

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](#_Toc170323955)

[Περίληψη 5](#_Toc170323956)

[Υπόβαθρο και Πλαίσιο Εργασίας 6](#_Toc170323957)

[Το μοντέλο δεδομένων RDF 6](#_Toc170323958)

[URI 6](#_Toc170323959)

[SPARQL 6](#_Toc170323960)

[Το έργο SeaLiT και τα δεδομένα του 7](#_Toc170323961)

[Τεχνολογίες 8](#_Toc170323962)

[Γλώσσα Προγραμματισμού 8](#_Toc170323963)

[Βιβλιοθήκες που χρησιμοποιήθηκαν: 8](#_Toc170323964)

[AG Grid: 8](#_Toc170323965)

[Λειτουργίες συστήματος 9](#_Toc170323966)

[Δεδομένα εφαρμογής 10](#_Toc170323967)

[Διεπαφή χρήστη (User Interface) 11](#_Toc170323968)

[Μοντέλο Διαμόρφωσης(Configuration model) 18](#_Toc170323969)

[Στα αρχεία που έχουν δημιουργηθεί στο παρόν project, υπάρχει εκείνο που αφορά την διαμόρφωση των ρυθμίσεων (configuration properties file : *config.properties*) 18](#_Toc170323970)

[Ιδιότητες αρχείου διαμόρφωσης 18](#_Toc170323971)

[Σχεδιασμός αρχείου διαμόρφωσης 18](#_Toc170323972)

[Παράδειγμα αρχείου διαμόρφωσης 18](#_Toc170323973)

[Εγκατάσταση Συστήματος 23](#_Toc170323974)

[Απαιτήσεις Συστήματος 24](#_Toc170323975)

[Troubleshooting 24](#_Toc170323976)

[Επίλογος 25](#_Toc170323977)

[Βασικά συμπεράσματα Αδυναμίες Μελλοντική εργασία (Βελτιστοποίηση εφαρμογής – Βελτιώσεις) 25](#_Toc170323978)

[Αναφορές / Πηγές 25](#_Toc170323979)

[Παραρτήματα 25](#_Toc170323980)

# Λίστα εικόνων

[Εικόνα 1: Αρχική σελίδα εφαρμογής 11](#_Toc170322431)

[Εικόνα 2: Σελίδα εμφάνισης μιας κατηγορίας 12](#_Toc170322432)

[Εικόνα 3: Εμφάνιση διαγραμμάτων columns μιας κατηγορίας 13](#_Toc170322433)

[Εικόνα 4: Δυνατότητα εμφάνισης πολλαπλών διαγραμμάτων την ίδια στιγμή 14](#_Toc170322434)

[Εικόνα 5: Σελίδα με περισσότερες πληροφορίες μιας καταχώρησης 15](#_Toc170322435)

[Εικόνα 6: Ένα URI, μπορεί να κληθεί από ένα πεδίο πολλαπλές φορές 16](#_Toc170322436)

[Εικόνα 7: Αποκλειστικά βασικά στοιχεία της εφαρμογής 18](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170322437)

[Εικόνα 8: Πεδίο με αριθμό των κατηγοριών των οντοτήτων 19](#_Toc170322438)

[Εικόνα 9: Στοιχεία που αφορούν κάθε κατηγορία 19](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170322439)

[Εικόνα 10: Παράδειγμα με στοιχεία κάθε κατηγορίας 19](file:////Users/alevas/Documents/Thesis/ProWeb/ThesisReport_AlexandrosVasilakis_3612.docx#_Toc170322440)

[Εικόνα 11: Σελίδα χωρίς την παράμετρο ΟPTIONAL στο query που δίνει τα αποτελέσματα 20](#_Toc170322441)

[Εικόνα 12: Σελίδα με την παράμετρο ΟPTIONAL στο query που δίνει τα αποτελέσματα 21](#_Toc170322442)

[Εικόνα 13: Σελίδα με περίπτωση που δεν έχουν όλες οι μεταβλητές τιμή 22](#_Toc170322443)

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. 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.

*Διαχείριση RDF δεδομένων*

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

[Apache Jena](https://jena.apache.org/) - Apache Jena is an open source Semantic Web framework for Java. It provides an API to extract data from and write to RDF graphs.

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)

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) 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, 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)

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 βιβλιοθήκες javascript.

### AG Grid:

Για την δημιουργία του πίνακα, χρησιμοποιήθηκε η παρούσα βιβλιοθήκη μιας και οι λειτουργίες που παρέχει είναι πολυάριθμες.

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

Με τα δεδομένα που χρησιμοποιήθηκαν εμφανίζει:

* + Ανάλογα με την οντότητα που θα επιλεγεί

Persons

Locations

Voyages

Ships

* 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, είχε επιλεγεί αρχικά το AG Charts, ωστόσο στην συνέχεια, λόγω πιο εύκολης και δωρεάν χρήση της, επιλέχτηκε η βιβλιοθήκη Chart JS.

Through .trig files, I found more information about a single record(That’s how I add them)

# Λειτουργίες συστήματος

1. 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:

* Αν περιλαμβάνουν κείμενο, τότε ο χρήστης μπορεί να ψάξει με βάση κάποια αλφαβητικό χαρακτήρα
* Αν περιλαμβάνουν ψηφία, τότε ο χρήστης μπορεί να ψάξει με βάση κάποιο αριθμητικό χαρακτήρα
* Αν περιλαμβάνουν ημερομηνίες, τότε ο χρήστης μπορεί να ψάξει τοποθετώντας συγκεκριμένα την χρήση ενός συγκεκριμένου template που αφορά ημερομηνίες

1. 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.

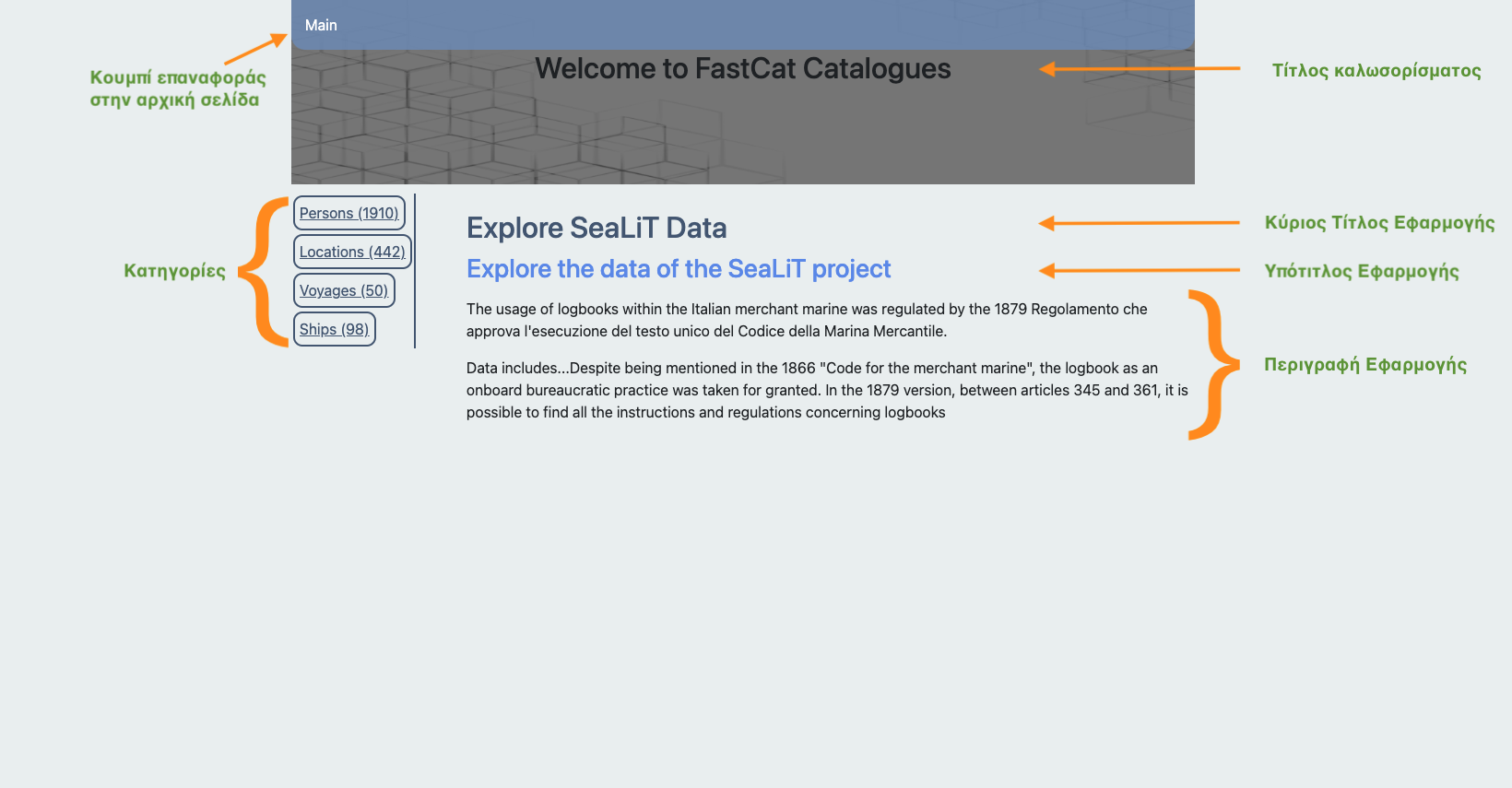
# Δεδομένα εφαρμογής

Τα δεδομένα που χρησιμοποιήθηκαν στην παρούσα εφαρμογή αφορούσαν πληροφορίες από την Ναυτική Ιστορία. Πιο συγκεκριμένα:

* Τοποθεσίες
* Οργανισμοί
* Ονόματα
* Πλοία
* Εγγραφές (Το αρχείο με την περισσότερη και πιο ουσιαστική πληροφορία)
* Λεξιλόγια

# Διεπαφή χρήστη (User Interface)

Κατά την είσοδο στην εφαρμογή βλέπουμε την αρχική σελίδα:

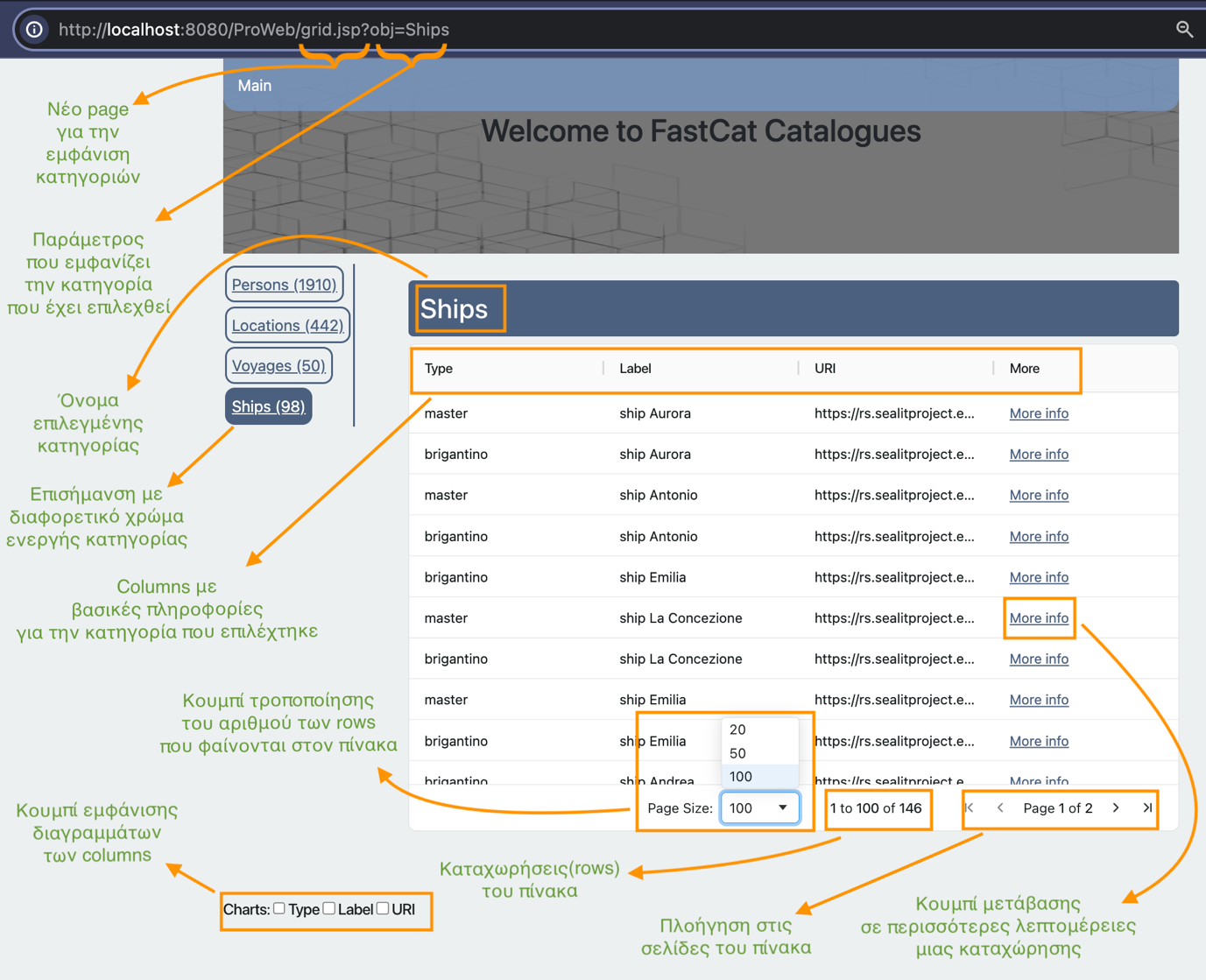


Εικόνα 1: Αρχική σελίδα εφαρμογής

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 2: 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 που επιλέχτηκαν(μέσω του query)
      * 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 3: Εμφάνιση διαγραμμάτων columns μιας κατηγορίας

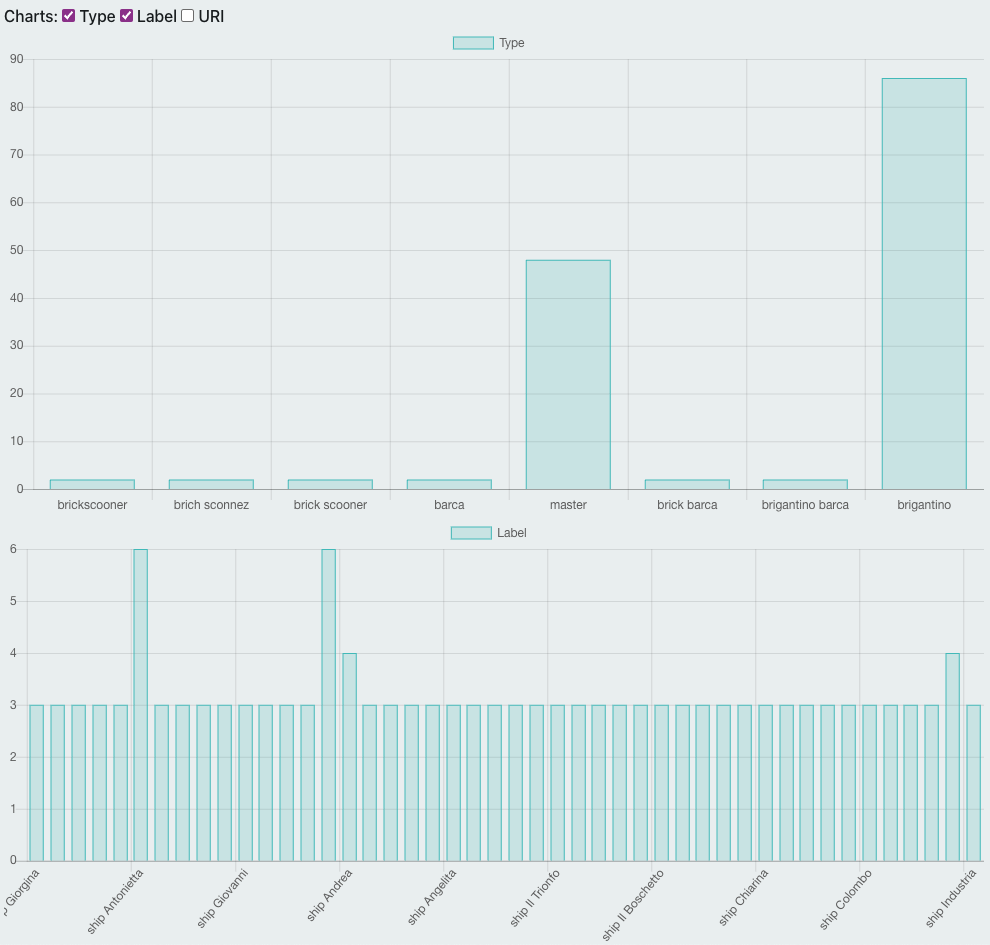
Σε αυτά ο χρήστης μπορεί να επιλέξει να δει το διάγραμμα που επιθυμεί.

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

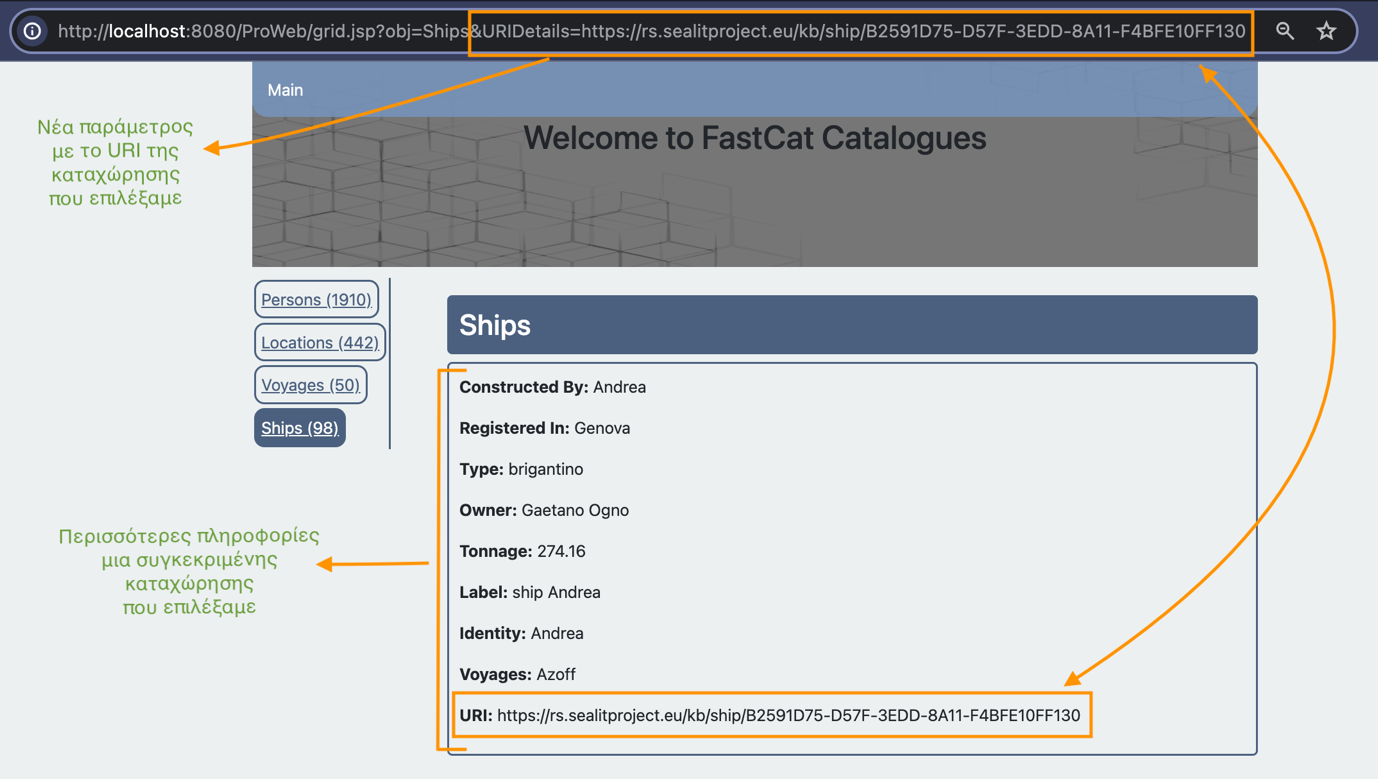
Η επιλογή του παραπάνω διαγράμματος έγινε επειδή αποτελεί ένα πολύ καλό παράδειγμα μιας και δίνει χρήσιμα αποτελέσματα.

Επιπλέον μπορούν να εμφανιστούν και πολλαπλά διαγράμματα την ίδια στιγμή



Εικόνα 4: 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.

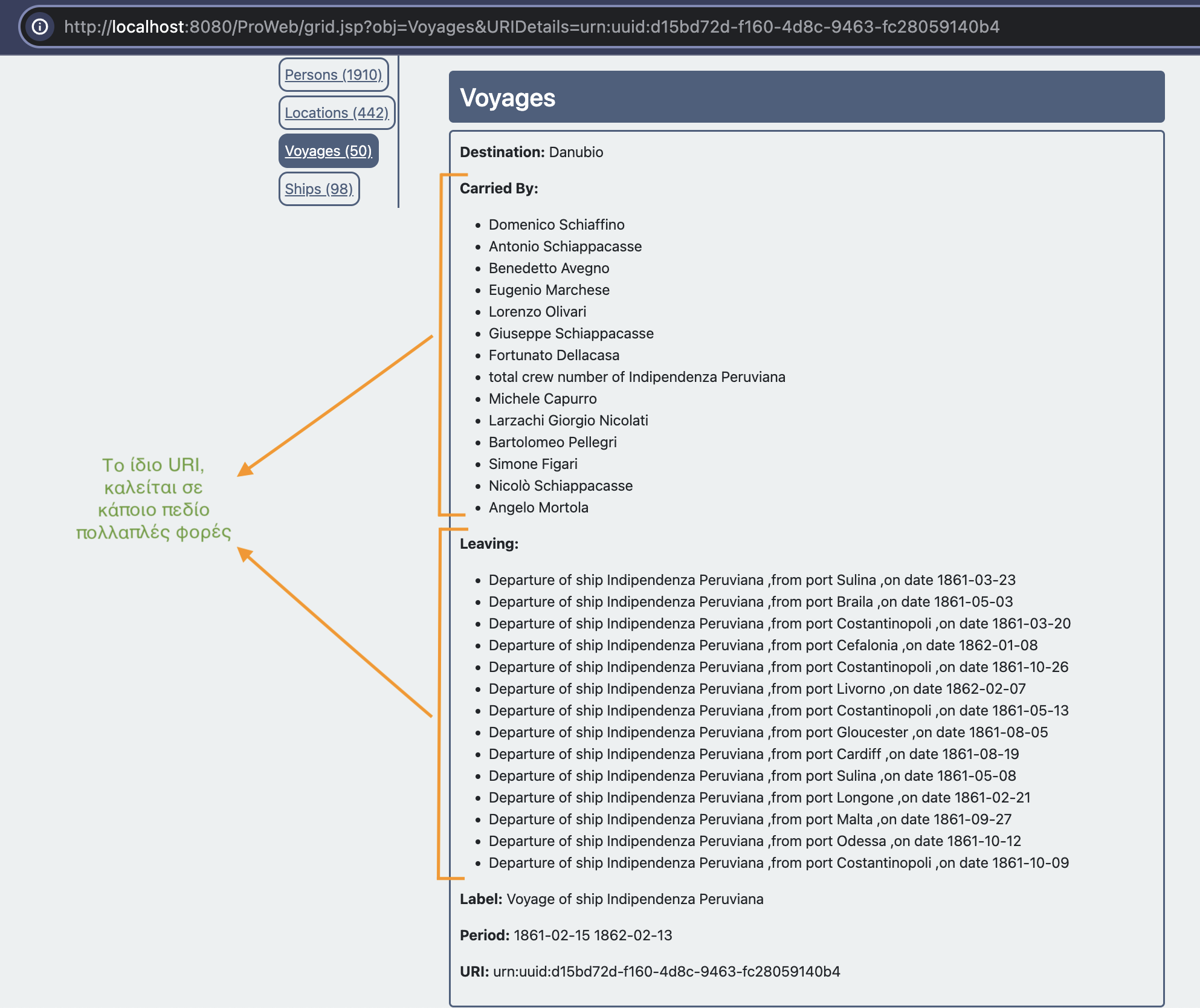


Εικόνα 5: Σελίδα με περισσότερες πληροφορίες μιας καταχώρησης

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

# 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

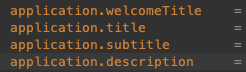
Το configuration file χρησιμοποιείται για δοκιμή, επιβεβαίωση, διόρθωση και βελτιστοποίηση των λειτουργιών της εφαρμογής που δημιουργήθηκε.

### 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 7: Αποκλειστικά βασικά στοιχεία της εφαρμογής

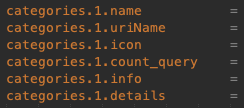
Ύστερα υπάρχει :

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



Εικόνα 8: 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 9: 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 10: Παράδειγμα με στοιχεία κάθε κατηγορίας

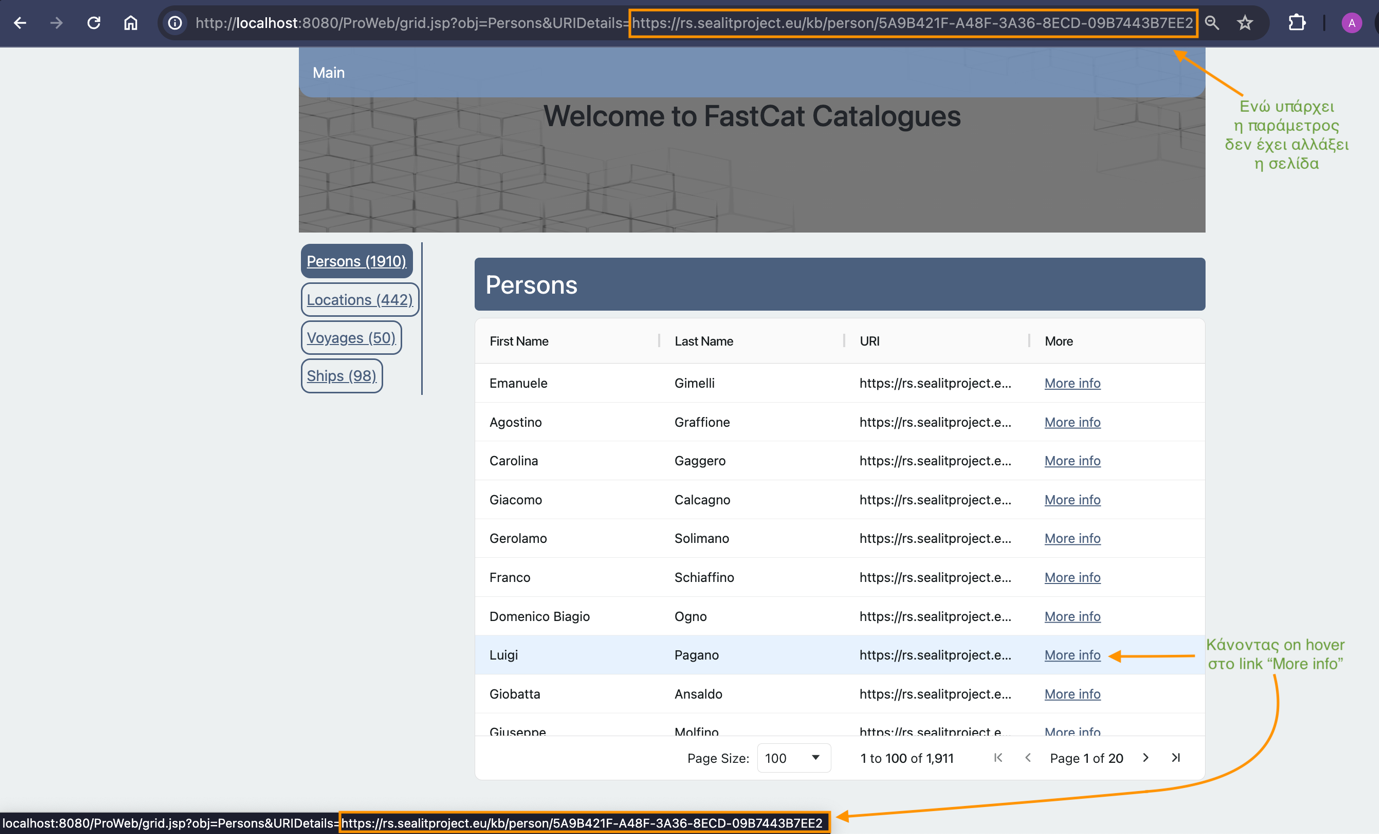
Όπως φαίνεται στην εικόνα από πάνω, υπάρχουν περιπτώσεις που η πληροφορία που χρειαζόμαστε, δεν δίνεται αμέσως, αλλά χρειάζεται να ακολουθήσουμε κάποιο path για να την πάρουμε

**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.

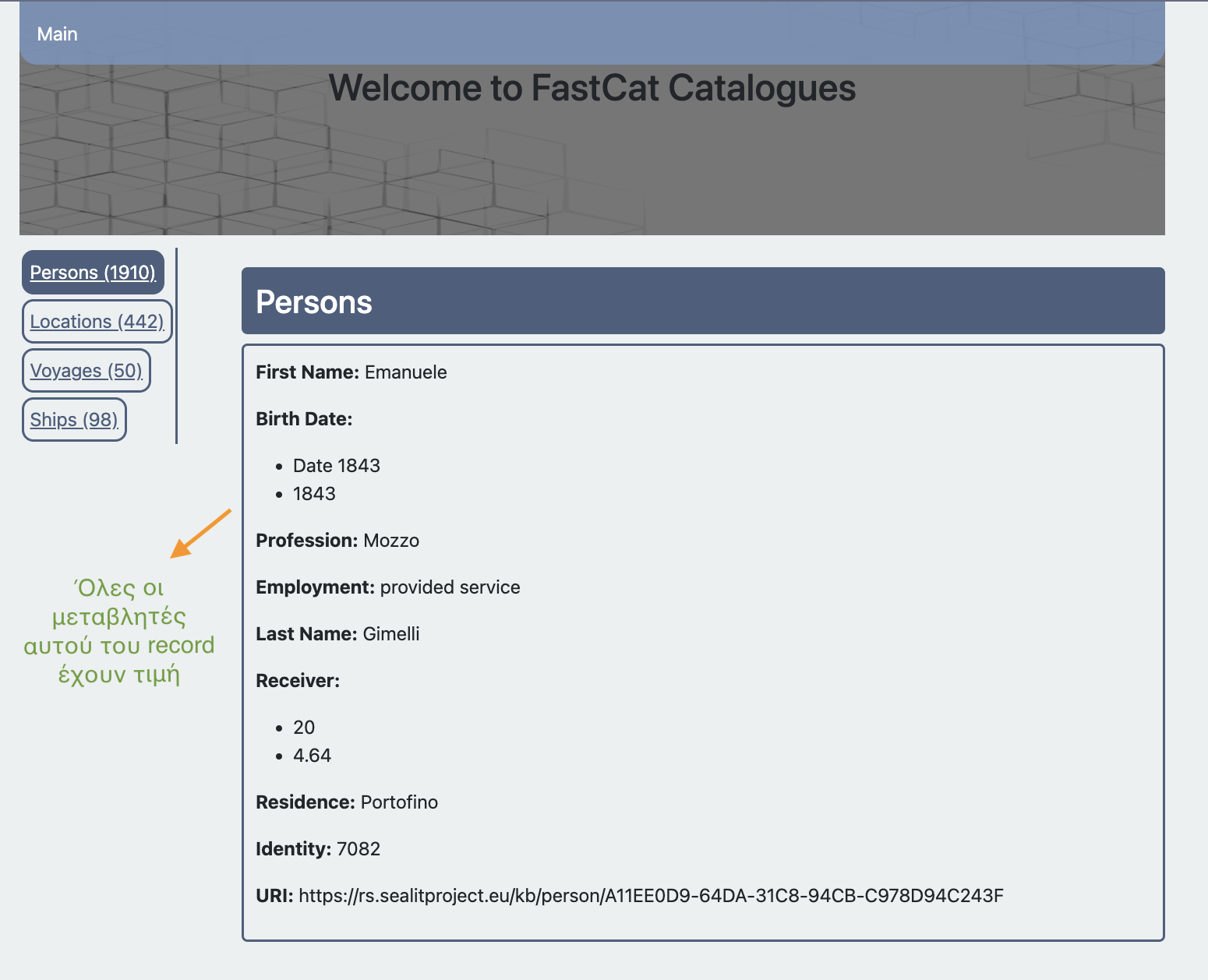


Εικόνα 11: Σελίδα χωρίς την παράμετρο ΟPTIONAL στο query που δίνει τα αποτελέσματα

**Με την χρήση της παραμέτρου OPTIONAL:**

Επιλύεται το πρόβλημα και έχουμε τα επιθυμητά αποτελέσματα.

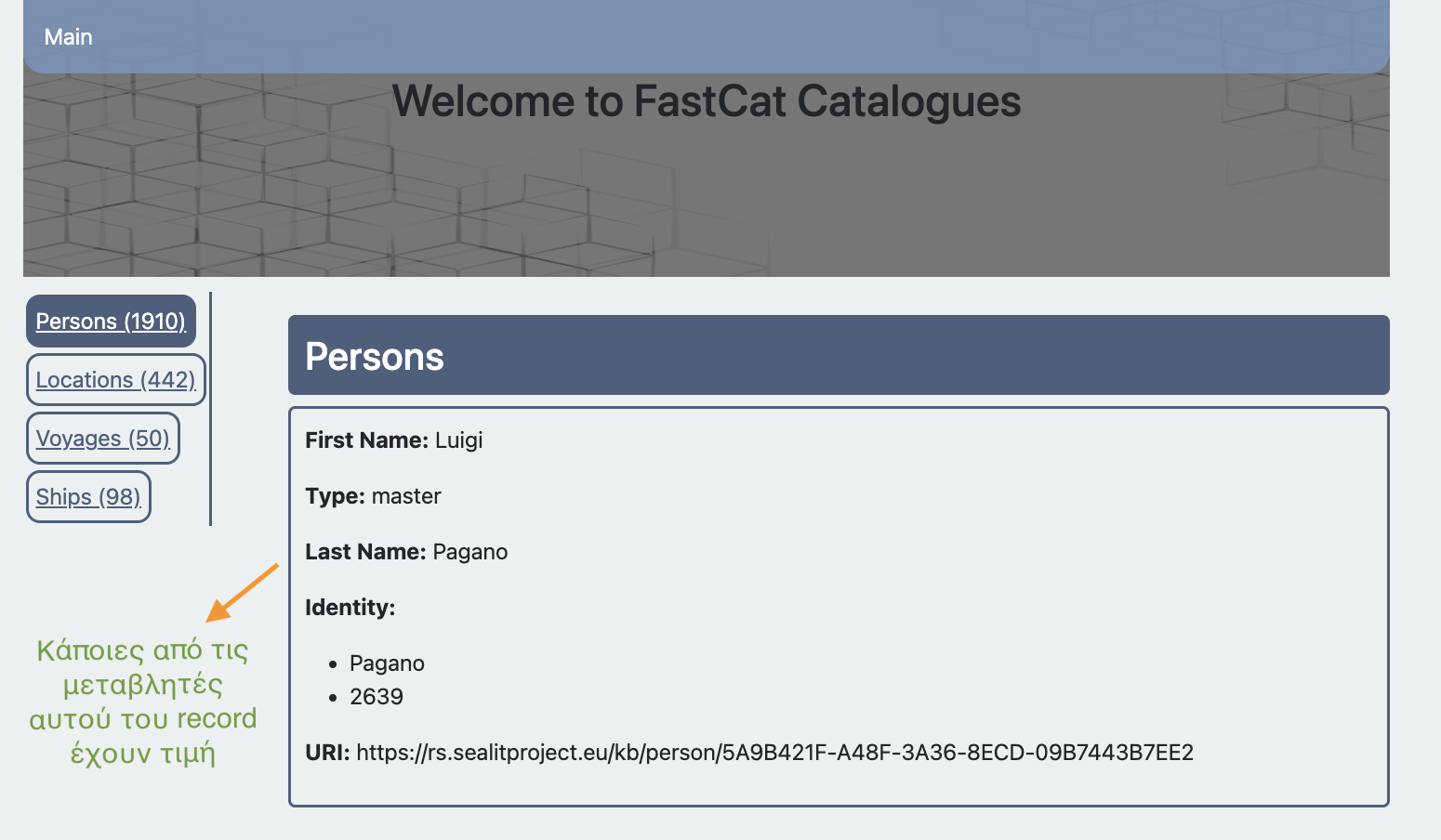
1η περίπτωση:



Εικόνα 12: 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 13: 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)
* IDE της επιλογής του προγραμματιστή (η δημιουργία του project έγινε με NetBeans έκδοσης 12.5)

Για την δημιουργία ενός Java Web Application έχει γίνει ήδη η διαμόρφωση του

* pom.xml αρχείου(Project Object Model), που χρησιμοποιείται για τον ορισμό dependencies και plugins, και διαμόρφωση της Maven εφαρμογής ενός JAVA Project

Σε αυτό θα βάλουμε όλα τα απαραίτητα dependencies και plugins, προκειμένου να λειτουργήσει ο server που θα αποφασίσουμε να χρησιμοποιήσουμε.

Αφού γίνει “Run File” στο main αρχείο του project(index.jsp), εκτελείται ο Apache Tomcat (server). Προκειμένου να γίνει η αυθεντικοποίηση απαιτείται από τον Τomcat Manager:

* + username: admin
  + password: admin

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

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 |  |

# 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

## 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

# 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/>