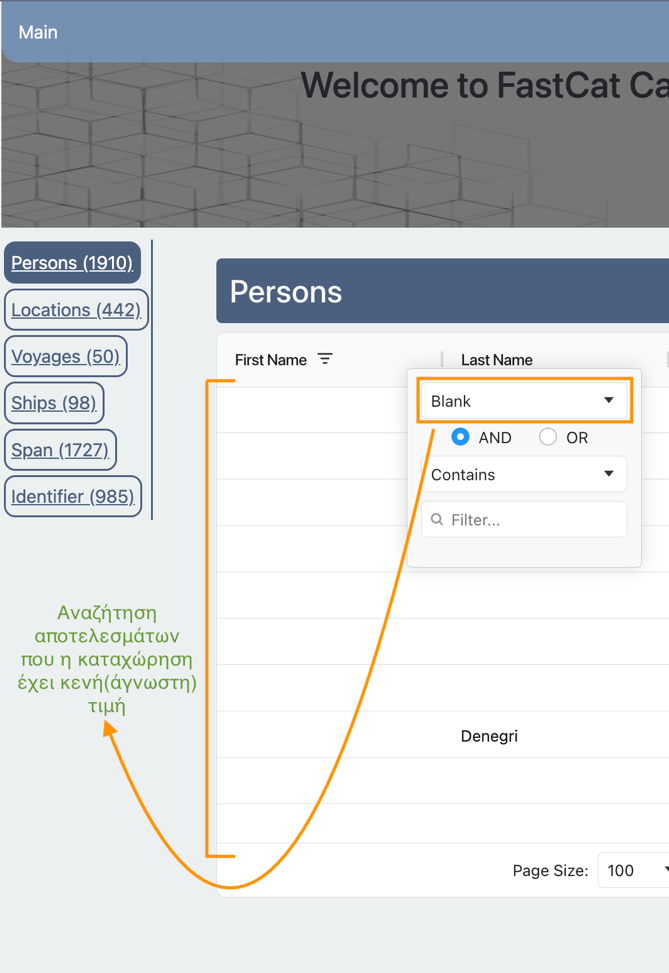
Στον πίνακα όπου εμφανίζονται τα δεδομένα κάθε κατηγορίας, για κάθε στήλη(ετικέτα-label) υπάρχουν φίλτρα.

Αυτά τα φίλτρα είναι από τις πιο χρήσιμες λειτουργίες της εφαρμογής, καθώς οι χρήστες μπορούν με άμεσο και γρήγορο τρόπο να βρουν οποιαδήποτε πληροφορία χρειάζεται.

Το φίλτρο που θα έχει κάθε στήλη προκύπτει από το περιεχόμενο τους.

Για κάθε περίπτωση έχει δημιουργηθεί ένα συγκεκριμένο template, όπου:

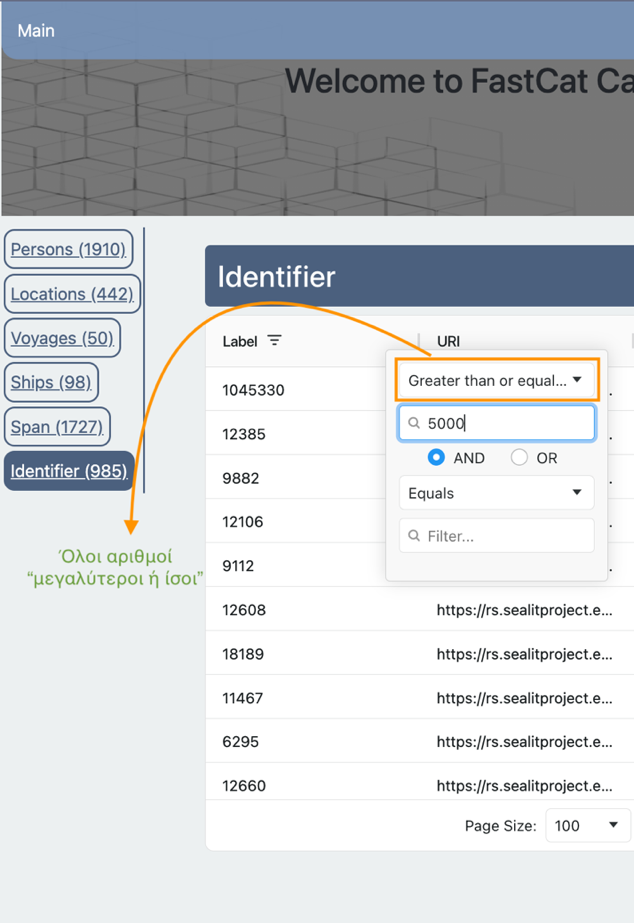
* Αν περιλαμβάνουν **κείμενο**, τότε ο χρήστης μπορεί με αλφαβητικούς χαρακτήρες να δει:
  + Contains: Αν υπάρχει ή όχι, κάτι που να ταιριάζει με αυτό που πληκτρολογεί
  + Εquals: Αν υπάρχει ή όχι, μια ολόκληρη η λέξη που αναζητεί
  + Begins/Ends with: Αν ξεκινάει ή τελειώνει με τους χαρακτήρες που έβαλε
  + Blank: Τις καταχωρήσεις της στήλης που είναι ή δεν είναι κενές



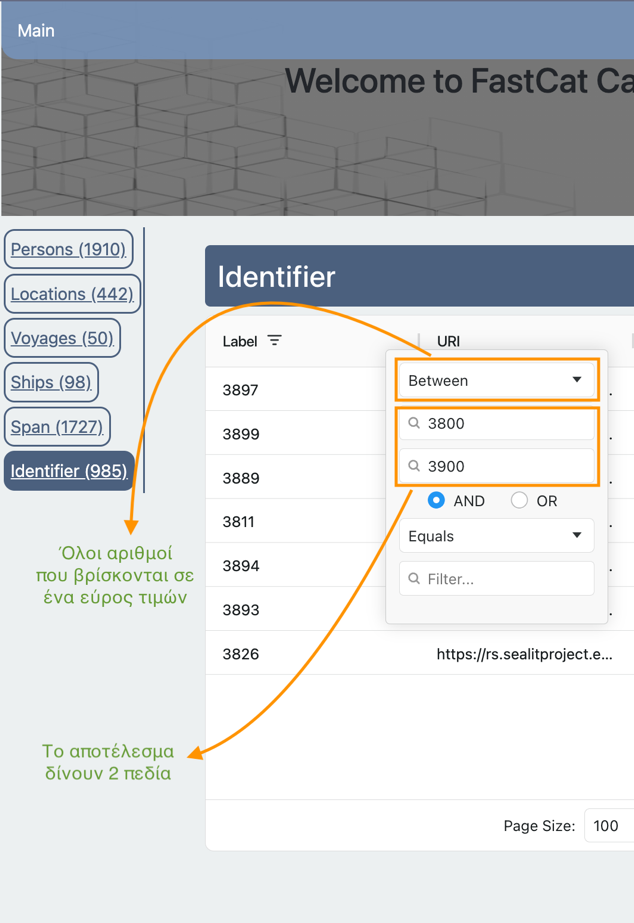
Εικόνα 6: Εμφάνιση αποτελεσμάτων με κενή τιμή καταχώρησης

Εικόνα 7:Αναζήτηση αποτελεσμάτων που "ξεκινούν με"

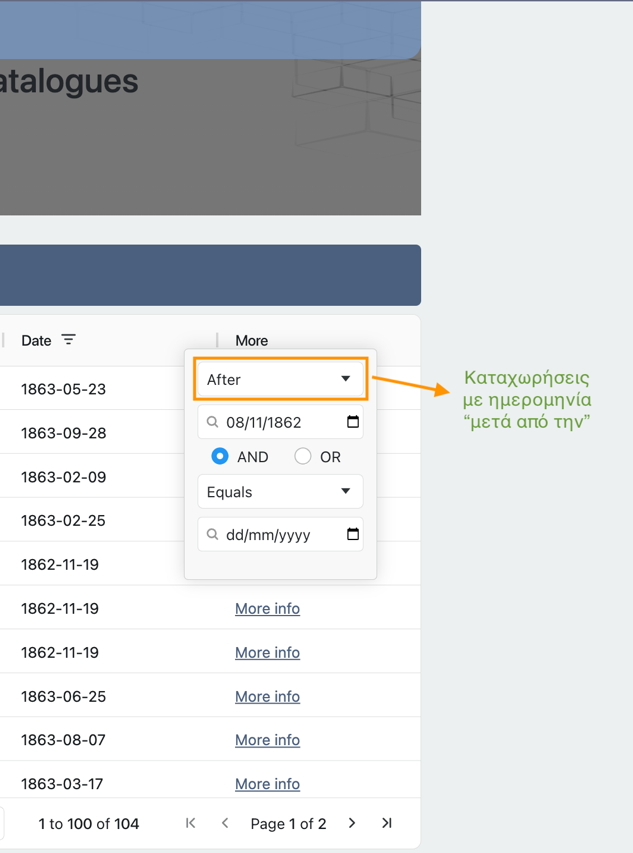
*(\*Δεν υπάρχει case-sensitivity στο search bar και ο χρήστης θα πάρει αποτέλεσμα είτε γράψει με κεφαλαία, είτε με μικρά γράμματα)*

* Αν περιλαμβάνουν **ψηφία**, τότε ο χρήστης μπορεί να ψάξει με βάση κάποιο αριθμητικό χαρακτήρα να δει:
  + Εquals: Αν κάτι είναι ίσο ή όχι, με τον αριθμό που πληκτρολόγησε
  + Greater/Less than: Αριθμούς μεγαλύτερους ή μικρότερους από αυτόν που καταχώρησε
  + Greater/Less than or equal to: Αριθμούς μεγαλύτερους/μικρότερους ή ίσους από εκείνο που καταχώρησε
  + Between: Αριθμούς μεταξύ των αριθμών που έβαλε(εδώ εμφανίζονται 2 πεδία που θα εισάγει τιμή ο χρήστης)
  + Blank: Τις καταχωρήσεις της στήλης που είναι ή δεν είναι κενές

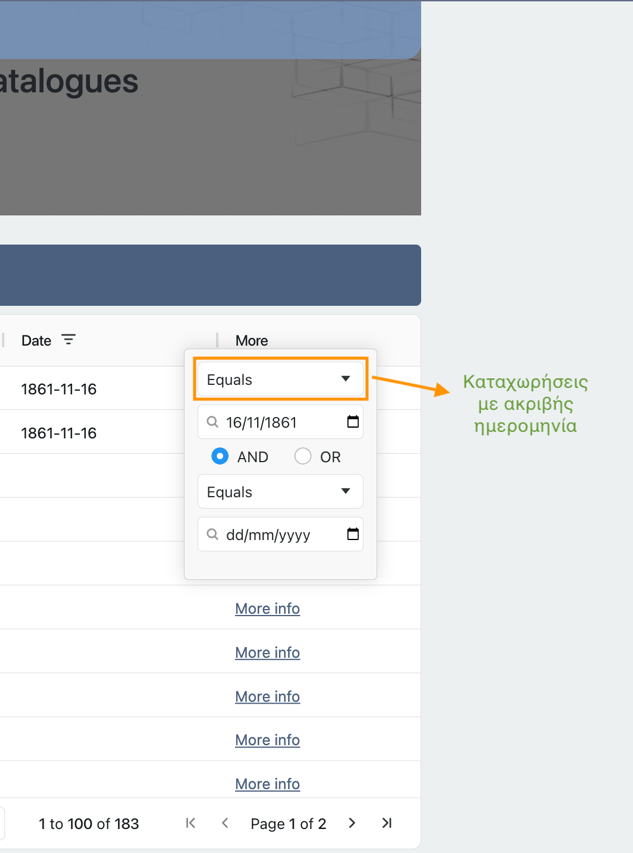
Εικόνα 8: Αναζήτηση αριθμούς "μεγαλύτερων ή ίσων από"



Εικόνα 9: Αναζήτηση αριθμών ενός εύρους τιμών

* Αν περιλαμβάνουν **ημερομηνίες**, τότε ο χρήστης μπορεί να δει:
  + Equals: Aν περιλαμβάνετε ή όχι, μια ακριβής ημερομηνία ή όχι
  + Before/After: Τις ημερομηνίες πριν ή μετά από εκείνη που καταχώρησε
  + Βetween: Τις ημερομηνίες μεταξύ ενός διαστήματος(εδώ εμφανίζονται 2 πεδία που θα εισάγει τιμή ο χρήστης)
  + Blank: Τις καταχωρήσεις της στήλης που είναι ή δεν είναι κενές

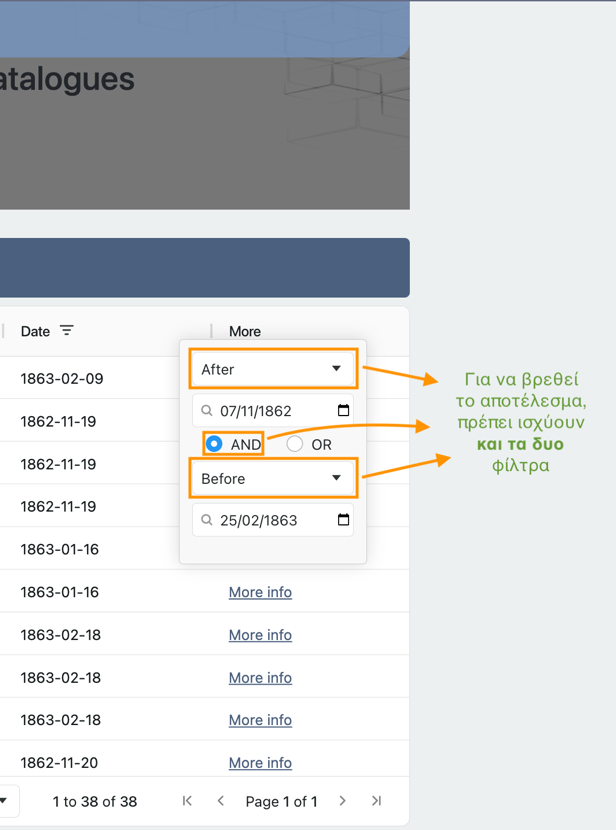
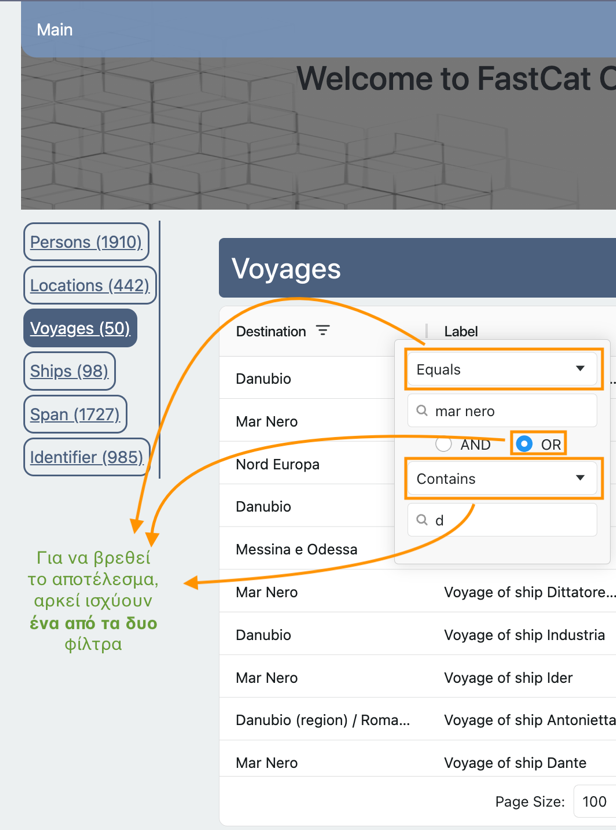
Εικόνα 10: Αναζήτηση ημερομηνιών "μετά από την"



Εικόνα 11: Αναζήτηση ακριβής ημερομηνίας

Μεταξύ των διαφόρων φίλτρων, είναι δυνατόν το φιλτράρισμα να γίνεται ταυτοχρόνως με 2 παραμέτρους, τις οποίες μπορεί ο χρήστης να διαλέξει αν το αποτέλεσμα θα το πάρει με την χρήση:

* + ΑND, όπου πρέπει και τα δυο φίλτρα να είναι αληθή
  + OR, όπου αρκεί το ένα από τα δυο φίλτρα να είναι αληθή.



Εικόνα 12: Ταξινόμηση που αρκεί να ισχύει **"ένα από τα δύο"** φίλτρα

Εικόνα 13: Ταξινόμηση που πρέπει να ισχύουν **"και τα δυο"** φίλτρα

### 

### **Διαγράμματα(Charts)**

Τα διαγράμματα προκύπτουν από τα δεδομένα κάθε στήλης και σχετίζονται με τον αριθμό εμφάνισης των δεδομένων που περιέχουν.  
Ανάλογα την περίπτωση μπορούν να περιέχουν σημαντική πληροφορία στον χρήστη, ανάλογα με την στήλη κάποιας οντότητας στην οποία βρίσκετε εκείνη την στιγμή.

ΦΩΤΟ

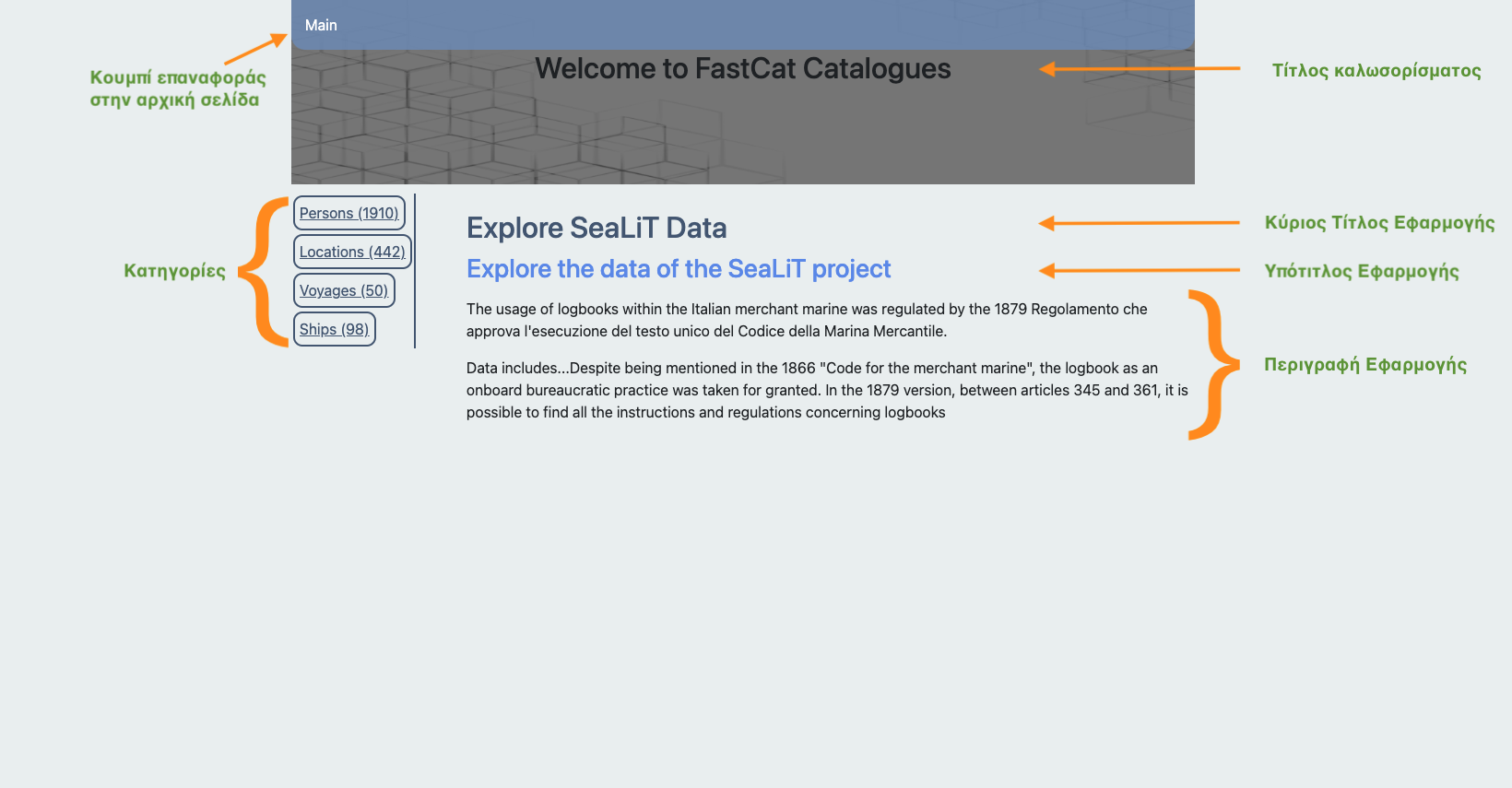
### **Προβολή λεπτομερειών μιας καταχώρησης**

### Όταν ο χρήστης ενδιαφέρεται να μάθει περαιτέρω πληροφορίες για μια συγκεκριμένη καταχώρηση, μπορεί να πατήσει το link που βρίσκεται στην τελευταία στήλη δεξιά του πίνακα. Το σύστημα εκεί του παρουσιάζει μια σελίδα με περισσότερες λεπτομέρειες για εκείνη την καταχώρηση, οι οποίες εξαρτώνται ανάλογα με την κατηγορία.

### ΦΩΤΟ

## Διεπαφή χρήστη (User Interface)

Κατά την είσοδο στην εφαρμογή βλέπουμε την αρχική σελίδα:

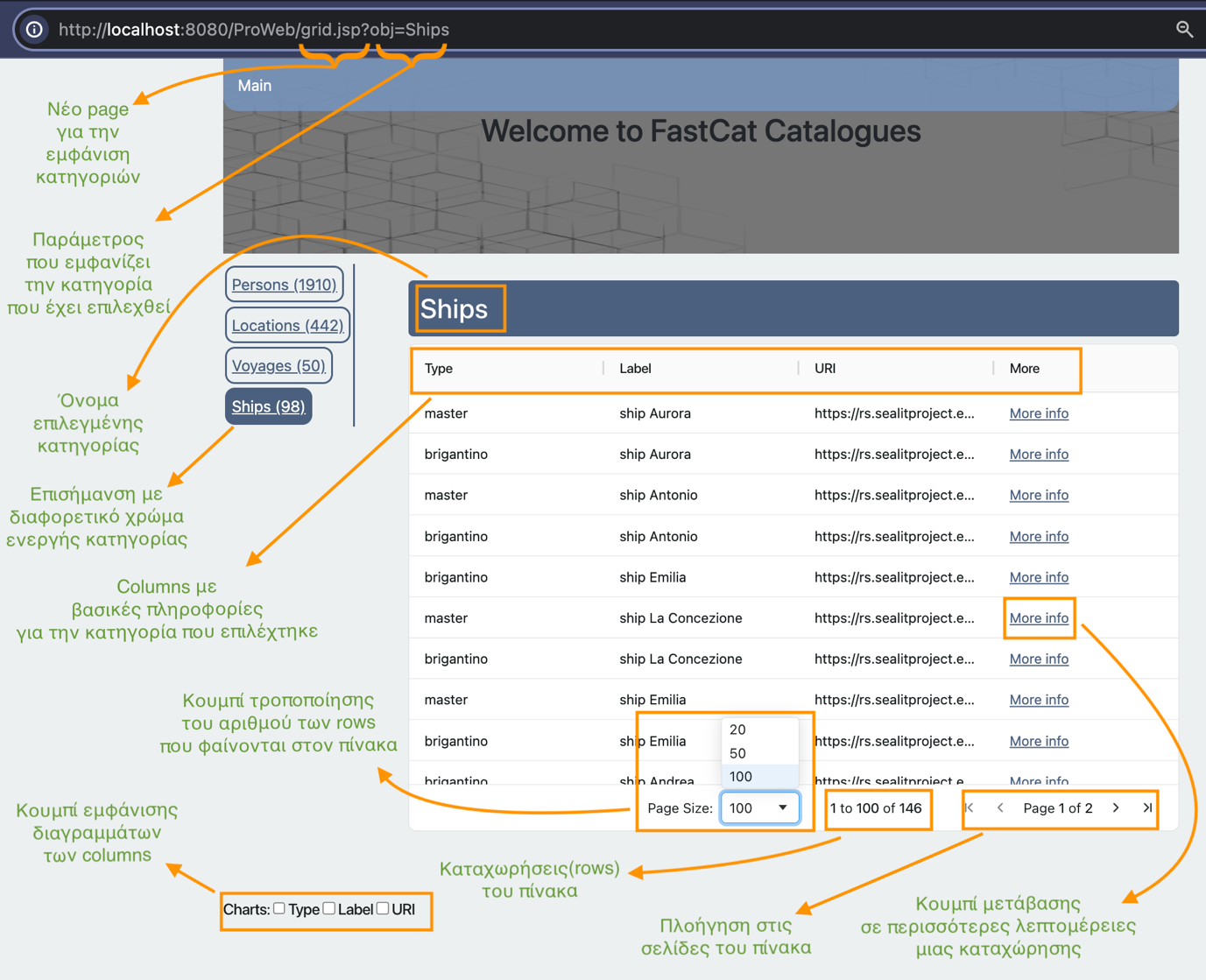


Εικόνα 14: Αρχική σελίδα εφαρμογής

Υπάρχουν:

* Κουμπί επαναφορά στην αρχική σελίδα
* Κατηγορίες εφαρμογής
* Τίτλος Καλωσορίσματος
* Κύριος Τίτλος Εφαρμογής
* Υπότιτλος Εφαρμογής
* Περιγραφή Εφαρμογής

Αφού επιλεχθεί μια κατηγορία εμφανίζεται η παρακάτω σελίδα:



Εικόνα 15: Σελίδα εμφάνισης μιας κατηγορίας

Υπάρχουν:

* Στο URL:
  + - grid.jsp: Το νέο page που δημιουργήθηκε για την αναπαράσταση των δεδομένων κάθε κατηγορίας
    - obj: Αποτελεί την παράμετρο που αναδεικνύει το όνομα της κατηγορίας που έχει επιλεχθεί

Εδώ θα είναι μια από τις:

* + - * Persons
      * Locations
      * Voyages
      * Ships
* Δεξιά από τις κατηγορίες υπάρχει:
  + - Όνομα επιλεγμένης κατηγορίας
    - Πίνακας δεδομένων επιλεγμένης κατηγορίας:
      * Στην πρώτη γραμμή εμφανίζονται τα ονόματα των βασικών columns  
        (Τα ονόματα προκύπτουν από αυτά που έχουμε ορίσει μέσω των queries στο configuration file)
      * Στο κάτω μέρος υπάρχουν οι λειτουργίες του:
        1. Δυνατότητα αλλαγής του μεγέθους των γραμμών της σελίδας
        2. Αριθμός γραμμών στην παρούσα σελίδα και των υπολειπόμενων
        3. Σελιδοποίηση
      * Στην τελευταία στήλη του πίνακα υπάρχει κουμπί μετάβασης σε περισσότερες λεπτομέρειες για την συγκεκριμένη καταχώρηση

Στην κάτω μεριά της προηγούμενης εικόνας υπάρχουν τα διαγράμματα(Charts)



Εικόνα 16: Εμφάνιση διαγραμμάτων columns μιας κατηγορίας

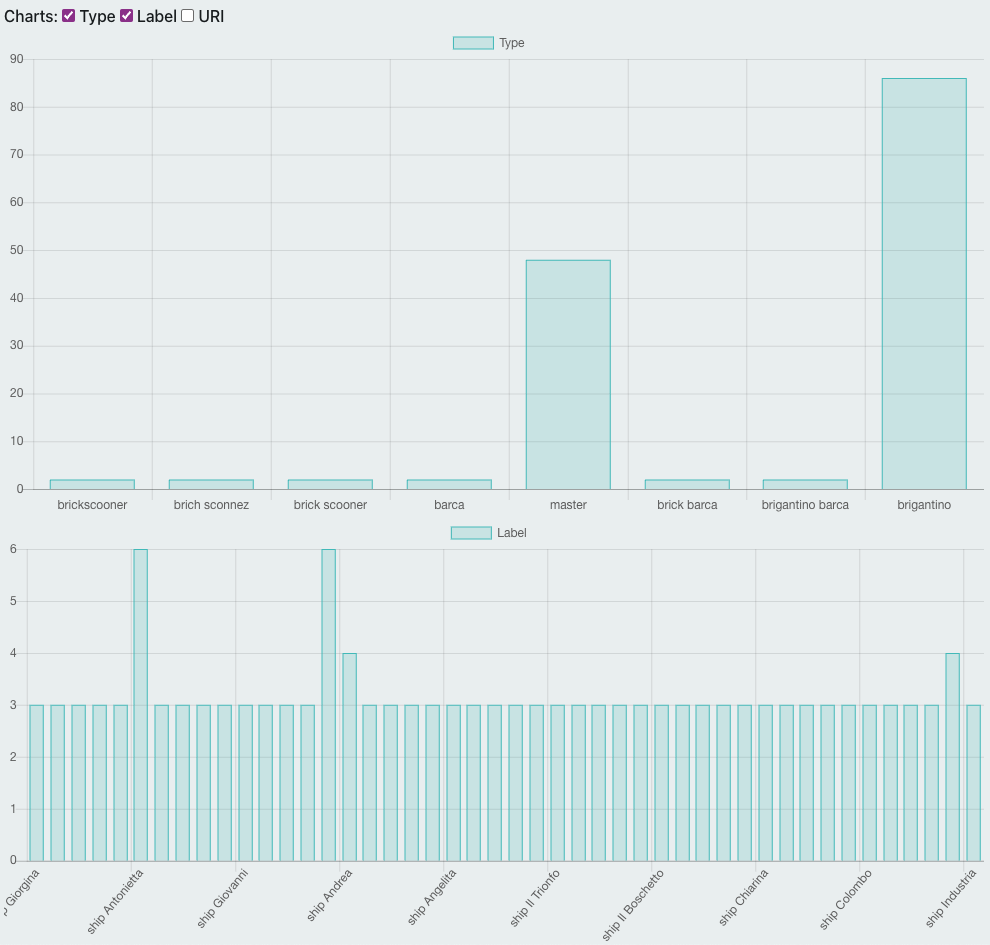
Σε αυτά ο χρήστης μπορεί να επιλέξει να δει το διάγραμμα που επιθυμεί.

Σε καθένα απ’ αυτά υπάρχουν οι εξής πληροφορίες:

* Στον κάθετο άξονα(στα αριστερά), υπάρχει ο αριθμός εμφάνισης των στοιχείων της επιλεγμένης στήλης του πίνακα.  
  (Το εύρος των τιμών προκύπτει από τα δεδομένα)
* Ο οριζόντιος άξονας (στην κάτω μεριά), δείχνει τα στοιχεία του επιλεγμένου column
* Όταν o κέρσορας πηγαίνει σε μια ομάδα στοιχείων του διαγράμματος, εμφανίζονται:
  + Το όνομα της ομάδας
  + Η ποσότητα εμφάνισης τους στον πίνακα

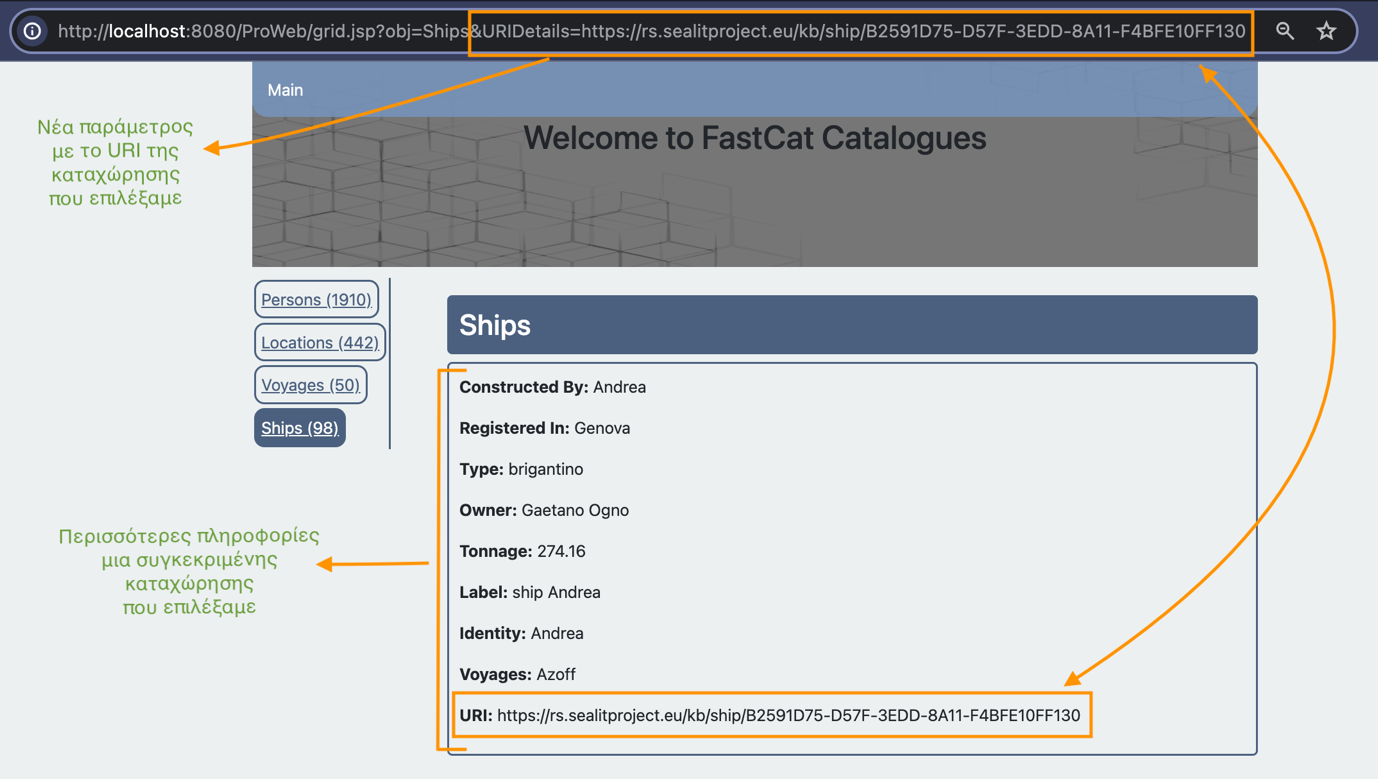
Η επιλογή του παραπάνω διαγράμματος έγινε επειδή αποτελεί ένα πολύ καλό παράδειγμα μιας και δίνει χρήσιμα αποτελέσματα.

Επιπλέον μπορούν να εμφανιστούν και πολλαπλά διαγράμματα την ίδια στιγμή



Εικόνα 17: Δυνατότητα εμφάνισης πολλαπλών διαγραμμάτων την ίδια στιγμή

Όταν πατήσουμε το “More info” link ενός record(row) στο δεξιό column του πίνακα κάθε κατηγορίας, μεταφερόμαστε σε νέα σελίδα του ίδιου jsp page, χρησιμοποιώντας την παράμετρο “URLDetails”, με (μοναδική) τιμή το URI της καταχώρησης που επιλέχτηκε.

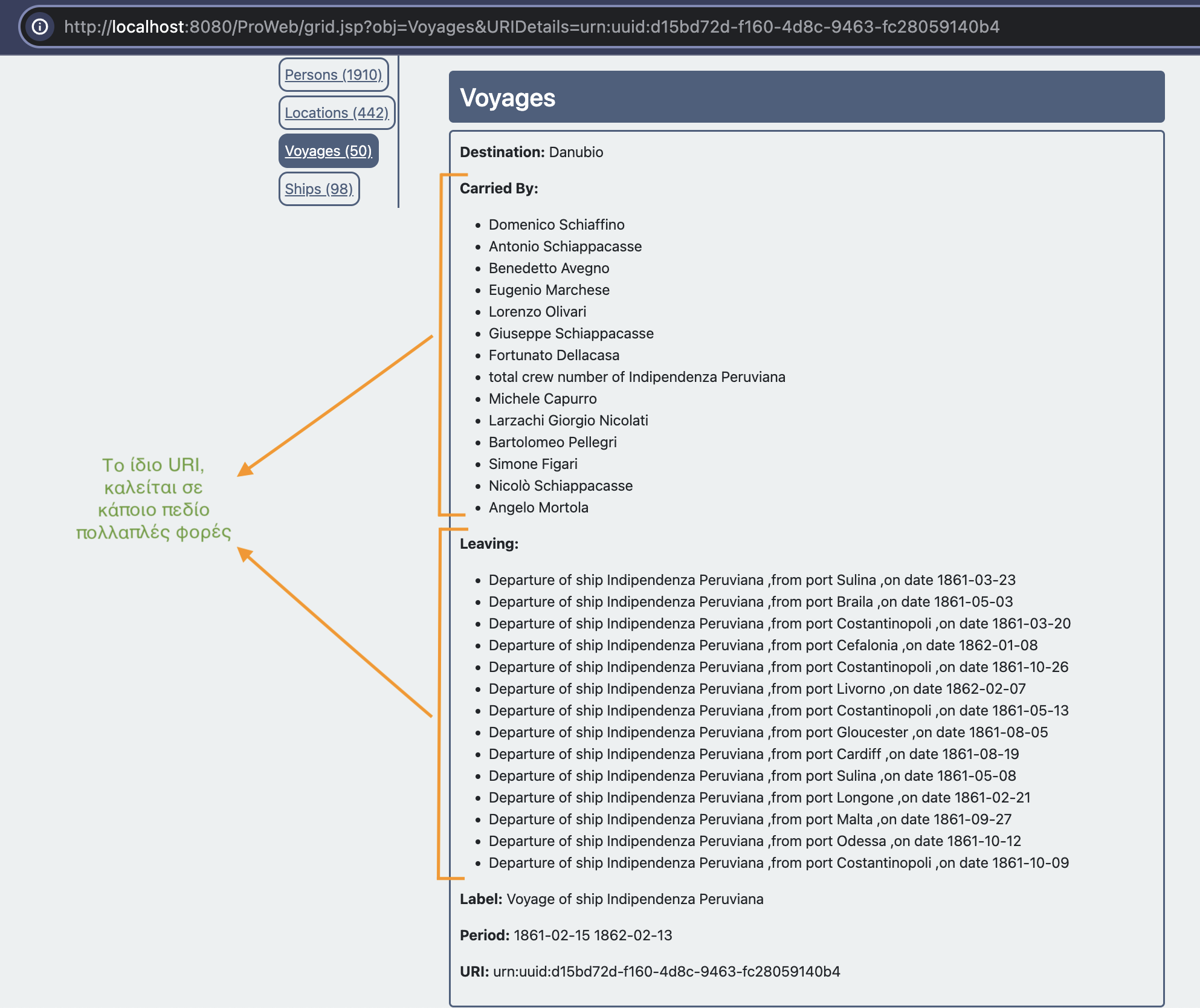


Εικόνα 18: Σελίδα με περισσότερες πληροφορίες μιας καταχώρησης

Αναφέρθηκε προηγουμένως ότι το URI κάθε καταχώρησης είναι μοναδικό. Ωστόσο μπορεί το ίδιο URI να καλείται πολλαπλές φορές σε ένα άλλο πεδίο.  
Όπως φαίνεται στο παρακάτω παράδειγμα, το Voyage του Ship: Indipendenza Peruviana

με URI: urn:uuid:d15bd72d-f160-4d8c-9463-fc28059140b4, ήταν από διαφορετικά λιμάνια

και πολλά άτομα είχαν κάνει αυτή το ταξίδι.



Εικόνα 19: Ένα URI, μπορεί να κληθεί από ένα πεδίο πολλαπλές φορές

Τεχνολογίες (, βιβλιοθήκες java, βιβλιοθήκες javascript,

### Γλώσσα Προγραμματισμού

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε είναι η Java.

Για το λόγο ότι χρειάστηκαν αρκετά κομμάτια html, ήταν καλύτερο να χρησιμοποιηθούν αρχεία jsp( Java server page), αντί για servlets.

Το jsp είναι μια server side programming τεχνολογία που επιτρέπει τη δημιουργία δυναμικής, ανεξάρτητης πλατφόρμας μεθόδου για τη δημιουργία Web apps. Με την χρήση του JSP υπάρχει δυνατότητα πρόσβασης σε όλο το Java API, με πρόσβαση στην βάση δεδομένων

### Βιβλιοθήκες που χρησιμοποιήθηκαν:

Για την υλοποίηση κάποιων λειτουργειών της εφαρμογής, χρησιμοποιήθηκαν τόσο βιβλιοθήκες Java, όσο βιβλιοθήκες JavaScript

Java Libraries

H εφαρμογή αξιοποιεί βιβλιοθήκες για την διαχείριση δεδομένων RDF, ερωτημάτων SPARQL και εκείνες που αφορούν:

* Χειρισμούς εισόδου-εξόδου
* Εισαγωγής Αρχείων
* Υποστήριξης ArrayList, Map, HashMap
* Διαχείρισης Σφαλμάτων
* Ανάλυσης και Εκτέλεσης Ερωτημάτων

#### AG Grid:

Για την δημιουργία του πίνακα, χρησιμοποιήθηκε η παρούσα βιβλιοθήκη μιας και οι λειτουργίες που παρέχει είναι πολυάριθμες όπως φιλτράρισμα, ταξινόμηση, σελιδοποίηση

Ο πίνακας οριοθετεί με κατάλληλο τρόπο τα columns κάθε κατηγορίας  
(Τα columns προκύπτουν με βάση τα queries που υπάρχουν στο configuration file)

Με τα δεδομένα που χρησιμοποιήθηκαν, αφού γίνει η επιλογή μιας οντότητας θα εμφανιστούν:

* Οι βασικές πληροφορίες για κάθε καταχώρηση, το οποίο προκύπτει από ένα SPARQL Query
* Επιπλέον στήλη που παραπέμπει σε σελίδα με περισσότερες πληροφορίες για κάθε καταχώρηση

Στο παρακάτω παράδειγμα εμφανίζεται αυτό στο οποίο αναφερόμαστε:

Έστω ότι επιλέγουμε την κατηγορία Ships -> εμφανίζονται όλες οι καταχωρήσεις που έχουμε για αυτή την κατηγορία με κάποιες βασικές πληροφορίες(columns). Στην προκειμένη περίπτωση είναι:

* URI
* Label
* ShipType

 -> Αφού για μια καταχώρηση(record)(row) επιλέξουμε να δούμε περισσότερες πληροφορίες πατώντας το “More Info”, σε νέα σελίδα εμφανίζονται (αν υπάρχουν)(εδώ έχει προστεθεί η δυνατότητα OPTIONAL, σε περιπτώσεις που για κάποιο πεδίο δεν είναι γνωστή αυτή η πληροφορία) περισσότερα πεδία. Στην περίπτωση του παραδείγματος είναι:

* Identity
* Registered In
* Owner
* Tonnage
* Constructed By

#### Chart JS

Για την δημιουργία των charts, επιλέχτηκε η βιβλιοθήκη Chart JS. Με την χρήση αυτής της βιβλιοθήκης, δίνεται η δυνατότητα με εύκολο και κατανοητό τρόπο προς τον χρήση, η οπτικοποίηση των δεδομένων για μια στήλη της κατηγορίας που επέλεξε να εμφανίσει.

# Μοντέλο Διαμόρφωσης(Configuration model)

Στα αρχεία που έχουν δημιουργηθεί στο παρόν project, υπάρχει εκείνο που αφορά την διαμόρφωση των ρυθμίσεων  
(configuration properties file : *config.properties*)

## Ιδιότητες αρχείου διαμόρφωσης

Σε αυτό μπορούμε να οριστεί (σε ένα preprocessing βήμα) διάφορες παραμέτροι του συστήματος, καθώς και λεπτομέρειες που αφορούν τα δεδομένα, όπως τις κατηγορίες οντοτήτων που μας ενδιαφέρει να εμφανίζονται στην εφαρμογή και τις σχέσεις που μπορεί να έχουν οι οντότητες μεταξύ τους.

Αποτελεί το βασικό αρχείο, η βάση δεδομένων, που χρησιμοποιεί η εφαρμογή όπου:  
- από την μια, με τα τωρινά δεδομένα, υπάρχει το ενδεχόμενο-δυνατότητα να δεχτεί αλλαγές με την πάροδο του χρόνου

- από την άλλη, αν αλλάξουν τα δεδομένα, τότε αλλάζουν κι αυτά που παρουσιάζονται στην πλατφόρμα.

Σε κάθε περίπτωση, εφόσον ακολουθήσουμε την ίδια δομή του αρχείου και η αλλαγή γίνει προσεκτικά και στοχευμένα, η πλατφόρμα θα παρουσιάσει οτιδήποτε περιέχει εκείνο το αρχείο ρυθμίσεων, με επιτυχία, χωρίς την παραμικρή τροποποίηση του κώδικα του project.

## Σχεδιασμός αρχείου διαμόρφωσης

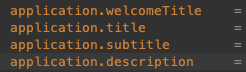
Το configuration file χρησιμοποιείται για δοκιμή, επιβεβαίωση, διόρθωση και βελτιστοποίηση των λειτουργιών της εφαρμογής που δημιουργήθηκε.

### Παράδειγμα αρχείου διαμόρφωσης

Το αρχείο, με τα δεδομένα που χρησιμοποιήθηκαν, έχει δομηθεί με τον εξής τρόπο:  
Περιλαμβάνει βασικά στοιχεία, αποκλειστικά για την εφαρμογή, τα οποία είναι:

* 1. Τίτλος Καλωσορίσματος
  2. Κύριος Τίτλος Εφαρμογής
  3. Υπότιτλος Εφαρμογής
  4. Περιγραφή Εφαρμογής

(Με τον τρόπο που φαίνονται στο πρώτο μέρος της [Πλοήγησης Εφαρμογής](#_Πλοήγηση_εφαρμογής))



Εικόνα 20: Αποκλειστικά βασικά στοιχεία της εφαρμογής

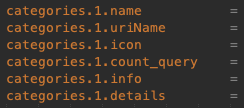
Ύστερα υπάρχει :

* το πεδίο που περιλαμβάνει τον αριθμό των κατηγοριών των οντοτήτων που υπάρχουν



Εικόνα 21: Πεδίο με αριθμό των κατηγοριών των οντοτήτων

και στην συνέχεια,

* στοιχεία που αφορούν κάθε μια κατηγορία( π.χ. categories.X.name (όπου “X”, ο αριθμός για κάθε κατηγορία)

Εικόνα 22: Στοιχεία που αφορούν κάθε κατηγορία

* + Όνομα
  + ΌνομαURI
  + Εικονίδιο
  + Αριθμό Εμφανίσεων (Αφορά τον αριθμό που ένα query βρίσκει μια κατηγορία με το όνομα που μας ενδιαφέρει)
  + Πληροφορίες (Μέσω ενός query, παίρνουμε τα βασικά columns που θέλουμε να εμφανίζονται σαν πρώτη εικόνα στον πίνακα κάθε κατηγορίας)
  + Λεπτομέρειες (Μέσω ενός query, παρέχονται περισσότερες λεπτομέρειες για το row, που ο χρήστης επέλεξε να δει)



Εικόνα 23: Παράδειγμα με στοιχεία κάθε κατηγορίας

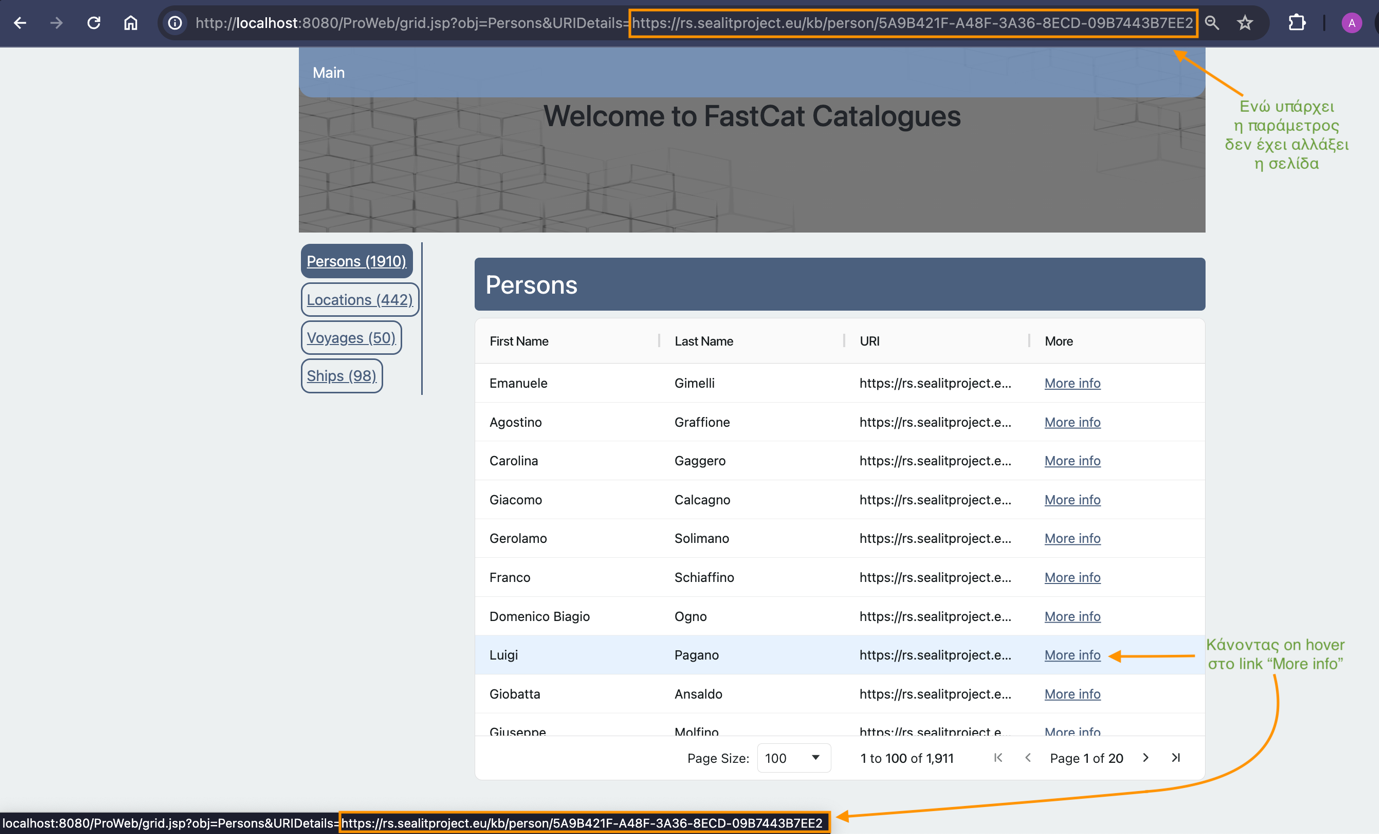
Όπως φαίνεται στην εικόνα από πάνω, υπάρχουν περιπτώσεις που η πληροφορία που χρειαζόμαστε, δεν δίνεται αμέσως, αλλά χρειάζεται να ακολουθήσουμε κάποιο path για να την πάρουμε

**Προβλήματα** που προέκυψαν:  
Υπάρχει η πιθανότητα κάποιο column του query, λόγω έλλειψης δεδομένων, να μην έχει τιμή.

**Επίλυση**:  
Σε αυτή την περίπτωση, έχει μπει σε κάθε τέτοιο column η επιπλέον παράμετρος OPTIONAL, προκειμένου να μην δημιουργείται πρόβλημα στην εμφάνιση των αποτελεσμάτων.

**Χωρίς την παράμετρο OPTIONAL:**

Όπως φαίνεται στην παρακάτω εικόνα, παίρνουμε την ίδια σελίδα με εκείνη του πίνακα, μαζί με την παράμετρο στο URL που περιλαμβάνει το URI του record(row) που επιλέχτηκε.

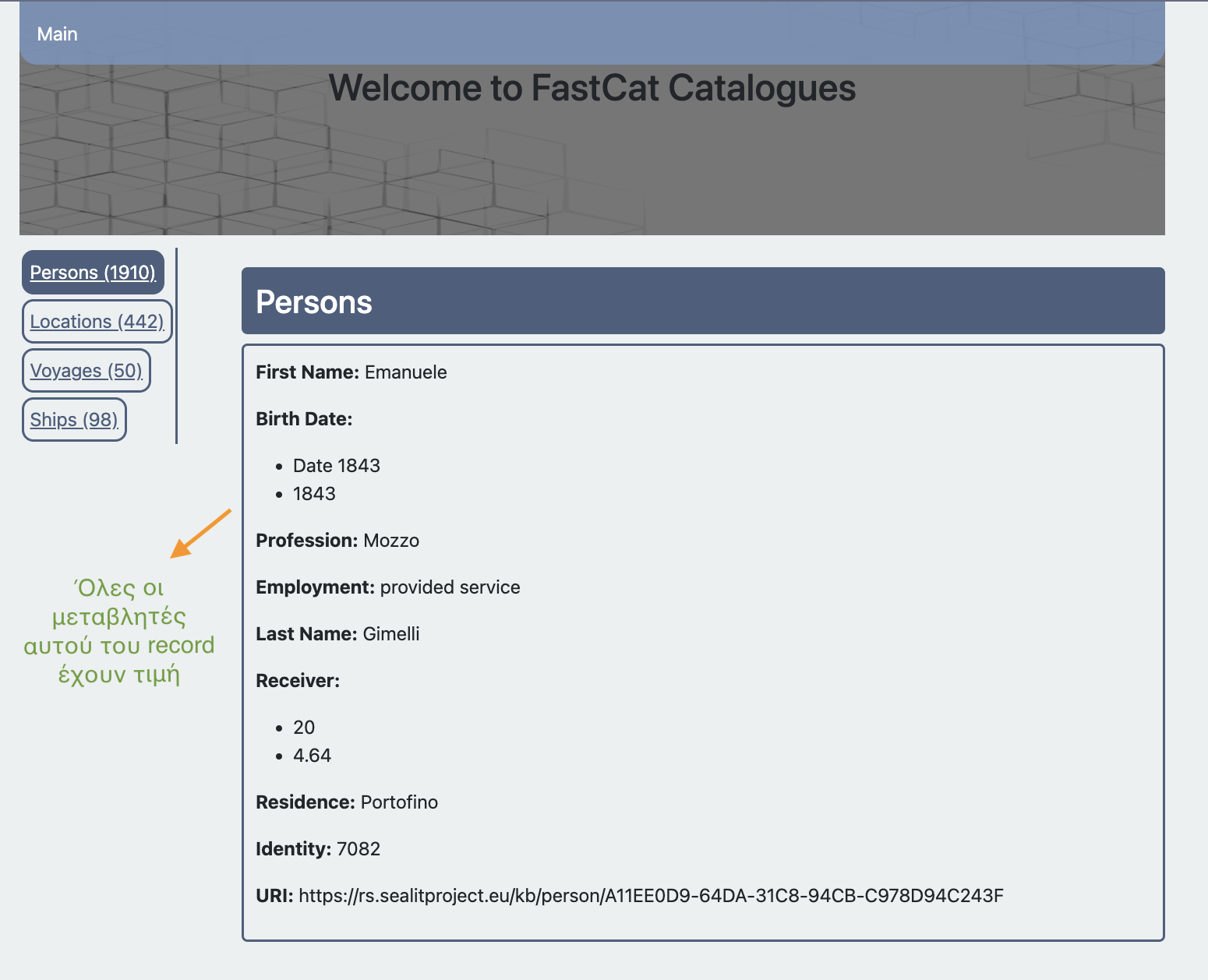


Εικόνα 24: Σελίδα χωρίς την παράμετρο ΟPTIONAL στο query που δίνει τα αποτελέσματα

**Με την χρήση της παραμέτρου OPTIONAL:**

Επιλύεται το πρόβλημα και έχουμε τα επιθυμητά αποτελέσματα.

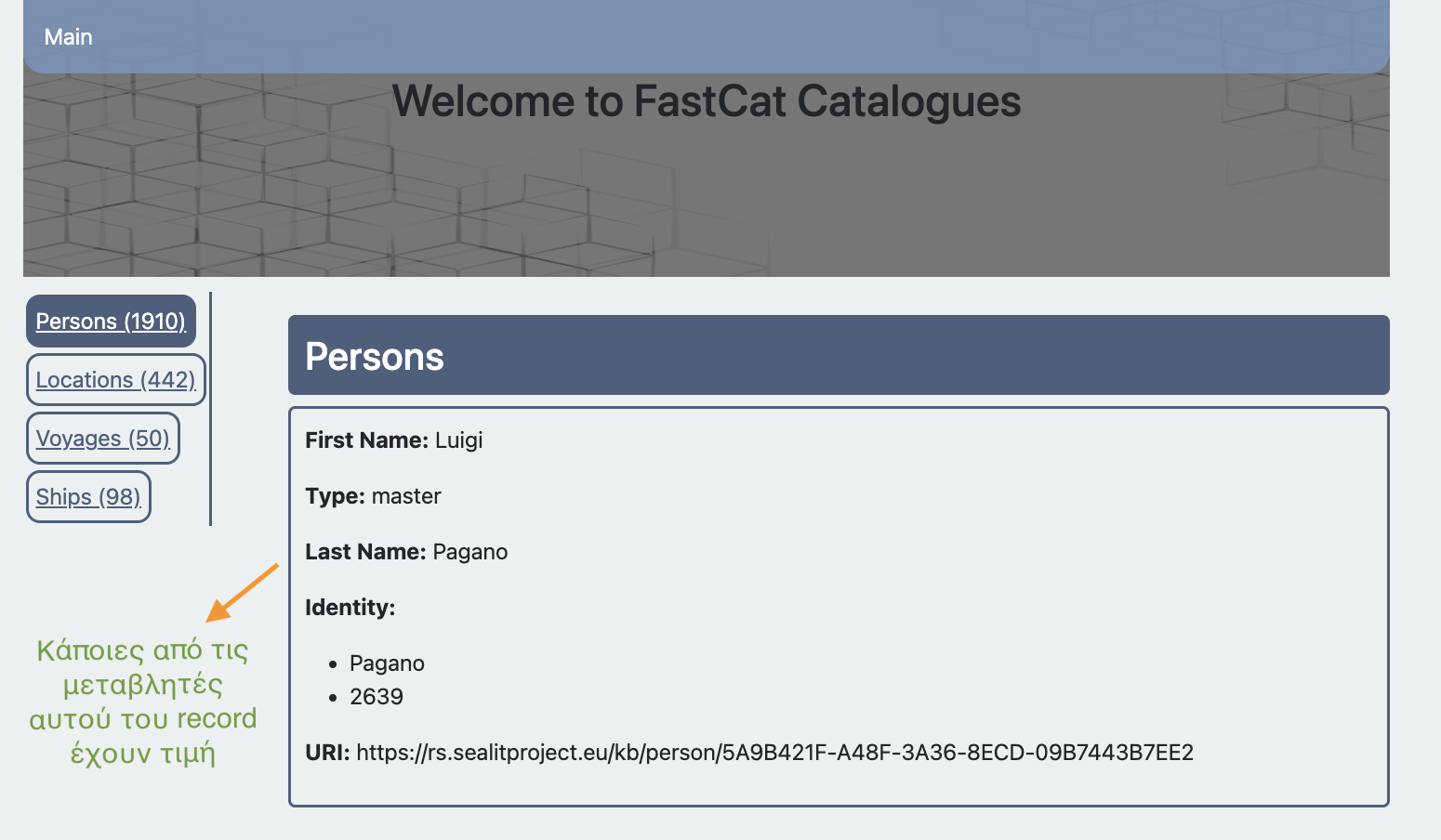
1η περίπτωση:



Εικόνα 25: Σελίδα με την παράμετρο ΟPTIONAL στο query που δίνει τα αποτελέσματα

Όπως φαίνεται στην εικόνα παραπάνω, υπάρχουν αρκετές πληροφορίες για την συγκεκριμένη καταχώρηση και όλες οι μεταβλητές έχουν τιμή.

2η περίπτωση:



Εικόνα 26: Σελίδα με περίπτωση που δεν έχουν όλες οι μεταβλητές τιμή

Από την άλλη, σε αυτή την εικόνα, πεδία όπως Birth Date, Employment, Receiver, δεν έχουν τιμή, χωρίς ωστόσο να δημιουργείται το παραμικρό πρόβλημα, αφού όλα τα υπόλοιπα (πεδία) εμφανίζονται κανονικά.

# Εγκατάσταση Συστήματος

Η εφαρμογή απαιτεί την εγκατάσταση των:

* Java Runtime Environment (η δημιουργία του project έγινε με την έκδοση 17.0.8.1)

Επίσης είναι απαραίτητο να οριστεί στον φάκελο *“resources”* , ο οποίος βρίσκεται στον main φάκελο του source folder (src/main/resources):

* Ο φάκελος με τα δεδομένα που θα τρέξει η εφαρμογή
* Το αρχείο config.properties που ορίζει συγκεκριμένα τα αντικείμενα που θα πάρω από τον από πάνω φάκελο

Με το συγκεκριμένο όγκο δεδομένων και τις τριπλέτες που τρέχει το αρχείο, η JAVA καταναλώνει Χ μνήμη στον μηχάνημα.

Αφού γίνει “Run” το project(index.jsp), εκτελείται ο Apache Tomcat (server).

Αυτό θα ανοίξει μια καρτέλα στον browser του μηχανήματος που τρέχει η εφαρμογή και θα γίνει μετάβαση στην σελίδα:

<http://localhost:8080/ProWeb>, όπου

* localhost: η IP address του δικτύου,
* 8080: το port που χρησιμοποιείται συνήθως από web servers
* ProWeb: Tίτλος της εφαρμογής που όριστηκε

όπου εμφανίζεται το main page της εφαρμογής

# Απαιτήσεις Συστήματος και Απόδοση

Το μηχάνημα στο οποίο έτρεχε η εφαρμογή, είχε τα παρακάτω χαρακτηριστικά:

|  |  |
| --- | --- |
| Λειτουργικό Σύστημα | MacOS Sonoma 14.2.1 |
| Επεξεργαστής | Apple M1 3.2GHz |
| Μνήμη | 16 GB RAM |

Η απόδοση της εφαρμογής, φυσικά έχει να κάνει με τον όγκο των δεδομένων, αφού όλα φορτώνονται στην μνήμη. Σε περίπτωση με πολύ μεγάλο όγκο δεδομένων, είναι απαραίτητο ενός ισχυρότερου μηχανήματος.

Τα δεδομένα SeaLit που έχουμε στην διάθεση μας και χρησιμοποιούνται στην εφαρμογή έχουν μέγεθος 100MB και ο αριθμός των triples(subject-predicate-object) είναι 104936.   
Για αυτό το μέγεθος του dataset, το πρόγραμμα, στο μηχάνημα που έγινε η υλοποίηση, είναι απόλυτα αποδοτικό.

Ο μέσος χρόνος που χρειάζεται το σύστημα, για να:

* Τρέξει η εφαρμογή, είναι ~12s
* Γίνει η πλοήγηση των διαφορετικών σελίδων της εφαρμογής, είναι ~5s.

# Troubleshooting

* + Read .trig extension files
  + Follow path to find the exact information about a column(label)
  + Πως να επιλέξουμε την γραμμή που θα εμφανίσει τα περισσότερα αποτελέσματα για κάθε row
  + Decoding - Encoding

# Επίλογος

## Βασικά συμπεράσματα

## Αδυναμίες

* Περιορισμένο scalability(επεκτασιμότητα). Κατά την εκκίνηση του προγράμματος, όλα τα δεδομένα φορτώνονται στην μνήμη. Κάτι που σε τεράστιο όγκο δεδομένων, (στην προκειμένη περίπτωση αφορά των αριθμό των τριπλετών που φορτώνονται) είναι βέβαιο ότι είτε δεν θα φορτώσουν καθόλου τα δεδομένα, είτε θα πάρει πολύ χρόνο.

## Μελλοντική εργασία (Βελτιστοποίηση εφαρμογής – Βελτιώσεις)

Οι δυνατότητες που μπορούν να γίνουν στον μέλλον είναι πολυάριθμες με κυρίαρχες τις εξής:

* Κουμπί λήψης, τόσο των δεδομένων του πίνακα και κάθε καταχώρησης ξεχωριστά, όσο και των διαγραμμάτων
* Διασύνδεση των κατηγοριών, με σκοπό την μετάβαση από μια κατηγορία σε άλλη, όταν βρισκόμαστε στην σελίδα που εμφανίζονται περισσότερες πληροφορίες μιας καταχώρησης του πίνακα.
* Επέκταση του navbar, με περισσότερες επιλογές
* Βελτίωση της ταχύτητας της εφαρμογής, για περιπτώσεις μεγάλου όγκου δεδομένων
* Δημιουργία επιπλέον διαγραμμάτων όπως διαγράμματος πίτας(pie chart), που θα μπορούν να δώσουν πιο κατανοητή πληροφορία για κάποια στήλη μιας κατηγορίας

# Αναφορές / Πηγές

* Πληροφορίες από τις διαλέξεις του μαθήματος «ΗΥ-561 Διαχείριση Δεδομένων στον Παγκόσμιο Ιστό» μέσω της πλατφόρμας elearn.uoc <https://elearn.uoc.gr/user/view.php?id=13674&course=4510>
* Επίσημη σελίδα Αpache Jena για την διαμόρφωση του API <https://jena.apache.org/tutorials/rdf_api.html>
* Δεδομένα SeaLit   
  <https://www.ics.forth.gr/isl/fast-cat#usecase>

# Παραρτήματα

1. <https://www.ag-grid.com/javascript-data-grid/reference/>
2. <https://www.chartjs.org/docs/latest/getting-started/>
3. <https://getbootstrap.com/docs/5.3/getting-started/download/>